ABSTRACT

Dissertation: INDUSTRIAL FLEXIBILITY IN THEORY AND PRACTICE

by: Matthew J. Reindorp
Ph.D., 2009

Dissertation advisor: Professor Michael Fu
Dissertation co-advisor: Professor Manu Goyal

At the heart of any decision problem is some degree of "flexibility" in how to act. Most often, we aim to extract greatest possible value from this inherent flexibility. The three essays compiled here are aligned with this same general aim, but we have an important secondary concern: to highlight the value of flexibility itself in the various situations we study. In the first essay, we consider the timing of an action: when to replace obsolete subsystems within an extensive, complex infrastructure. Such replacement action, known as capital renewal, must balance uncertainty about future profitability against uncertainty about future renewal costs. Treating renewal investments as real options, we derive the unique, closed-form optimal solution to the infinite horizon version of this problem and determine the total present value of an institution's capital renewal options. We investigate the sensitivity of the solution to variations in key problem parameters. The second essay addresses the promising of lead times in a make-to-order environment, complicated by the need to serve multiple customer classes with differing priority levels. We tackle this problem with a "model free" approach: after preparing a discrete-event simulation of a make-to-order production system, we determine a policy for lead time promising through application of a reinforcement learning algorithm. The third essay presents an empirical analysis of new product launches in the automotive industry, showing that manufacturing flexibility is one key indicator of superior productivity during launch. We explore the financial dimensions of the apparent productivity differences and show that the use of flexible manufacturing increases an automobile plant's likelihood of being chosen to host a new product launch.

INDUSTRIAL FLEXIBILITY
IN
THEORY AND PRACTICE

by

Matthew J. Reindorp

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2009

Advisory Committee:
Professor Michael Fu, Chair/Advisor
Professor Manu Goyal, Co-Advisor
Professor Michael Ball
Professor John Baras
Professor Shreevardhan Lele

ACKNOWLEDGMENTS

I am distinctly grateful to my advisor, Professor Michael Fu, for inspiration and guidance of the research presented in chapters 2 and 3 of this dissertation. I am likewise grateful to Professor Manu Goyal for suggesting and advising the research presented in chapter 4.

My sincere thanks go to the other members of my dissertation committee - Professor Michael Ball, Professor John Baras, and Professor Shreevardhan Lele - for their time and constructive comments, and to Professor Serguei Netessine for helpful discussions of the work in chapter 4.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 DISSERTATION OVERVIEW

The concept of "flexibility" is a common point of return for the three essays in this dissertation. Since some element of flexibility lies at the heart of any decision problem, there is nothing new nor hardly more important than showing how to extract greatest possible value from it. The studies here attempt, however, to add a little to this essential concern, by recognizing the value of flexibility itself in the various situations we study. This aim is akin to the classic sensitivity analysis of mathematical programming, which reveals the marginal value of each resource that is needed for the task at hand.

While sensitivity analysis is generally applicable to static, deterministic optimization problems, flexibility is the analogous concern in dynamic, stochastic settings. Faced with one or more elements of uncertainty, a more flexible position is on that "leaves available a larger set of future positions at any given level of cost" (Jones and Ostroy, 1984). This

larger set of positions enhances the decision maker's ability to take advantage of new information. Correspondingly, any benefit from flexibility is to be understood in the sense of average or expected returns.

Just as the marginal value of a resource in a static optimization problem may be zero or close to it, the value of flexibility is likely to be small in situations where the variance in key processes is limited. Moreover, we make no assumption that flexibility will be the most important element in our analyses: other problem parameters - including ones that may be beyond immediate managerial control - and the quality of the solution approach itself may have greater impact on overall returns. Nevertheless, as long as it is present, flexibility is a dimension that warrants exploration in dynamic contexts.

The sections below provide a brief summary of the chapters to follow, highlighting the form of flexibility that appears in each case and the key findings.

1.2 Capital Renewal as a Real Option

An archetypal context for flexibility is the question of timing for an investment decision. In chapter 2 we consider the problem of timing for *reinvestments*. For large, capital intensive institutions, reinvestment in existing infrastructure is a recurrent necessity: obsolescence of key subsystems entails higher operational costs and renders a business less attractive to discerning clients. Reinvestment policies must, however, balance uncertainty about future profitability against uncertainty about future renewal costs: acting too soon or too late may diminish the value that renewal activity can create.

As with initial investment decisions, real options analysis provides a natural framework for the modeling and analysis of the reinvestment problem. Option pricing in this context is synonymous with the valuation of flexibility in timing. Treating renewal investments as real options, we derive an optimal capital renewal policy for an institution subject to exogenous price levels. We calculate the total present value of an institution's capital renewal

options and show it is increasing in the variance of infrastructure evolution. Moreover, we illuminate the impact of key policy parameters on the optimal policy: specifically, cost scales, price trends, infrastructure decay rates, and variability in the infrastructure process.

1.3 Dynamic Lead-time Promising

In addition to the constraints imposed by inventories and production capacity, the need to serve multiple customer classes with differing priority levels can put great strain on a firm's ability to quote accurate lead times. While a mixed integer programming model allows lead times to be fixed for batches of accumulated demands (Chen et al., 2002), approximate dynamic programming methods hold the promise of a policy based solution, where lead times can be quoted in real time. Chapter 3 proposes a "model free" approach to this problem: after preparing a stochastic simulation of a make-to-order production system, we determine a policy for lead-time promising through application of a reinforcement learning algorithm, specifically, relative Q-learning for average reward (Gosavi, 2003).

Our simulation studies show how average rewards tend to vary with the likelihood of supply chain disruptions, the cost of production overtime, and production capacity constraints. The dimension of flexibility lies here in the range of lead time offers that the firm can make to its customers, and we find that in cases of higher demand variance, the value of flexibility can be as important as the quality of our policy generation algorithm.

1.4 Automotive Launch Productivity

Product flexibility in automobile manufacture enables the dramatic growth in model variety that the industry has seen in the last 20 years, but the same flexible manufacturing methods appear to have a negative impact on overall productivity (Van Biesebroeck, 2007). Our analysis in chapter 4 suggests that flexibility may nonetheless be beneficial for productivity in the context of a new product launch. Our results are based on eight years of recent

data from North American automotive plants. Besides flexible manufacturing practices, prior experience with similar products proves to be an important factor for productivity performance during launch. Corresponding to these findings, we show that manufacturers appear more likely to locate a launch at plants that have flexible production capabilities and/or relevant prior experience.

CHAPTER 2

CAPITAL RENEWAL AS A REAL OPTION

2.1 MOTIVATION

Large businesses and institutions often possess a substantial infrastructure on which their economic activity is deeply dependent. Capital intensive industries such as energy production, telecommunications, and transportation provide obvious examples of this dependence, but it also occurs in enterprises that are essentially identifiable with their fixed assets, such as hotels, museums, or universities. In all these cases, the condition of infrastructure is an important managerial concern: aging facilities entail higher operational costs and concomitantly render the business less attractive to discerning clients. Nevertheless, it is rarely either practical or desirable to replace an entire infrastructure; instead, problems must be addressed by replacement of subsystems, such as roofs, HVAC equipment, pipelines, pavements, or storage tanks. This process, known as "capital renewal" or "recapitalization," is key to restoring the economic value of infrastructure. Capital renewal activities should

therefore never be unnecessarily postponed, but the size of the expenditures involved usually also means that they should not be undertaken prematurely. Optimal timing of capital renewal strikes a balance between these two considerations and provides maximal return on renewal investments.

The problem of timing replacement or maintenance for a deteriorating system - which could be anything from a piece of office equipment to a military aircraft - has been extensively studied by economists and management scientists (Wang, 2002). Capital renewal presents a subtle but important difference from these traditional problems, however, in that it intentionally effects only a partial replacement of the system at hand. This is approach is peculiar to extensive, complex infrastructures, where full replacement may be excluded due to prohibitive cost or the need to preserve some historical continuity, yet regular maintenance of infrastructure subsystems cannot prevent their increasing technological obsolescence. As with full replacement or maintenance, capital renewal aims to restore the economic value of infrastructure, but it does so by replacing only the subset of the entire infrastructure that is no longer economically viable.

The practical distinction between traditional replacement or maintenance problems and the capital renewal scenario entails additional requirements in the modeling of the latter. Besides the common concern with increasing operating costs, renewal timing must recognize that obsolescing subsystems imply an increasing potential replacement cost, due to the greater number or extent of subsystems that will ultimately need replacement. Moreover, since operating and replacement costs are surely dependent on current and future price levels, a capital renewal model should feature an exogenous price process, which in turn conditions the revenues generated by either the current or a renewed infrastructure. In order to examine this complex interplay of revenue and costs across potentially unlimited renewal cycles, something other than a model of *pure renewal* is needed. Pure renewal implies that the entire system is restored to a mathematically identical copy of its initial instance. In order to reflect the financial exigencies of long term planning, a capital renewal

model must distinguish the *infrastructure system* from the background reality of price levels, since the latter may not be at all affected by the renewal of the former. This distinction can be achieved through the use of separate price and system processes, which are related through appropriate functional forms.

Existing literature on replacement or maintenance problems has hardly begun to admit the distinction between system renewal and price continuation that is essential for a capital renewal model. The field of real options analysis provides the closest approaches to date. The real options perspective has greatly influenced the theory - and to some extent, the practice - of capital budgeting in the last 25 years (McDonald, 2006). While early work in this area was primarily concerned with the opportunity to invest in a new project, where exogenous, stochastic price levels condition the potential return on investment, subsequent research has used the same framework to show how variability in costs also conditions the potential return on replacement investments. Nevertheless, the models either envision pure renewal, where the key cost variable is returned to at a nominal level across all possible replacements, or they confine the analysis to a single replacement decision. The capital renewal model presented here will also utilize real options theory, but we shall fully allow for the impact of price on the system renewal decisions in an infinite horizon context.

The essence of the real options approach is to make explicit allowance for the value of flexibility in the timing of an investment decision. If the current expected net present value (NPV) of the investment is less than the value inherent in the mere "option" to invest, delaying the investment will be preferable to acting immediately. When applied to capital renewal decisions, real options analysis allows us to make analogous provision for the value of flexibility: reinvestment should not be undertaken unless the expected present value of the renewed infrastructure, minus current the cost of renewal, is at least equal to the *combined* value of the existing infrastructure and the option to renew it. Using this key condition, we can derive an institution's optimal capital renewal policy in an infinite horizon context, and specify the expected value created through the series of optimally

executed renewals.

The development of this analysis is organized as follows. In section 2.2 we take a closer look at trade and academic literature that addresses capital renewal or related problems. Section 2.3 formulates and illustrates our capital renewal model. Section 2.4 analyzes the model in a deterministic context and shows the state-based version of the optimal renewal policy, in anticipation of the analysis of the stochastic version of the model. Section 2.5 presents the stochastic analysis using the real options approach, resulting in closed form expressions for the optimal renewal policy and the total expected value of the renewal activity. Section 2.6 shows some numerical studies of the optimal policy's response to its various parameters and looks at the performance of the infinite horizon solution in finite-horizon contexts. Section 2.7 gives concluding remarks and thoughts for further research.

2.2 LITERATURE SURVEY

The majority of writing on the subject of capital renewal appears to lie in trade publications, commercial websites, or governmental studies. This literature is generally of a discursive nature. Westfall (2001), for example, in a white paper for the company Tradeline, Inc., gives an overview of the capital renewal policies in use at IBM, Du Pont, and Freddie Mac. June (2003) reviews the need for reinvestments in academic facilities. A discussion paper by the Conference Board of Canada highlights the importance of renewal investments for the Canadian and U.S. natural gas industry (Roland George, Purvin & Gertz, 2004). The National Research Council of Canada investigated the domain of asset management and its findings are available in the proceedings of several conferences (Vanier, 2000).

Nevertheless, detail of some quantitative approaches to renewal is available. Biedenweg and Hutson (1982) and Kaiser (1984) argue for the use the life-cycle estimates in budgeting facilities renewal and replacement expenses. This approach is highly case-specific, however, requiring that each facility be analyzed in terms of subsystems, each of which should

receive its own life cycle and replacement cost estimate; the resulting data are used to make projections for renewal needs over a five-year period. Rush (1991) introduces the Facility Condition Index (FCI), a more general approach to facilities management. FCI is defined[1] as the ratio of estimated cost of remedying any current deficiencies (CD) in a facility to estimated current replacement value (CRV) of the facility,

$$\text{FCI} = \frac{\text{CD}}{\text{CRV}}, \qquad \text{CD},\ \text{CRV} \geq 0. \tag{2.1}$$

FCI is widely used as a capital planning tool (Amekudzi and McNeil, 2008). Most U.S. federal agencies base their facilities management plans on the index (Cable and Davis, 2005). It is a "current" measure that can be updated as needed, and its dimensionless nature permits comparisons and benchmarking across facilities or institutions. Thus FCI can serve as an objective for maintenance and renewal considerations. Consider, for example, Figure 2.1 and the accompanying text, both drawn from a recent capital planning report of the National Aeronautics and Space Administration (NASA, 2008).

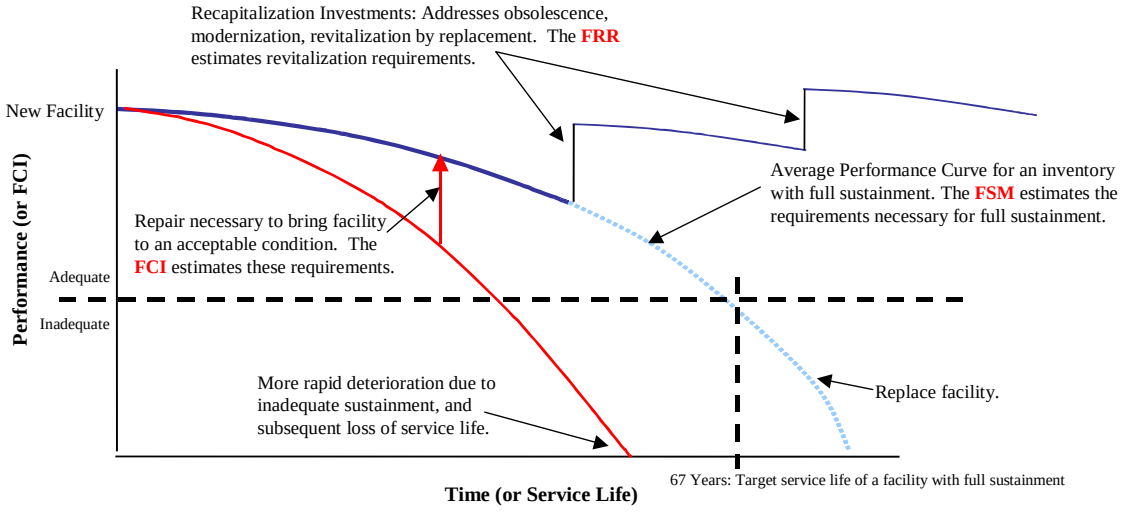


Figure 2.1: Facility Life Cycle Performance Curve, NASA Capital Plan, April 2008

[1]The definition given in (2.1) entails FCI $\in [0, \infty)$, but the index is often stated in the form $1 - \text{FCI}$, with range $[1, -\infty)$.

> On the Y-axis is the Performance or Facility Condition Index (FCI) of the building; on the X-axis is Time or the Service Life of the Facility. The first curve (red) represents a new facility that has not had the proper repairs and in which subsequent deterioration foreshortens the life of a facility. The second curve (blue) represents a facility that has had the proper repairs (vertical red arrow) to sustain the FCI of the facility to the target age of 67 years. The amount needed to sustain facilities for their useful life is called the Facility Sustainment Metric (FSM). *The third and following partial curves (purple) represent recapitalization of investments in which modernizations or repair-by-replacement has occurred to extend the facility beyond the average target age. The Facilities Revitalization Rate (FRR) (black vertical bar) is the amount needed to revitalize facilities to this standard*[2].

NASA's capital renewal model thus appears equivalent to the Facility Revitalization Rate (FRR) that characterizes the discontinuous curves in the diagram. Nevertheless, the FRR is not a decision rule, but a consequence of funding policies: according to Cable and Davis (2005), FRR is "an indication of how often a facility is completely revitalized. It is calculated by dividing CRV by annual facility revitalization funding." There does not seem to be here any optimization of the revitalization trajectory with respect to financial measures. In other words, the problem of setting a *renewal policy* that will minimize the expected present value of capital expenditures (or, in the context of private enterprise, to maximize NPV of capital investments) appears not to be addressed. In particular, the reduction of capital renewal policy to an annual funding allowance suggests that an institution would be less able to respond to deviations of facility conditions from the mean trajectory, which might advance or postpone the need for capital investments.

Turning to research literature, we find that little explicit attention has been given to the capital renewal problem. This apparent lack of academic interest is surprising, considering the close relation of the problem to full replacement or preventative maintenance, which are classic studies in business economics. Work on these problems predates Hotelling (1925), who nevertheless provides perhaps the first consistent analysis of the depreciation of an asset, by assuming interdependence between the operating costs of a production machine

[2]Emphasis added

and its value. Terborgh (1949) subsequently suggests that machine operating costs may be taken to grow at a constant rate, greatly facilitating the determination of replacement intervals.

The earliest examples of stochastic replacement models appear to come from Weiss (1956) and Welker (1959). These and other works of the same period are surveyed by McCall (1965). Most are concerned with systems that only have two possible states, operative or failed, and the replacement policies derived for these are temporal rules. In contrast, Derman (1963) analyzes a model in which the state S of the system can move through $n + 1$ consecutive levels, with 0 and n denoting new and failed, respectively. The optimal replacement policy for such a system is shown to be a control limit: for some computable $i \in \{0, \ldots, n\}$, replace when $S \geq i$.

Although temporal rules remain more common, a distinct stream of research has pursued the idea that replacement decisions can depend on some observable state of the underlying system. Wang (2002) describes these as "failure limit policies" and summarizes the key features of typical models in the class. Other notable examples and extensions of the approach include the work by Rust (1987), Hopp and Nair (1994), and Hartman (2000). Despite the different perspectives covered by these works, they all share one important point: the uncertainty in the system is restricted to some measure of cost. This restriction means that no allowance is made for the evolution of prices in the system environment. In the context of capital renewal, recognition of price as a variable is essential, since changes price levels may impact system revenue as well as replacement costs.

With price included as a variable, profit becomes the appropriate objective for a capital renewal decision. From this perspective, capital renewal is an investment timing problem. The analysis of investment decisions under uncertainty has been revolutionized through the introduction of the real options perspective, of which McDonald and Siegel (1986) provide one of the earliest examples. In contrast to traditional discounted cash flow analyses of investment opportunities, which requires only that the expected NPV of the asset acquired

be positive, real options analysis suggests that investment should only be made if the expected NPV of the opportunity is at least as great as the value of retaining flexibility over the timing of the decision.

The first study to admit replacement decisions in the real options framework is provided by McLaughlin and Taggart (1992). Their analysis addresses primarily the opportunity cost of refitting an idled plant to produce a different, more profitable product, but they consider also how this cost changes if a plant can be replaced at the end of its useful life. Product prices are exogenous to the decision process, but the cost of any action (refitting an existing factory or building a replacement) is constant across time, unlike in the case of capital renewal.

The work of Dixit and Pindyck (1994) remains an invaluable reference for the study of real options theory. As with most early works in the field, however, the authors are primarily concerned with the decision to investment in a new project, and assume in most cases that the project, once undertaken, produces a fixed output indefinitely. They analyze one scenario in which the lifetime of a project follows a Poisson process, and each project can be replaced at the end of its useful lifetime, thereby entailing an infinite series of investment opportunities. Nevertheless, this reinvestment model differs in three key ways from a capital renewal problem. First, it assumes that the profitability of a project remains unchanged through its lifetime. Second, the costs of reinvestment are constant across the entire infinite horizon. Third, the reinvestment option is not always present, but only comes into existence with the death of the existing project. Each investment decision is therefore analogous to the one required by an entirely new project. Upon expiration of one project, it may be optimal to delay reinvestment if current price levels are not sufficiently high, but management does not otherwise have the opportunity to optimize profits value across the infinite horizon, for example, by reinvesting in a project before it dies[3].

Subsequent work by Mauer and Ott (1995) effectively allows for variable costs in re-

[3]Admittedly, assuming constant profitability and constant reinvestment costs, reinvestment would not anyway be optimal during the lifetime of a project.

placement decisions: although the authors take the purchase price of a new asset to be fixed, their model makes salvage cost a function of current operating cost, so the cost of renewal is effectively state dependent. Nevertheless, the model omits the effect of a truly exogenous price process: the initial operating cost of a replacement system is nominally the same as the original system. This seems implausible, since the horizon of the analysis is infinite and a discount factor is applied to the nominal costs.

Several more recent works present analyses that mirror the capital renewal problem in many key respects. All take an infinite horizon perspective with an exogenously specified discount factor, but each includes some constant factor of revenue or cost that seems inappropriate in that context. The scenario that Yilmaz (2001) considers is certainly one of renewal investment, in that the system is partially faulty and the firm incurs compounding losses until it chooses to fix the fault; but the renewal is only effected once, and the cost of renewal is taken to be constant. In the model of Dobbs (2004), we have an infinite series of entire system replacements, each at constant nominal cost. Adkins and Paxson (2006) also envision an infinite series of system replacements, and allow for uncertainty in revenues as well costs during each interrenewal period, but upon each renewal, revenue and cost revert to their initial nominal levels.

The theoretical approach of our analysis is essentially consonant with these latter studies, but we tailor our model to the specific requirements of the capital renewal problem. We retain the infinite horizon context, but our operating profits and renewal costs are conditioned throughout by an exogenous price process. Renewal activity causes reversion to an initial state, but the financial impact of this is interpreted indirectly, rather than directly in terms of cash flows: renewed infrastructure allows us to realize fully our profit potential, which can only be partially realized through decayed infrastructure.

2.3 MODEL FORMULATION

We construct the capital renewal model from two key random variables. The index P describes the market price of the the firm's product or services (it is assumed to be a price taker). The state variable S describes the profitability of the firm's infrastructure. The evolution of P and S proceeds according to two geometric Brownian motions,

$$dP = \alpha_P P dt + \sigma_P P dz_P,$$
$$dS = \alpha_S S dt + \sigma_S S dz_S.$$

We assume that the price index does not influence the firm's profitability, so dz_P and dz_S are taken to be uncorrelated. The drift coefficient α_P is positive, reflecting the tendency for prices to increase. We take α_S to be negative, however, reflecting the key assumption that aging infrastructure tends to reduce profitability.

The use of a geometric Brownian motion process to describe a price index or the value of some underlying asset is typical of all real options analyses. Its application to the case of an infrastructure process alone is less common, but is needed as a way to distinguish the impact of infrastructure on the firm's profit from the exogenously given price level. The impact of infrastructure on profit is described elsewhere by means of a single geometric Brownian motion with negative drift (*e.g.*, Blazenko and Pavlov, 2004), obscuring the fact that profit may experience nominal growth in the short term, despite decaying profitability, if the growth rate of price levels is greater. In our model, the interaction between price and profitability is determined by other function forms employed for the model, as described next.

First, we assume that the firm's output rate is constant and denote the initial values of P and S by P_0 and S_0. When a firm chooses to renew its infrastructure, the prevailing value of P is unchanged, but S reverts to S_0. The instantaneous profit of the firm, π, is a function of variables P and S. Intuitively, $\pi(P, S)$ is increasing with P and S (but note

that S tends to decrease with time). The cost of infrastructure renewal, k, is also a function of the variables P and S. Intuitively, $k(P, S)$ is increasing with P but decreasing with S (again, S tends to decrease with time, so k tends to increase).

The following functional forms and initial values capture these intuitive trends in profit and renewal cost:

$$\begin{aligned} P_0 = S_0 &= 1, \\ \pi(P, S) &= PS, \\ k(P, S) &= k_0 \frac{P}{S}, \quad k_0 > 0, \end{aligned}$$

where k_0 is a constant of proportionality between potential renewal costs and the profitability level S.

The profit equation follows directly from the definition of profitability as the net fraction of revenue that a firm realizes as profit. For the renewal cost equation, we apply the "common wisdom" that asset deterioration is naturally exponential (Rodney, 2007), entailing exponential increases in potential renewal costs, and the further observation that infrastructure depreciation has a proportionally negative impact on profitability (Blazenko and Pavlov, 2004). Rather than introduce a third variable to the model, we combine these two points and use S as a proxy for the impact of infrastructure condition on renewal costs.

Since there is a positive cost of renewal at $t = 0$, it doesn't make economic sense for the firm to renew immediately. As time passes, however, the profit level of the firm falls increasingly far below what could be generated with renewed infrastructure. Concomitantly, the cost of renewal increases. If renewal is delayed too long, the net value gained from renewal will be less than optimal.

If renewal activity can create positive net value, the number of optimal renewal epochs for the firm is potentially unlimited. Thus we will analyze the renewal problem in the context of an infinite time horizon. All profits and costs will be discounted at rate $\rho > \alpha_P$,

in order to ensure that the firm has a finite NPV at the outset.

In order to visualize the trade off between earlier and later renewal, consider first a deterministic version of the model outline above (*i.e.*, take $\sigma_P = 0$ and $\sigma_S = 0$). Take $k_0 = 1.00$, so the cost of renewal is initially equal to the firm's annual profit. Suppose that P increases at a rate of 4.25% per period, while the the firm's profitability tends to decay at a rate of 2.00% per period. Assuming a discount rate of 6.00%, the discounted level of profit and cost are shown by the blue and red lines in Figure 2.2 below. The black line shows the discounted profit that could be earned at any moment with renewed infrastructure. Thus we can see that the capital renewal problem is to decide when to move from the blue curve to the black curve, thereby incurring the prevailing renewal cost. For example, Figure 2.2 shows a policy of renewal at 15 year intervals. Our objective is to maximize the area under the resulting *discontinuous* profit trajectory, minus the total cost of the renewals.

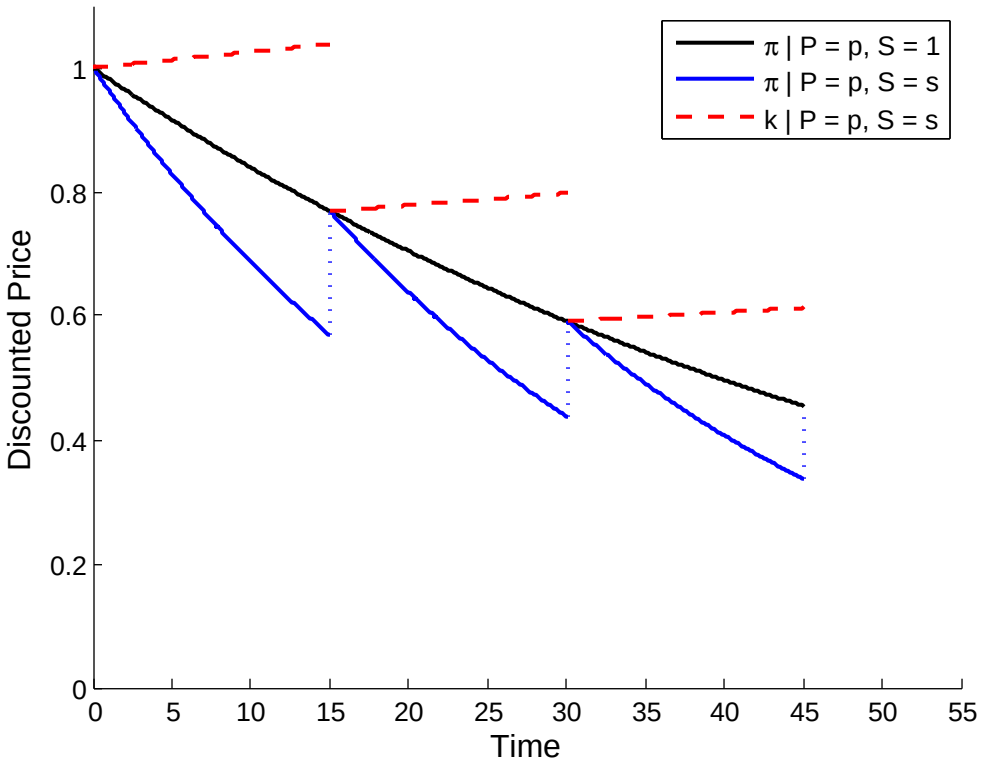


Figure 2.2: Visualization of Deterministic System

2.4 ANALYSIS OF DETERMINISTIC MODEL

Suppose the standard deviations σ_P and σ_S are identically zero in the stochastic differential equations defining P and S. Since the curves in Figure 2.2 are all exponential, we can infer immediately that the renewal periods of constant length T would entail only a difference in the initial discounted price level at the beginning of each period. Also, if we consider $\mathcal{P}_t$ to be the discounted price at time t (shown by the black line in Figure 2.2), then $\mathcal{P}_t$ is monotonically decreasing from $\mathcal{P}_0 = 1$. The nth renewal occurs at $t = nT$ and Π_n is the profit earned during the $(n+1)th$ period, so we can write

$$\Pi_n = \mathcal{P}_{nT}\Pi_{(n-1)}.$$

Under the assumption of equal interrenewal times, the profits of each period constitute a geometric series. A similar argument applies for the part of the renewal cost that depends on P, so the profit over an infinite horizon is

$$\int_0^T e^{(\alpha_P+\alpha_S-\rho)t}dt + \sum_{n=1}^{\infty}\left(\int_0^T \mathcal{P}_{nT}e^{(\alpha_P+\alpha_S-\rho)t}dt - k_0\frac{\mathcal{P}_{nT}}{S_T}\right).$$

Factoring out the geometric series of price points that condition the period profits, we arrive at the following objective:

$$\max_T\left\{\frac{1}{1-e^{(\alpha_P-\rho)T}}\left(\frac{e^{(\alpha_P+\alpha_S-\rho)T}-1}{\alpha_P+\alpha_S-\rho} - k_0e^{(\alpha_P-\alpha_S-\rho)T}\right)\right\}.$$

In order to simplify notation, make the following definitions:

$$a = (\alpha_P + \alpha_S - \rho)$$
$$b = (\alpha_P - \alpha_S - \rho)$$
$$c = (\alpha_P - \rho)$$

$$v_1(T) = \frac{1}{1-e^{cT}}$$
$$v_2(T) = \frac{e^{aT}-1}{a} - k_0e^{bT}$$

The first order necessary condition for maximization of the objective function then becomes

$$v_1(T)v_2'(T) + v_1'(T)v_2(T) = 0,$$

which can easily be solved for T. This is also a sufficient condition for the problem solution, since the objective is quasi-concave on the domain $[0, \infty)$.

In order to see better how this deterministic analysis relates to the stochastic problem, we can eliminate T and find the stopping condition in terms of P and S only. Note first that e^{cT} is the discounted price $\mathcal{P}_T$ at time T. Likewise, e^{aT} gives the product $\mathcal{P}S$ at time T, while e^{bT} gives the ratio $\mathcal{P}/S$ at T. We can therefore define functions $V(\mathcal{P})$ and $\pi(\mathcal{P}, S)$ that are equivalent to $v_1(T)$ and $v_2(T)$:

$$V(\mathcal{P}) \equiv \frac{1}{1-\mathcal{P}} = \frac{1}{1-e^{cT}} = v_1(T)$$

$$\pi(\mathcal{P}, S) \equiv \frac{\mathcal{P}S - 1}{a} - k_0\frac{\mathcal{P}}{S} = \frac{e^{aT}-1}{a} - k_0 e^{bT} = v_2(T).$$

The first order condition in terms of $\mathcal{P}$ and S becomes

$$V(\mathcal{P})d\pi(\mathcal{P}, S) + dV(\mathcal{P})\pi(\mathcal{P}, S) = 0.$$

Solving for $\mathcal{P}$ in terms of S yields

$$\mathcal{P} = \left(1 + \frac{(\alpha_P - \rho)(S(S-1) - k_0 a)}{\alpha_S(S^2 + k_0 a)}\right).$$

Since the system is deterministic, we also have $\mathcal{P} = S^{(\alpha_P - \rho)/\alpha_S}$, so the optimal renewal point is given by

$$\frac{1 - S^{-g}}{g} = \frac{S(S-1) - k_0 a}{S^2 + k_0 a}, \quad \text{where } g = \frac{\rho - \alpha_p}{\alpha_s} < 0. \tag{2.2}$$

Basing renewal activity on the state variable S is more appropriate in the stochastic case, since profit and cost depend on the value of this variable, not on the time since last renewal. Thus even with a policy of renewal when $S = s$, the interrenewal periods may be quite different in length. For example, the interval of 15 years used to generate Figure 2.2 corresponds in the deterministic case to $S = 0.7408$. Taking this value and the other parameters that generated Figure 2.2, but setting $\sigma_P = 0.02$ and $\sigma_S = 0.01$, a sample path may be similar to the one shown in Figure 2.3. The second interrenewal period in Figure 2.3 is clearly about 5 years longer than the other two, even though the all renewals are triggered when the system state reaches $S = 0.7408$. Thus we would like to derive an optimality equation, similar expression to (2.2), for the case of a system with stochastic price and profitability variables.

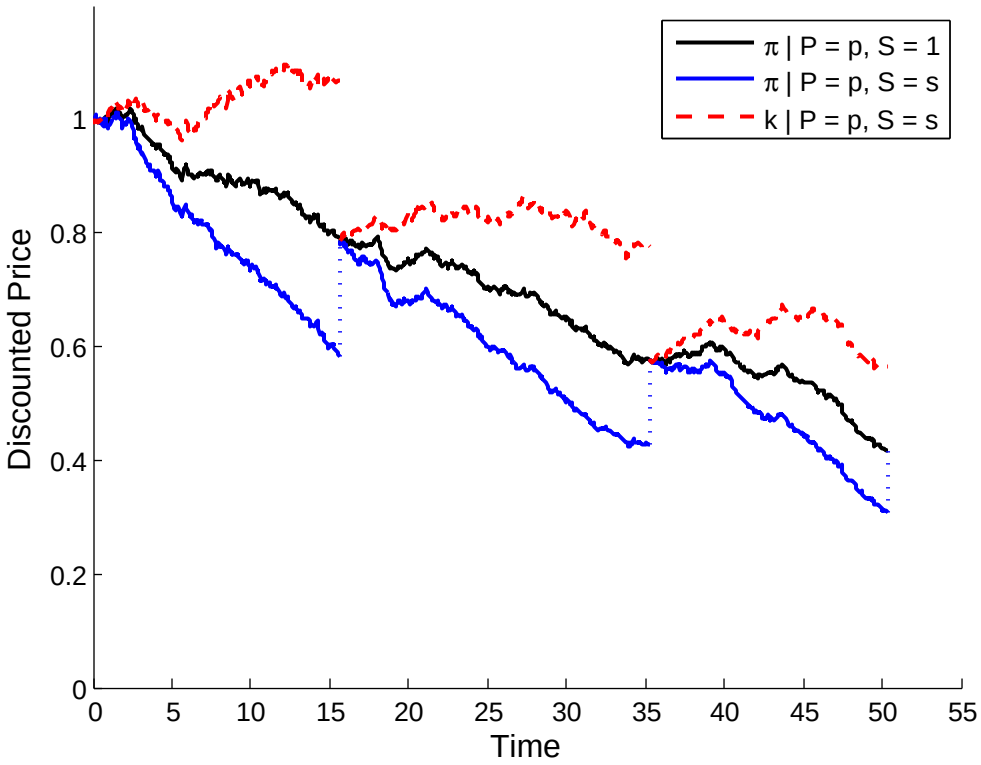


Figure 2.3: Visualization of Stochastic System

2.5 REAL OPTIONS MODEL

Assume that the value of the firm at any point is given by $F(P, S)$. Furthermore,

$$F(P, S) = F_v(P, S) + F_o(P, S), \tag{2.3}$$

where $F_v(P, S)$ is the firm value with exclusion of all future renewals, and $F_o(P, S)$ is the additional value that may be realized through an optimal renewal policy.

Following the standard dynamic programming approach to real options analysis, we derive a differential equation for F by requiring that the return from owning the firm during time dt must equal the cash flow from the firm during dt, plus any capital gain in the firm value:

$$\rho F dt = \pi dt + \mathrm{E}(\mathrm{d}F)$$

Expanding the term $\mathrm{E}(\mathrm{d}F)$ by Itō's Lemma shows

$$\frac{1}{2}\sigma_P^2 P^2 \frac{\partial^2 F}{\partial P^2} + \frac{1}{2}\sigma_S^2 S^2 \frac{\partial^2 F}{\partial S^2} + \alpha_P P \frac{\partial F}{\partial P} + \alpha_S S \frac{\partial F}{\partial S} + PS - \rho F = 0 \tag{2.4}$$

A particular solution to the above PDE is given by the expected value of all future profits, excluding renewals, which we defined as $F_v(P, S)$ above:

$$F_v(P, S) = \frac{PS}{\rho - \alpha_P - \alpha_S} = -\frac{PS}{a}. \tag{2.5}$$

The value of renewal options is therefore given by the homogeneous solution to the PDE,

$$F_o(P, S) = AP^\beta S^\gamma, \tag{2.6}$$

where A, β and γ are coefficients to be determined through consideration of the boundary conditions for the system.

Substituting the homogeneous solution into the homogeneous part of the PDE shows

that the coefficients β, γ must satisfy an elliptical characteristic equation,

$$\sigma_P^2\beta^2 + \sigma_S^2\gamma^2 + \beta(2\alpha_P - \sigma_P^2) + \gamma(2\alpha_S - \sigma_S^2) - 2\rho = 0 \tag{2.7}$$

Additionally, the following "value matching" condition is imposed to ensure that the firm value immediately prior to renewal is equal to the value of the firm after renewal, minus the cost of the renewal itself:

$$AP^\beta S^\gamma - \frac{PS}{a} = AP^\beta - \frac{P}{a} - k_0\frac{P}{S}$$

$$\Rightarrow AP^\beta Sa(S^\gamma - 1) = P(S(S-1) - k_0 a) \tag{2.8}$$

Finally, the partial derivative of (2.8) with respect to P or S gives us two "smooth pasting" conditions that ensure optimality of the renewal point:

$$A\beta P^{\beta-1} Sa(S^\gamma - 1) = S(S-1) - k_0 a \tag{2.9}$$

$$A\gamma P^{\beta-1} SaS^\gamma = S^2 + k_0 a \tag{2.10}$$

Dividing (2.8) by P and comparing the result with (2.9) shows immediately that $\beta = 1$. Thus P drops out of all three equations. Substituting for A from (2.10) into (2.8) yields our optimality equation for this system,

$$\frac{1 - S^{-\gamma}}{\gamma} = \frac{S(S-1) - k_0 a}{S^2 + k_0 a}, \tag{2.11}$$

where γ is determined by setting $\beta = 1$ in (2.7):

$$\gamma = \frac{-(2\alpha_S - \sigma_S^2) \pm \sqrt{(2\alpha_S - \sigma_S^2)^2 + 8\sigma_S^2(\rho - \alpha_P)}}{2\sigma_S^2}. \tag{2.12}$$

Since (2.12) generally yields two values for γ, we need to consider whether both are relevant to the stopping problem. Note first that (2.12) always has one negative solution and one positive solution. This follows from the observation that $\gamma_0 \equiv (\sigma_S^2 - 2\alpha_S)/(\sigma_S^2) > 0$ lies at the center of the ellipse defined by (2.7), so (2.12) can be rewritten

$$\gamma = \gamma_0 \pm \sqrt{\gamma_0^2 + 2\left(\frac{\rho - \alpha_P}{\sigma_S^2}\right)}.$$

The representation above shows that either $\gamma > 2\gamma_0 > 0$ or $\gamma < 0$, because $\rho - \alpha_P > 0$. We will thus use γ_p and γ_n to refer respectively to the positive and negative roots of (2.12), while γ will be used where an assertion is valid for either root.

Considering the numerator of (2.12), we find

$$\lim_{\sigma_S \to 0}(\sigma_S^2 - 2\alpha_S) + \sqrt{(\sigma_S^2 - 2\alpha_S)^2 + 8\sigma_S^2(\rho - \alpha_p)} = 4|\alpha_S|$$
$$\lim_{\sigma_S \to 0}(\sigma_S^2 - 2\alpha_S) - \sqrt{(\sigma_S^2 - 2\alpha_S)^2 + 8\sigma_S^2(\rho - \alpha_p)} = 0,$$

since $\alpha_S < 0$. The first result clearly entails $\lim_{\sigma_S \to 0} \gamma_p = \infty$. In the second case, application of l'Hôpital's rule gives $\lim_{\sigma_S \to 0} \gamma_n = g$. Thus we can ensure that (2.11) corresponds to (2.2) as $\sigma_S \to 0$ if we use only γ_n in our calculations.

Substituting the initial conditions $P_0 = S_0 = 1$ into (2.6) shows that the total present value of the firm's renewal options is given by the constant A. From (2.10) we have

$$A(S, \gamma) = \frac{S^2 + k_0 a}{a\gamma S^{\gamma+1}}. \tag{2.13}$$

Assuming provisionally the existence of a pair (S_n, γ_n) that simultaneously solve (2.11) and (2.12), the present value of the firm's renewal options is $A(S_n, \gamma_n)$.

Since the denominator of $A(S_n, \gamma_n)$ is positive, we see that a renewal policy will not add value if $S_n \leq \sqrt{-k_0 a}$. Thus we only need to investigate the characteristics of solutions to (2.11) and (2.12) on the interval $(\sqrt{-k_0 a}, 1]$. This can be achieved by considering prop-

erties of the left and right side of (2.11) for comprehensive and mutually exclusive cases of $k_0 a$. The results are stated in three propositions below; proofs of these are given in the appendix to this chapter (p. 138 ff.).

Proposition 1. *If $1 + 4k_0 a > 0$, there is a unique solution S_n to (2.11) on $(\sqrt{-k_0 a}, 1]$ for each solution γ_n of (2.12).*

Proposition 2. *The solution $S_n \in (\sqrt{-k_0 a}, 1]$ established by Proposition 1 is strictly decreasing in k_0. Moreover, holding $a < 0$ fixed, we have*

$$1 + 4k_0 a \uparrow 1 \Rightarrow S_n \uparrow 1,$$
$$1 + 4k_0 a \downarrow 0 \Rightarrow S_n \downarrow \frac{1}{2}.$$

Proposition 3. *If $1+4k_0 a < 0$, there is no solution to (2.11) on $(\sqrt{-k_0 a}, 1]$ for any solution γ_n of (2.12).*

These three propositions entail in principle that we can meaningfully and unambiguously discuss the characteristics of the solutions S_n and $A(S_n, \gamma_n)$ with respect to the problem parameters. Nevertheless, the implicit form of equation (2.11) and its dependence in turn on (2.12) effectively exclude an analytic approach to this task. We will thus rely instead of computational studies, which allow us to illustrate the response of S_n or $A(S_n, \gamma_n)$ to parameter changes.

The base case for the computational studies is as follows:

- System drift (α_S): -0.020
- Price delta[4] ($\rho - \alpha_P$): -0.015
- System variance (σ_S): 0.010
- Cost scale (k_0): 1

In the next section we focus in turn on each of the base case parameters, showing the impact of variations in it while holding the other parameters constant.

[4]The optimality equations only feature ρ and α_P in the combination $\rho - \alpha_P$, so we fix $\rho = 0.06$ and vary α_P for our numerical studies.

2.6 NUMERICAL EXPERIMENTS

We conduct three distinct sets of experiments. First, we consider how the optimal renewal policy changes as we vary parameter values from the base case. Second, we look at corresponding changes in the total value of renewal options. In order to put the option values in perspective, we also show for each case the expected system value *without* renewal activity. Finally, in order to evaluate the performance of the infinite horizon policy in finite horizon contexts, we provide a set of simulation studies.

RENEWAL POLICY

Our experiments suggest initially that the presence of stochasticity in the model does not have substantial impact on S_n, the threshold level for capital renewal. Instead, the renewal threshold appears more sensitive to k_0, the scaling factor for cost of renewal, and to the price delta, $\rho - \alpha_P$.

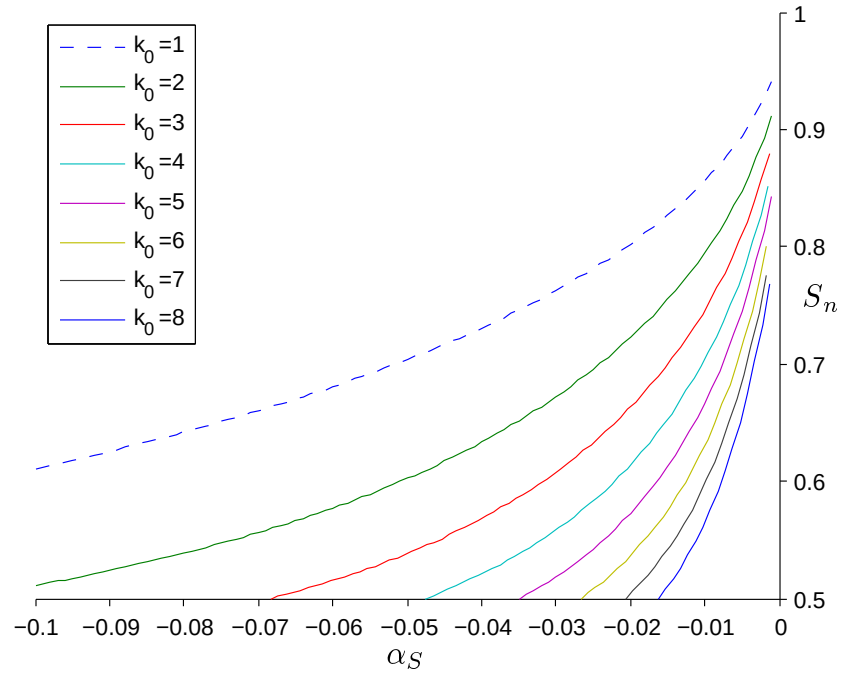
Figure 2.4a illustrates the variation of S_n with k_o and α_S, the drift coefficient for profitability. The range of threshold levels is clearly much greater than the range shown in Figure 2.4b, where σ_S and α_S are varied. Indeed, the differences in S_n in Figure 2.4b are essentially indistinguishable once we move outside the plausible range for α_S (*e.g.*, if $\alpha_S < -0.10$).

The practical significance of these small differences in the renewal threshold appears much greater, however, when we consider the corresponding expected time to renewal. This is because the expected time to renewal depends in turn on the system parameters σ_S and α_S. Following Harrison (1985), we calculate the expected time to reach S_n from $S_0 = 1$ as

$$\mathrm{E}[T] = \frac{2}{\sigma_S^2 - 2\alpha_S} \ln\left(\frac{1}{S_n}\right). \tag{2.14}$$

Figures 2.5a and 2.5b portray the same variation as Figures 2.4a and 2.4b, but in terms of expected time to renewal $\mathrm{E}[T]$, instead of S_n. In Figure 2.5a we see that when the scale of

(a) S_n versus α_S and k_0 (fixed $\rho - \alpha_P = 0.015, \sigma_S = 0.010$)



(b) S_n versus α_S and σ_S (fixed $\rho - \alpha_P = 0.015, k_0 = 1$)

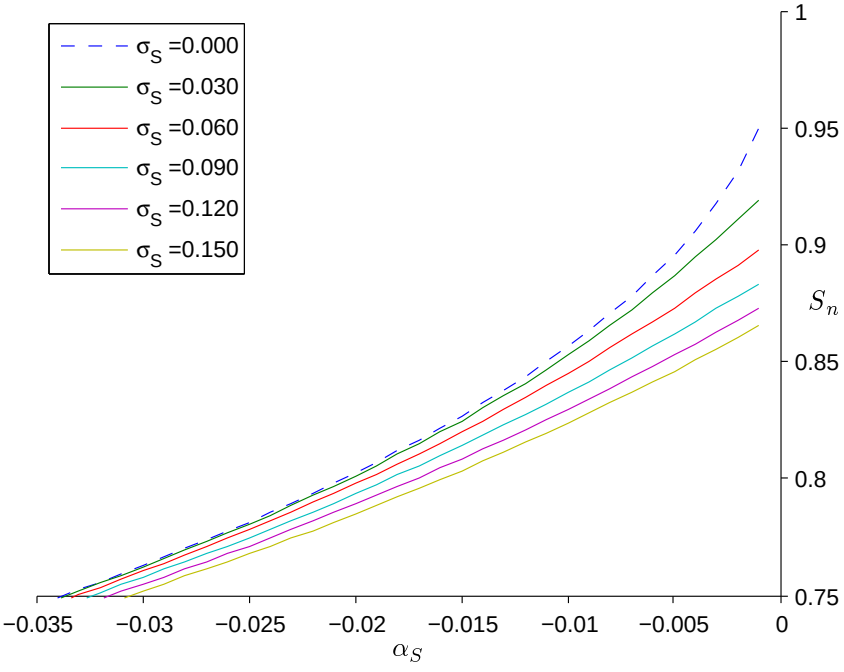
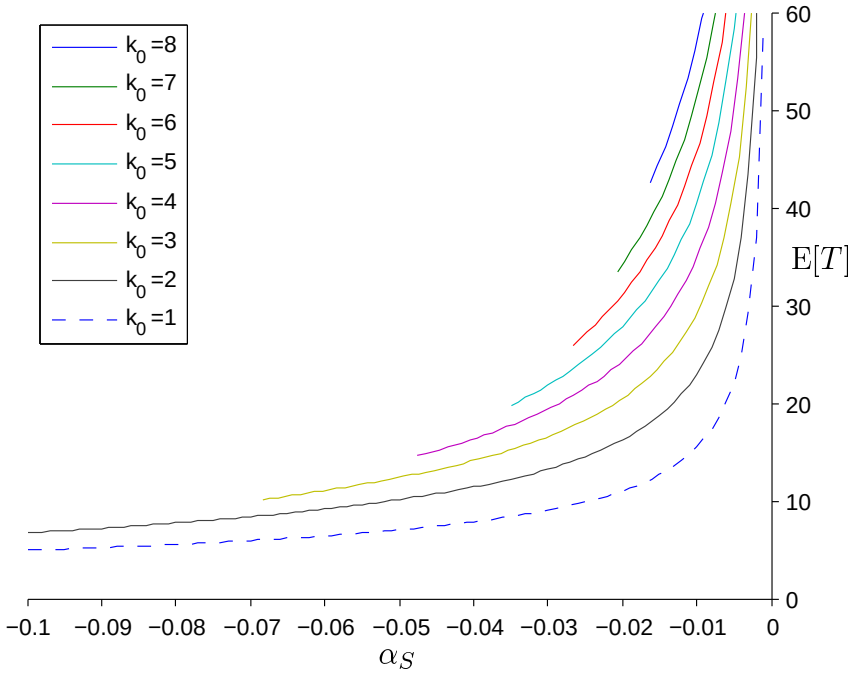


Figure 2.4: Variation of S_n with α_S, σ_S, and k_0

(a) $E[T]$ versus α_S and k_0 (fixed $\rho - \alpha_P = 0.015, \sigma_S = 0.010$)



(b) $E[T]$ versus α_S and σ_S (fixed $\rho - \alpha_P = 0.015, k_0 = 1$)

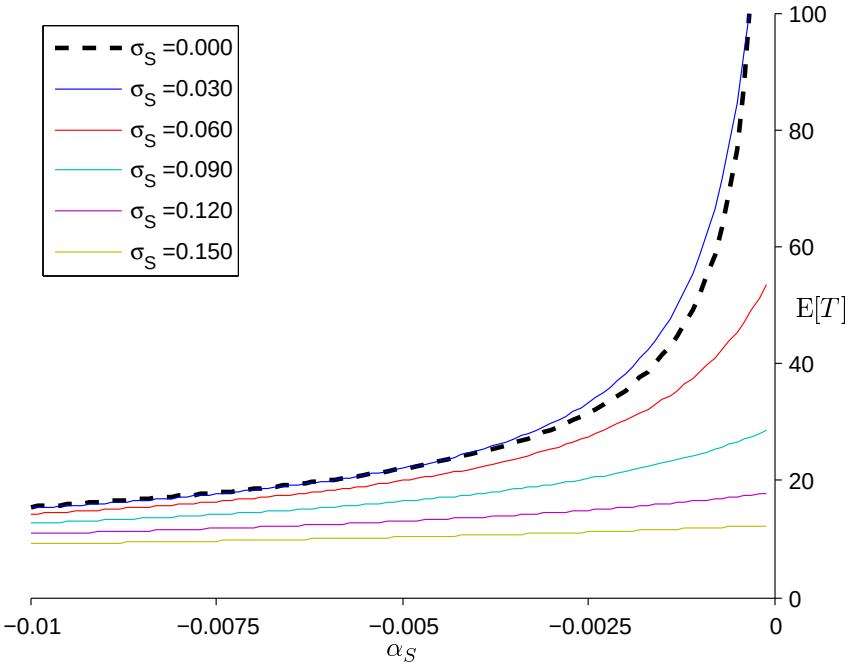
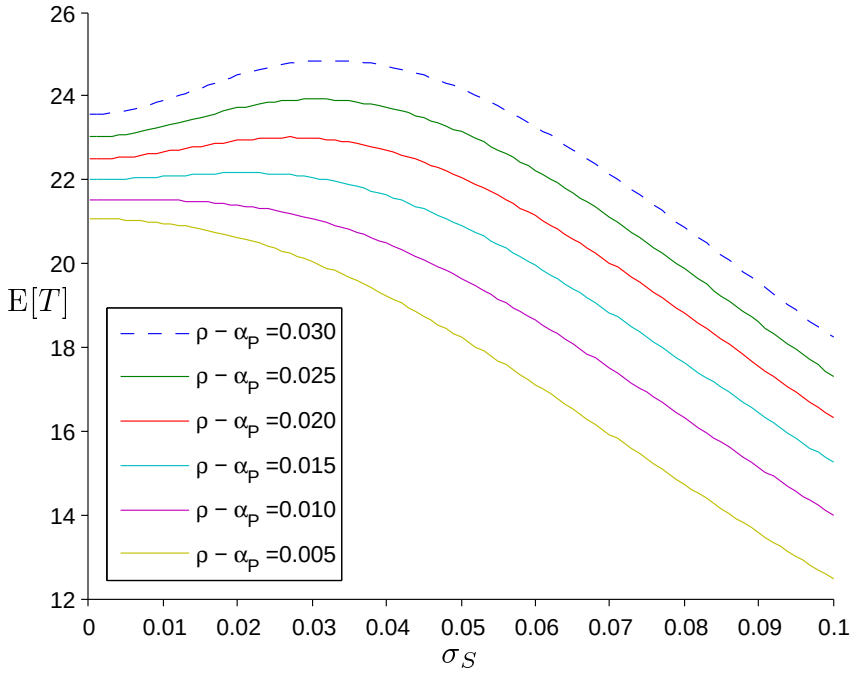


Figure 2.5: Variation of $E[T]$ with α_S, σ_S, and k_0

(a) $\mathrm{E}[T]$ versus α_P and σ_S (fixed $\alpha_S = -0.0050, k_0 = 1$)



(b) $\mathrm{E}[T]$ versus α_P and σ_S (fixed $\alpha_S = -0.0005, k_0 = 1$)

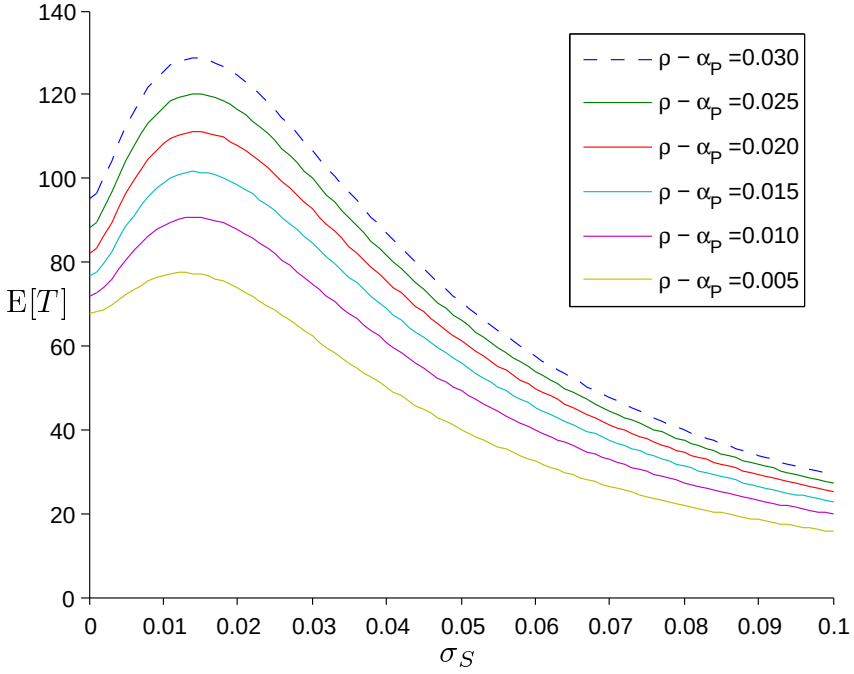


Figure 2.6: Variation of $\mathrm{E}[T]$ with α_P and σ_S

renewal costs is relatively low, *i.e.*, when k_0 is small, increasing α_S, the expected annual rate of profitability decline, has little effect on the expected time to renewal. Comparing Figure 2.5b with Figure 2.4b, we see that although increased variance of profitability implies a lower renewal threshold, the corresponding expected time until renewal is generally *shorter* than in cases where the variance is smaller. Nevertheless, there is some ambiguity in the cases with smallest variance.

Since Figures 2.4a - 2.5b indicate that greatest variation in S_n or $\mathrm{E}[T]$ appears for less negative values of α_S, we fix $\alpha_S = -0.0050$ in Figure 2.6a, in order to illustrate the effect of the difference $\rho - \alpha_P$ as well as σ_S. While we smaller values of $\rho - \alpha_P$ indicate the same diminishing effect of variance on $\mathrm{E}[T]$ that we saw in Figure 2.5b, we see that for larger values of $\rho - \alpha_P$, increasing variance may initially entail an increase in $\mathrm{E}[T]$.

The implication of Figure 2.6a is taken further in Figure 2.6b. Here we further reduce α_S to -0.0005, approaching a limit where profitability is more similar to Brownian motion without drift. Since the deterministic optimal renewal policy given in (2.2) is the limit of the stochastic policy (2.12) when $\sigma_S \to 0$, we can take the limit $\alpha_S \to 0$ in (2.2) and infer that $S_n \to 1/2$ when $\sigma_S = 0$. In this case, 2.14 gives $\mathrm{E}[T] \to \infty$. When $\sigma_S \to \infty$, however, we find $\mathrm{E}[T] \to 0$. The implication here is that for values of σ_S in an intermediate range, such as $(0.05, 0.15)$, the value of $\mathrm{E}[T]$ will be practically significant. This entails that even with negligible drift, a finite optimal expected renewal cycle may exist, due of the stochastic nature of the firm's profitability.

RENEWAL OPTION VALUE

We turn now to consider the response of F_o, the expected value of the firm's renewal options, to changes in the system parameters. Here, the initial expected NPV *without* any renewals serves a point of comparison for renewal option value. From (2.5) and 2.6, this NPV is $F_v(1,1) = 28.6$ in the base case of our computational studies. In each of the figures that illustrate the value of renewal options (Figures 2.7 - 2.9), we show the corresponding

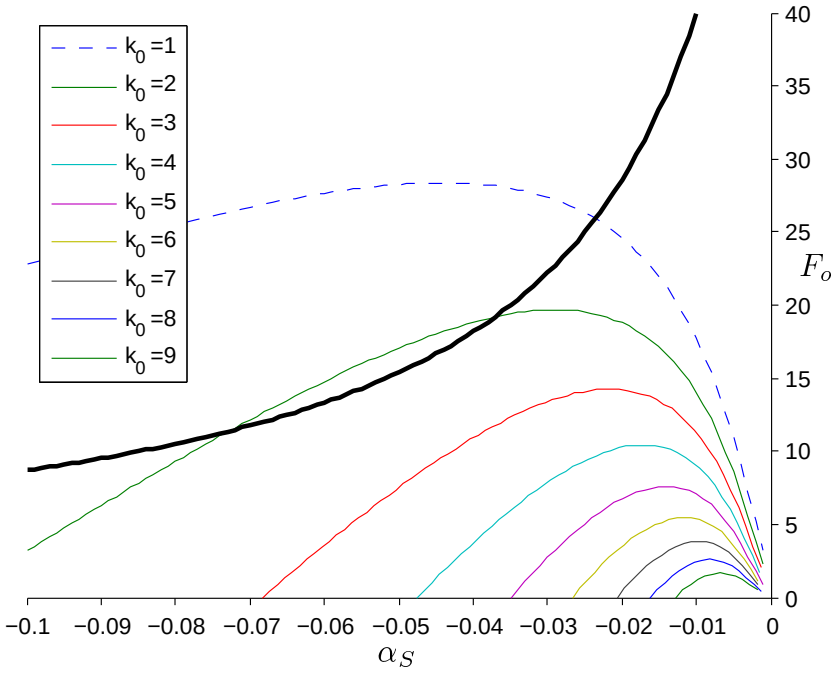


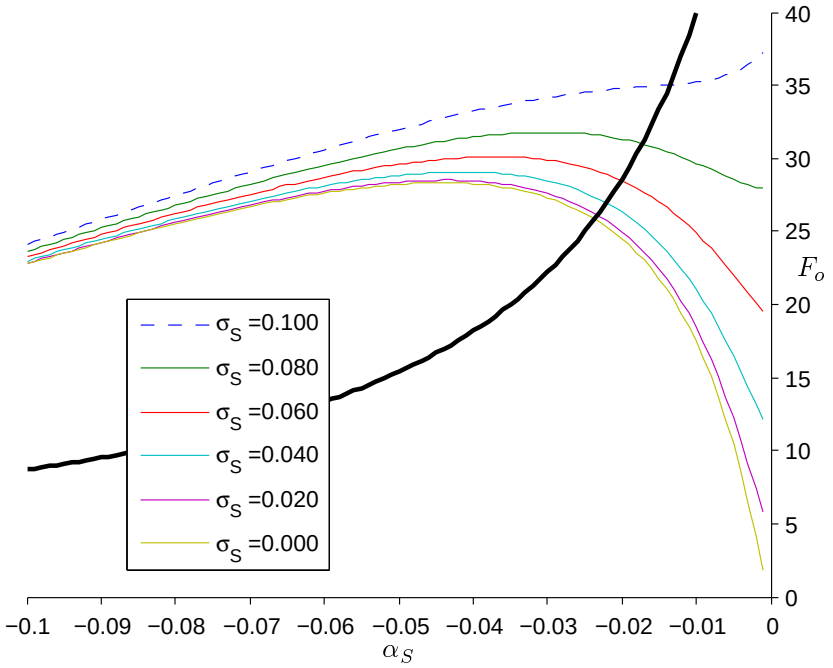
Figure 2.7: Option value F_o versus α_S and k_0 (fixed $\sigma_S = 0.010, \rho - \alpha_P = 0.015$)

values of F_v by a thicker black curve.

Holding σ_S and $\rho - \alpha_P$ fixed at their base values, Figure 2.7 suggests that F_o is decreasing in cost scale k_0 and convex in system drift α_S. Moreover, while F_o is substantial relative to F_v for intermediate levels of system drift and lower cost scales, extreme values of α_S bring a rapid decrease in the relative magnitude of F_o. When α_S is small, renewals add little or no value at any cost scale, and when α_S is large, renewals are only profitable for the lowest cost scales. Of course, these results are all contingent on the base value of system variance, $\sigma_S = 0.010$. Responses at other levels of variance may differ. Since cost scale has a great effect on the value of F_o but does not change the value of F_v, however, we will consider separately a high cost and a low cost scenario when examining the variation of F_o with other parameters.

Considering different levels of system variance σ_S, Figure 2.8a suggests that in the low cost scenario, renewal activity may still be profitable when α_S is small, provided σ_S is larger than the base case value. This divergent set of outcomes corresponds to curves on

(a) fixed $k_0 = 1, \rho - \alpha_P = 0.015$



(b) fixed $k_0 = 6, \rho - \alpha_P = 0.015$

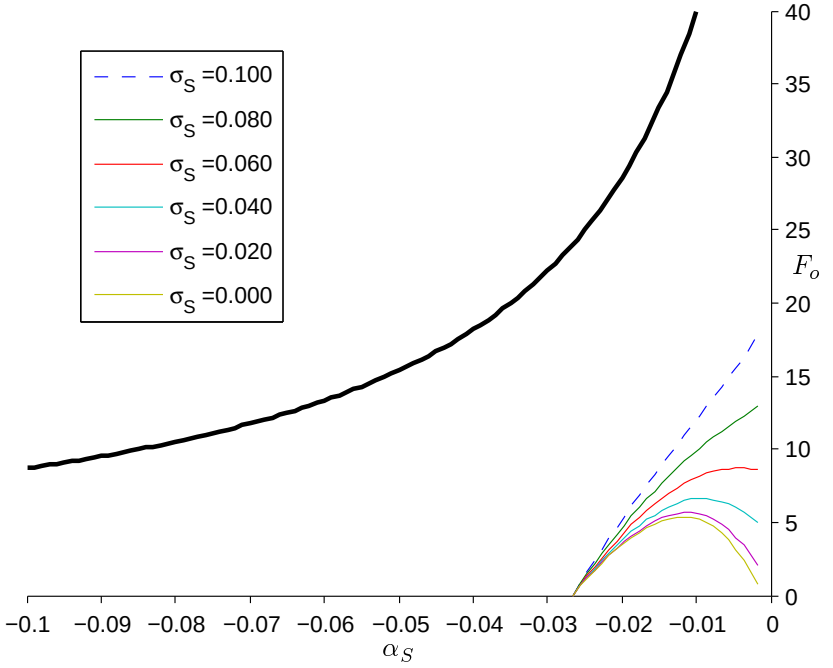


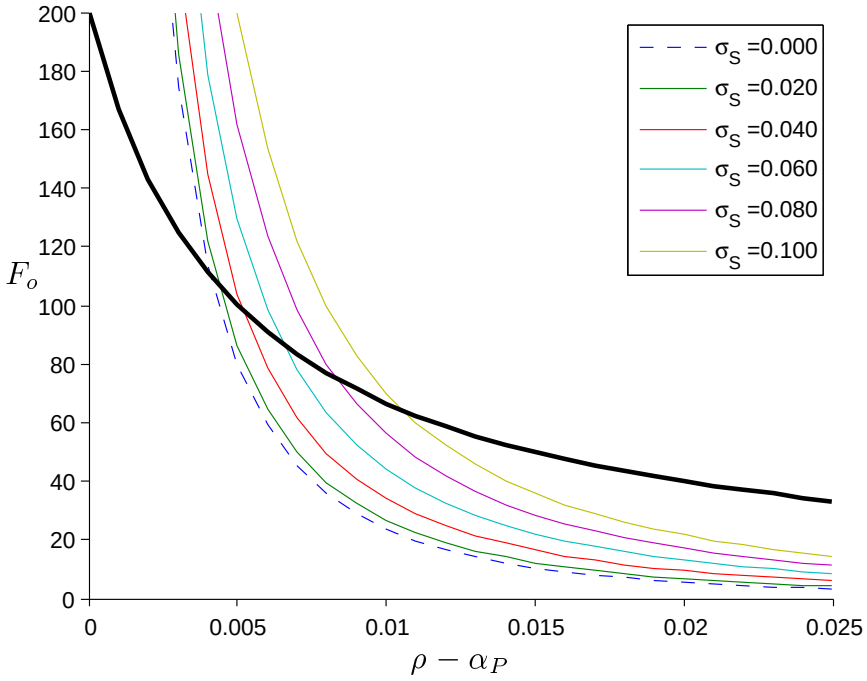
Figure 2.8: Option value F_o versus α_S and σ_S

the right side of the figure. Here, if σ_S is small, renewal options have little value, as in the right side of Figure 2.7. Nonetheless, for large values of σ_S the value of renewal options can still be large. We also see in Figure 2.8a that the impact of system variance on F_o is qualitatively similar to its impact on S_n or E$[T]$ (Figures 2.4b and 2.5b). Variance is only important for smaller values of system drift α_S. If α_S is large, increases or decreases in variance entail little change in the magnitude of F_o. The drift effectively overwhelms the short-term variations in profitability, so that the renewal options tend towards their deterministic value.

The higher cost scenario for changes in system variance, Figure 2.8b, is essentially a miniature version of the low cost scenario. Although the option value is less overall and its rate of change with system drift appears larger, the qualitative insights are the same within the range of drifts for which the total value of renewal options is positive.

The effect of the price delta $\rho - \alpha_P$ on F_o is shown in Figures 2.9a and 2.9b, which correspond respectively to the lower cost scale and the higher cost scale. A small value of system drift is used in both ($\alpha_S = -0.005$), in order to show a greater spread across the different levels of system variance σ_S. As far as the the system variance is concerned, we see again that the case of the higher cost scale appears qualitatively the same as the lower cost scale. Higher costs force all curves closer together and bring an overall decrease in the value of renewal options. Nevertheless, in both figures, the magnitude of F_o is substantial relative to F_v at lower and intermediate levels of $\rho - \alpha_P$. When the rate of price growth is small in comparison to the discount rate, however, as shown on the right of the figures, renewals are less profitable . Effectively, the value realized through renewal activity is more heavily discounted here, so the potential for renewal activity in the more distant future is inconsequential to the present. This makes sense in light of Figure 2.6b and the discussion above: expected renewal periods are longest in the case of heavier discounting and low system drift.

(a) fixed $k_0 = 1, \alpha_S = -0.005$



(b) fixed $k_0 = 6, \alpha_S = -0.005$

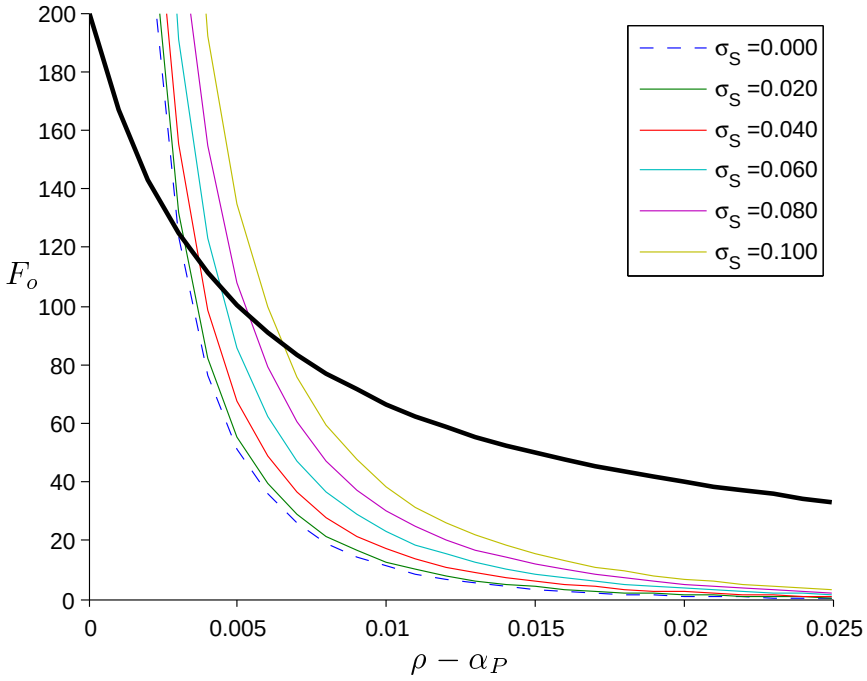


Figure 2.9: Option value F_o versus $\rho - \alpha_P$ and σ_S

FINITE HORIZON PERFORMANCE

The preceding computational studies were conducted entirely in the infinite horizon context. Financial plans generally have a finite horizon, however, even where long-term capital budgeting and infrastructure management are concerned. Thus the performance of the infinite horizon policy S_n in finite horizon contexts is a question of inevitable practical importance. In order to address this, we perform a series of simulation experiments. In each experiment, we simulate the the operating profit and renewal costs from a range of renewal policies, including the infinite horizon policy S_n, across a period of time T.

In light of our findings about the expected value of renewal options, the values we use for system drift α_S and price delta $\rho - \alpha_P$ in our experiments should be such that the outcomes are not trivial, *i.e.,* we wish to exclude values where the infinite horizon analysis already tells us that renewals are unlikely to be profitable. We therefore fix $\alpha_S = -0.005$ and $\rho - \alpha_P = 0.005$ for all experiments. A value for price variance is needed for the simulation, even though it does not affect the calculation of the optimal infinite horizon renewal policy, so we set $\sigma_P = 0.020$ throughout.

Given the fixed parameters, we examine the finite horizon performance of renewal policies at two different levels of the system variance σ_S and at two different levels of the cost scale k_0, entailing four distinct systems. For each of the four systems, we consider four horizons: $T = 50$ years, $T = 100$ years, $T = 150$ years, and $T = 200$ years. We thus have a total of sixteen simulations.

In each case, finite horizon performance is calculated as the total discounted net profit derived from the system to time T, plus a salvage value, V_T. In order to put policies on an equal footing for comparison, and to allow comparison of each with the expected system NPV from the infinite horizon problem, we use an estimate of the salvage value that conforms to our calculation of the infinite horizon NPV,

$$V_T = e^{-\rho T} P_T S_T F(1,1). \tag{2.15}$$

From 2.3, the infinite horizon NPV is $F(1,1) = F_v(1,1) + F_o(1,1)$. Given the fixed values of α_S and $\rho - \alpha_P$, we find immediately that $F_v(1,1) = 100$ for all experiments, whereas $F_o(1,1)$ follows from 2.6 and 2.13.

Table 2.1 details the key values for the simulations. Each simulation includes 4,000 independent replications of each policy, using common random numbers.

Simulation	σ_S	k_0	$F(1,1)$
A	0.005	1	180.4
B	0.005	6	151.3
C	0.050	1	215.9
D	0.050	6	176.1

Table 2.1: Simulation Experiments

The results of the experiments appear in Figures 2.10 - 2.13. Each figure shows simulated policy thresholds S on the horizontal axis and policy performance on the vertical axis. The performance of the infinite horizon policy is distinguished by a solid vertical line, while best performance in each simulation is distinguished by a dashed vertical line. For example, in Figure 2.10a, the infinite horizon policy is approximately $S_n = 0.90$, while $S^* = 0.95$ is approximately the best-performing policy. Dashed lines on either side of the performance curve show a 95% confidence interval for the results.

The first figure in each set shows that the performance of the infinite horizon solution is almost certainly not optimal for a short horizon, $T = 50$, but we also see that there is likely to negligible difference between it and the finite horizon optimum when $T = 150$ or greater. Indeed, the performance of the infinite horizon policy and the performance of the optimal finite horizon policy tend to be statistically indistinguishable for $T = 100$. Simulation B (Figure 2.11) presents the only contrast to this pattern, since the performance of the infinite horizon policy there remains clearly suboptimal for all but the longest horizon. This agrees,

however, with our insights derived earlier, since system B combines low infrastructure variance with a high cost scale and the value of renewal options is small. (Table 2.1 also shows that system B has the lowest total value of renewal options.) In this case, the infinite horizon renewal policy corresponds to expected renewal every 63 years, so we would not expect to renew at all in a capital budget for $T = 50$ years. Simulation suggests, however, that one renewal in a 50 year period would be optimal, at about the 37-year mark. With a system and context like this, a finite horizon analysis is needed to determine the optimal renewal policy. Otherwise, the infinite horizon solution performs well.

In all results from the simulation experiments it is apparent that the performance curve drops more sharply to the right of the optimal finite horizon policy than to the left of it, and that the infinite horizon policy is always situation to the left. Thus we can conclude that in the finite horizon context, *extending* the expected time between renewals is always preferable to shortening it. This observation perhaps sheds some light on the otherwise unfortunate phenomenon of public and private infrastructure being invariably *in need* of renewal, rather than too frequently renewed.

2.7 CONCLUSIONS

Capital renewal decisions have received little attention from academic researchers, despite their close relation to the classic problems of system replacement or maintenance. Our treatment here is inspired by the real options approach to investment and replacement investment timing, but our contribution to this literature lies in explicitly recognizing the impact of exongenous prices on capital renewal requirements; preexisting research has viewed replacement investments as pure renewal activity, without recognition of background prices, which is quite implausible in the context of repeated renewals.

Several managerial insights emerge from our study. Most importantly, we affirm that timely capital renewal may greatly enhance the return infrastructure investment. Neverthe-

less, we have identified some specific instances where renewal may be less profitable: firms facing high costs for renewal or large cost of capital should renew far less frequently and will extract less benefit from the renewals they do undertake. We also show that the impact of variance in infrastructure depreciation is limited in cases where either the expected rate of depreciation is great, or where the firm's cost of capital is high, relative to its expected rate of price growth. Conversely, when infrastructure tends to depreciate slowly and price growth is comparable to the cost of capital, variance in depreciation may be important for the optimal renewal policy. A small amount of variance may advance the expected renewal schedule, whereas higher levels of variance will tend to delay renewals.

These insights follow directly from our analysis of the infinite horizon renewal problem. Our study of the performance of the infinite solution in finite horizon contexts amplifies the analytical findings. Where the infinite horizon analysis shows a large expected value for renewal activity, the performance of the infinite horizon solution is also close to optimal within all but the shortest finite horizon problems. Where the infinite horizon analysis shows little value for renewal activity, the infinite horizon solution performs poorly on finite horizon problems. These cases would be better addresses by a finite horizon analysis. Finally, we note that from a purely financial perspective, the simulation studies suggest that if optimal renewal activity cannot be achieved, either because of lack of funding or because it may be difficult to establish precisely a current measure of infrastructure profitability, delaying renewal activity is better than advancing it.

One shortcoming of the current study is that geometric Brownian motion processes can entail implausible increases in the infrastructure state variable on some sample paths of the model. Future work could remedy this through further simulation work, with infrastructure described by a non-increasing jump process. Future work could also consider the influence of constraints or competition on renewal decisions: in the former case, when financial resources are limited, prioritization of renewal funding will be needed; in the latter case, an oligopolistic setting will entail endogeneity of price levels.

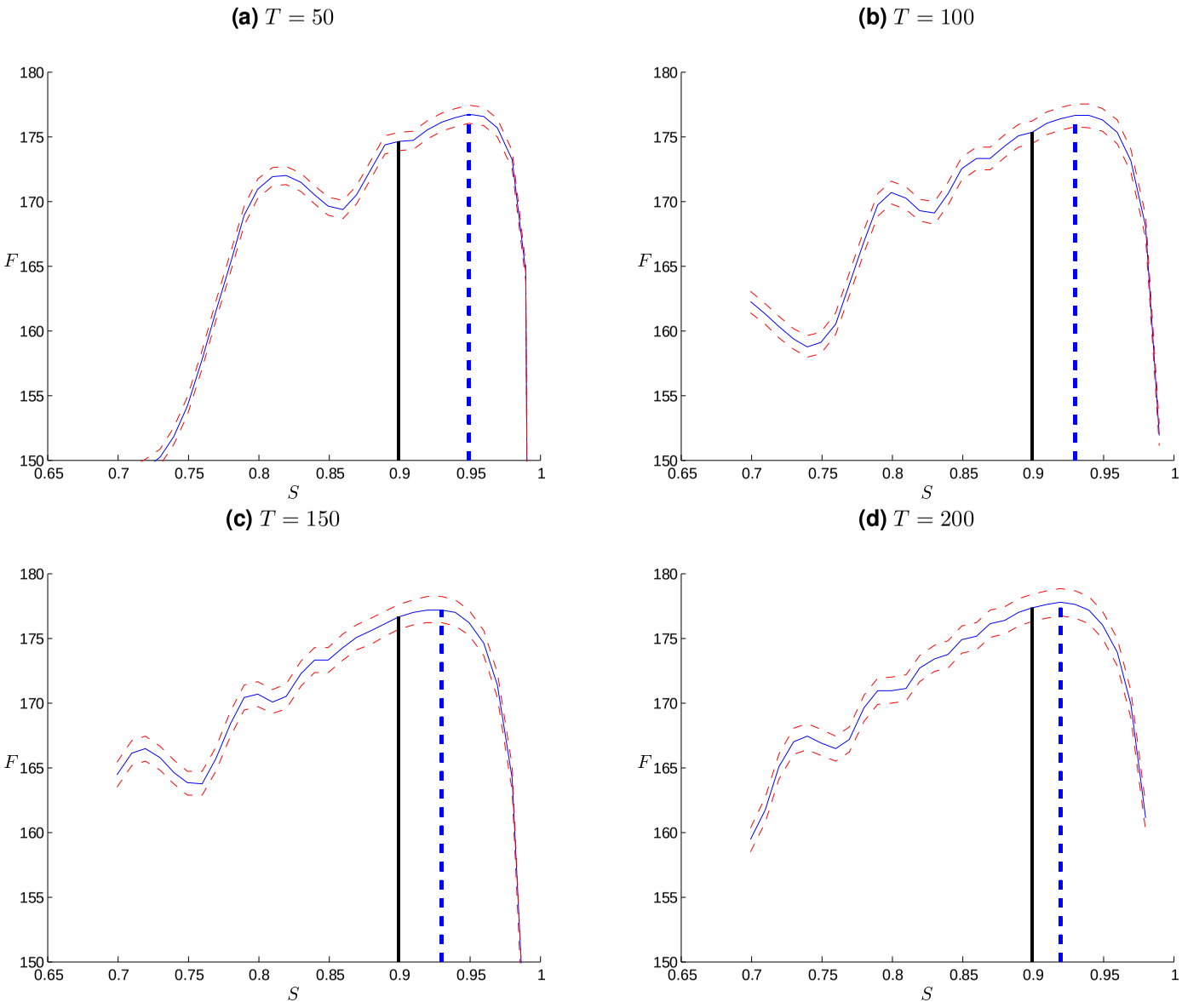


Figure 2.10: Finite Horizon, System A: $\sigma_S = 0.005$, $k_0 = 1$, $F(1,1) = 180.4$

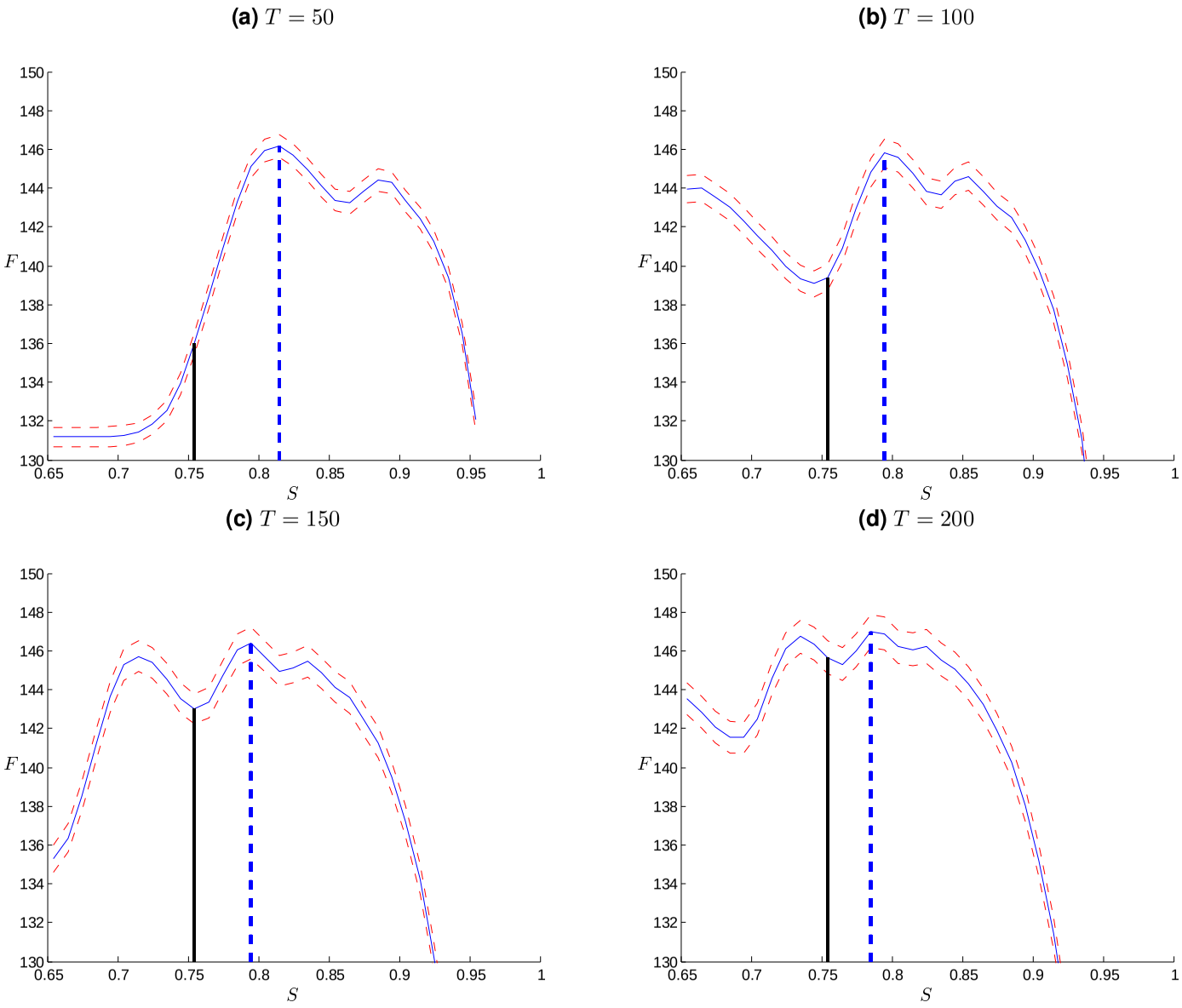


Figure 2.11: Finite Horizon, System B: $\sigma_S = 0.005$, $k_0 = 6$, $F(1,1) = 151.3$

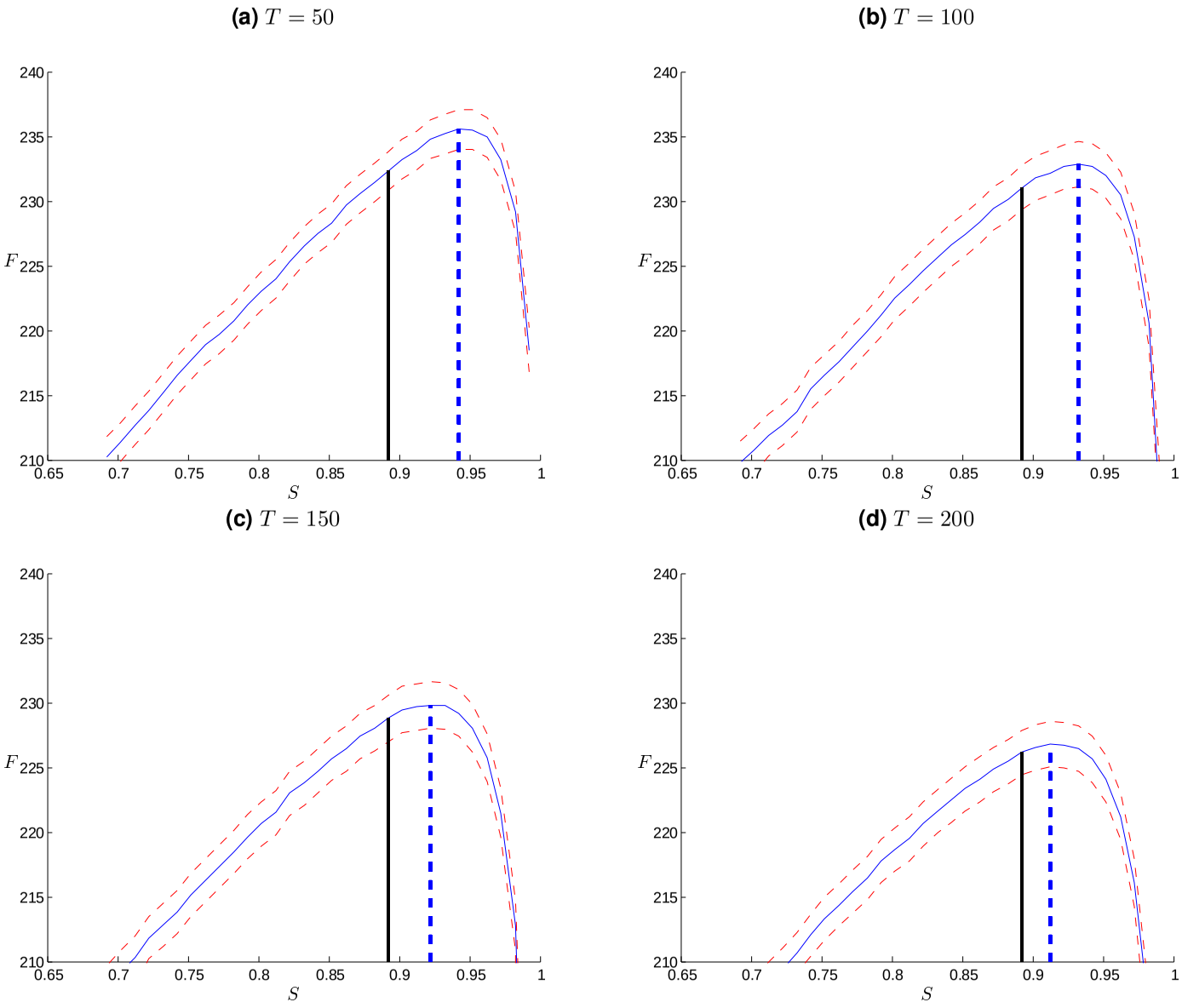


Figure 2.12: Finite Horizon, System C: $\sigma_S = 0.050$, $k_0 = 1$, $F(1,1) = 215.9$

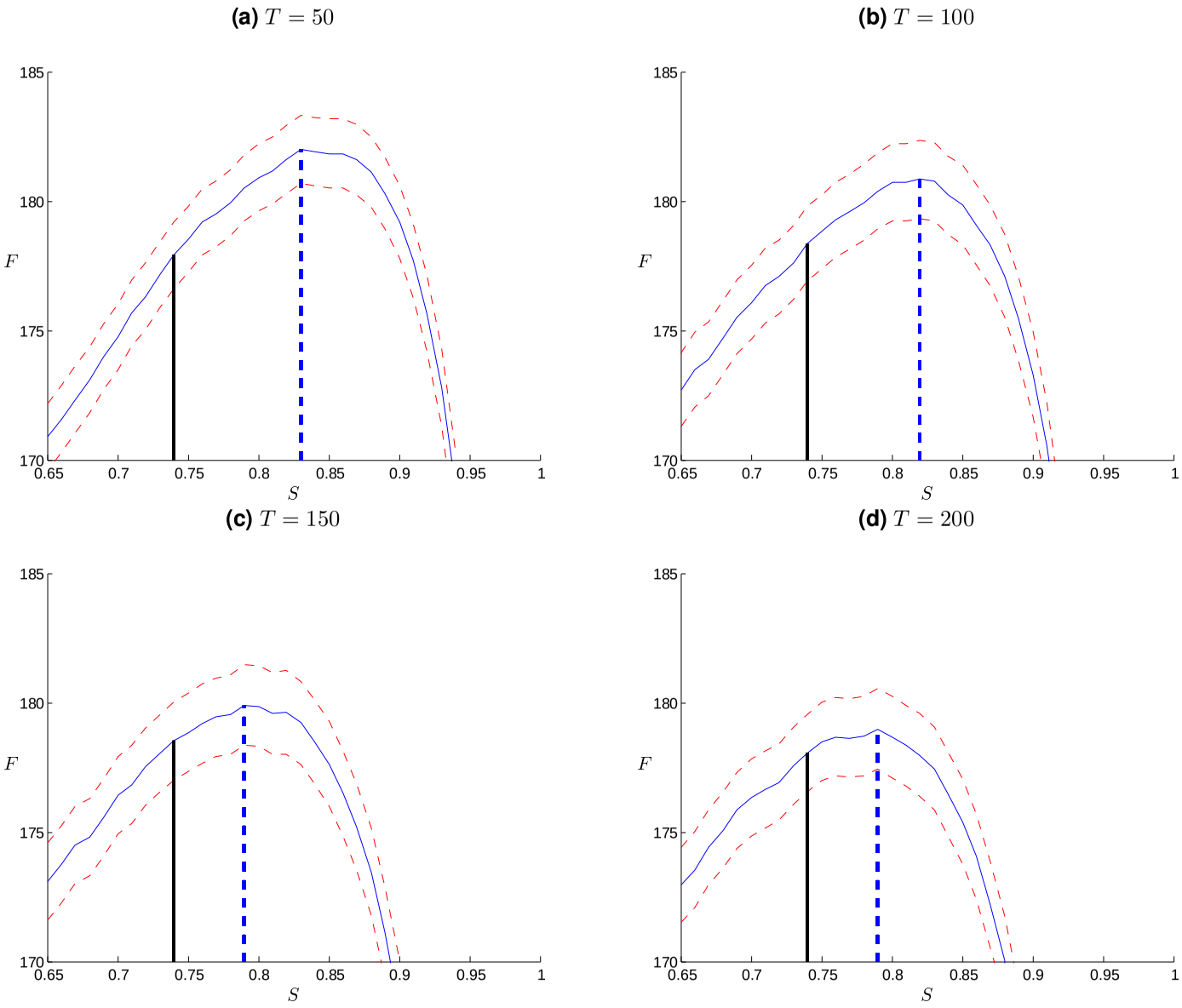


Figure 2.13: Finite Horizon, System D: $\sigma_S = 0.050$, $k_0 = 6$, $F(1,1) = 176.1$

CHAPTER 3

DYNAMIC LEAD TIME PROMISING

3.1 MOTIVATION

Electronic commerce allows the terms of a contract for goods or services to be arranged within a matter of minutes or even seconds. A consumer may evaluate options and read reviews at length before deciding what and where to purchase, but the transaction itself generally follows a quick, simple formula: select the item from the relevant website, choose any desired options, verify the cost and delivery terms at "checkout," then authorize payment and receive a printable receipt. The attractiveness of this "online shopping" model is is apparently great for consumers and firms alike, despite the potential complications of doing business remotely - *e.g.*, the possibility of delayed or incorrect product deliveries. The convenience and scope of the online market has lead to widespread participation.

Of course, the quickly and easily navigable steps of an online purchase are enabled by systems that are themselves quite sophisticated. From the firm's perspective, demands

arrive in rapid succession; in order to specify products, costs, and delivery terms reliably for each, information and operations management must proceed at the same fast pace. Thus, organizations that can track inventory, optimize production, and schedule shipments in a "real time" manner may gain a distinct competitive advantage.

Advanced ATP (available-to-promise) systems are designed to address these business goals. Whereas conventional ATP is defined as the "uncommitted portion of a company's inventory and planned production, maintained in the master schedule to support customer order promising" (Wallace and Dougherty, 1987), *advanced* ATP is a mechanism that "dynamically allocates and reallocates resources to promise and fulfill customer orders" (Chen et al., 2002). In this chapter we analyze lead time promising as a Markov Decision Problem and use approximate dynamic programming methods to lay the foundation for a novel advanced ATP system.

The ability to offer different lead times to customers is an important means by which a firm can achieve efficient allocation of its resources in the face of demand or supply uncertainty. Intuitively, if the range of possible lead time offers for new demands is greater, the firm can more flexibly handle disruptions and challenging conditions. The benefits of this flexibility may be tempered, however, by customers' natural propensity to reject lead time offers that are undesirably long. The optimal level of lead time flexibility is thus ultimately an essential question for the design of an advanced ATP system.

In order to evaluate the marginal value of lead time flexibility, however, we must first be able to determine an optimal lead time promising policy for each given system configuration. This prerequisite task is the focus of most of the current study. Our setting of the lead time promising problem is enhanced by the assumption that customers constitute multiple classes, each class having a distinct set of lead time expectations. Since two key considerations for classification of a production system are (a) whether it serves single or multiple classes of customer and (b) whether lead times are determined endogenously or exongenously, our work is related to much existing literature. Nevertheless, the formula-

tion of our multiple class, endogenous lead time problem distinguishes it not only from other studies of this type of system, but also from studies of multiple class, exogenous lead time systems and from studies of single class, endogenous lead time systems. In general, research in these areas considers production systems where jobs are scheduled individually on a small number of servers and the time required for each job is important (Cheng and Gupta, 1989). Contrastingly, our model envisions a system that has capacity to process hundreds or potentially thousands of jobs in a single unit of production time. Thus the challenge is not individual ordering of jobs for service, but construction of an appropriate production "batch" from a rapidly arriving stream of heterogeneous orders.

The organization of this chapter is as follows. In section 3.2 we consider briefly the historical development of the lead time promising problem, from early literature through to the latest research on advanced ATP systems. Section 3.3 describes in detail the formulation of the problem we study and outlines the additional assumptions that will be used to provide solutions. Section 3.4 presents two small versions of the problem and shows their explicit solution using a Markov chain model. This allows us to generate some exact results for later use as benchmarks, and also to illustrate the curse of dimensionality and the curse of modeling. Section 3.5 introduces Reinforcement Learning as a solution to these difficulties, then discusses alternative formulations of the problem within the context of Q-learning, a common variant of Reinforcement Learning. In particular, we describe a formulation that adjusts the perspective from which lead time promises are made and allows us to take advantage of structure inherent in the system. Section 3.6 presents our results. We evaluate the performance of each of the different Q-learning formulation, and show how these performances vary with key problem parameters: the level of lead time flexibility, the frequency of supply chain disruptions, the capacity of the system, and the cost of overtime. We also compare the performance of our best Q-learning results with the results from the Markov chain model. Finally, section 3.7 summarizes the work and gives thoughts for next steps in the analysis of the lead time promising problem.

3.2 LITERATURE REVIEW

Early queueing models of production systems with multiple classes generally take customers' lead time requirements to be exogenous constraints. Cox and Smith (1961) consider systems where distributions of service time are determined by the "priority" class of each customer. They analyze performance objectives such as mean cost or mean time in queue, both for preemptive and non-preemptive priorities. Studies by Harrison (1975) and Tcha and Pliska (1977) extend the basic models, adding the possibility that individual rewards depend on a customer's class and the length of time in queue. The key result for this type of system is the optimality of a weighted shortest expected processing time rule, also known as a $c\mu$ rule. Mieghem (1995) generalizes the $c\mu$ rule, and recent work has provided even more sophisticated policies for exogenous lead time queues: the generalized longest queue (GLQ) rule and generalized largest delay (GLD) rule. See, for example, Bertsimas et al. (1998), Stolyar and Ramanan (2001), and Van Mieghem (2003).

Conway (1965) appears to be the first to stand apart and observe that system performance may be improved if lead times are not taken as fixed requirements, but instead are to some extent determined endogenously by the firm. Despite computational limitations, several subsequent studies of the time use simulation methods to evaluate different rules for making lead time quotes: examples include Eilon and Hodgson (1967), Eilon and Chowdhury (1976), and Weeks (1979). Analytical work in the multiple class setting is extended to cover endogenous lead times by, for example, Seidmann and Smith (1981), Bertrand (1983), Bookbinder and Noor (1985), and Wein (1991). As for single class problems with endogenous lead times, Spearman and Zhang (1999) undertake minimization of average leadtime for an $M/M/1$ queue, where service constraints are imposed for the percentage of jobs delivered on time or for average tardiness. Hopp and Sturgis (2001) extend these results to $M/G/1$ queues and consider also a constraint of average tardiness as a percentage of leadtime.

Most of works from the late 1990s and early 2000s explicitly anticipate the impact of

new technology on order promising and fulfillment systems, and thus *advanced* ATP has emerged as a very general concept, subsuming previously distinct research questions and ERP tools within a comprehensive task of supply chain optimization. Indeed, advanced ATP is not a single approach but "a variety of methods and tools to enhance the responsiveness of order promising and the reliability of order fulfillment" (Pibernik, 2005). Some of the first work in this direction comes from Taylor and Plenert (1999), who use a heuristic technique called finite capacity promising to track traditional ATP quantities and generate feasible lead times. Hegedus and Hopp (2001) develop a lead time policy for a model that include a procurement stage as well as a manufacturing stage, thus allowing for uncertain vendor lead times. Subsequent notable contributions come from Chen et al. (2002), featuring integer programming methods; Jeong et al. (2002), with a scheduling heuristic that combines linear programming and minimum setup time, and Moses et al. (2004), using a search algorithm on a memory resident database to determine the earliest availability of resources to meet a newly arrived demand.

3.3 MODEL FORMULATION

Suppose a firm manufactures and sells a single range of products, but that it distinguishes several different "priority" classes among its customers. Customers in higher classes expect (and will pay for) shorter lead times than customers in lower classes. We assume that a range of possible lead times is fixed for each class; when receiving a customer's order, the firm makes an explicit lead time offer from the relevant class range. A lead time offer that is as short as possible within its class will be accepted, but longer lead times incur an increasing likelihood that the customer will decide to withdraw the order. Since the production schedule for any given day receives lower class demand assignments before higher class assignments, the lead times offered to the former must leave "sufficient space" for the latter; if necessary, by offering non-preferential lead times to certain lower class

demands.

In accordance with this general scenario, our model of the basic parameters and rules of the firm's production system is as follows.

- The production schedule is tracked across a rolling horizon. Day 0 always denotes the current day (where production occurs), while day 1 denotes tomorrow, day 2 the day after tomorrow, *etc*. The load of day y at any instant is the number of demands already assigned for production on the y^{th} day after today. Time, measured in days, is given by $t \in \mathbb{R}^+$, and thus we denote the load of day y at time t by $l_y^t \in \mathbb{N}$ (throughout this chapter, we take $\mathbb{N}$ to include zero).

- There are n classes of demand, denoted by integers $1, 2, \ldots, n$. A demand in a given class i can receive one of N_i possible lead time offers, quoted in days. A specific lead time offer option for class i is denoted by $\phi_{ij} \in \mathbb{Z}^+$, $1 \leq j \leq N_i$. For example, a lead time offer of $\phi_{ij} = x$ means that a demand is offered production on the x^{th} day after today. We assume $\phi_{ij} > 0$, meaning that no new demands can be offered production on the current day, and $\phi_{ij} < \phi_{ik}$ for $j < k$.

- Demands arrive according to a Poisson process with rate λ per day and random (independent) splitting between classes. A demand is from class i with probability δ_i, where $\sum_i \delta_i = 1$.

- For $j > 1$, the probability that the customer in class i will reject an offer of ϕ_{ij} is given by $h_{ij} \in \mathbb{R}^+$.

- The nominal production capacity of the system is b units. Loads greater than b incur production overtime costs: if $l_y^t \geq b$, an assignment to day y will incur a penalty of $u(l_y^t - b + 1)$, where $u \in \mathbb{R}^+$ represents unit overtime cost.

- The maximum capacity for any production day is $c > b$ units. Demands will be lost if they cannot be assigned to a day with load less than c.

- Revenue $r(i)$ is generated when a lead time offer is accepted by a demand from class i. We normalize the revenues by setting $r(1) = 1.0$ and assume that $r(i) > r(j)$ for $i < j$, so that higher priority demands are more valuable than lower priority demands.

The basic model described above does not recognize any resource uncertainty, which could result, for example, from inventory supply chain disruptions. If resources are not available as expected at the start of a production day, some demands will have to be rescheduled for production on later days. This additional dimension can be added through a stochastic perturbation of the nominal capacity:

- At the start of each production day, a capacity loss occurs with probability τ, independent of any previous occurrences of capacity loss. If there is a capacity loss, the magnitude of the loss is $\theta\%$ of the nominal capacity parameter b, where $\theta \sim DU[0, 100]$ is independent of τ.

- If capacity loss entails that a demand of class i needs to be rescheduled, a cost of $x(1 + n - i)$ is incurred, where $(\min_{i,j} \phi_{ij}) \leq x \leq (\max_{i,j} \phi_{ij})$ is the number of days the demand is delayed.

- If capacity loss is so severe that a demand of class i cannot be rescheduled, a cost of $(1 + \max_{i,j} \phi_{ij})(1 + n - i)$ is incurred.

A lead time promising policy subject to resource uncertainty will presumably be different than a policy for deterministic resources. By distinguishing the elements of resource uncertainty from the basic system configuration, we allow for separate analysis and comparison of the two cases.

We turn now to the decision mechanism in our model. Let Σ be the state and parameter space of an instance of our production model, so that $\sigma(t) \in \Sigma$ is a vector specifying load information for all days within the production horizon at time t and all fixed parameters for the given system configuration. A stationary policy π determines a lead time offer for a

new demand from class i,

$$\pi(i, \sigma(t)) \in \{\phi_{i1}, \ldots, \phi_{iN_i}\}. \tag{3.1}$$

Thus we can define the average system reward resulting from π,

$$\xi(\pi) \equiv \lim_{n\to\infty} \frac{1}{n} \sum_{k=0}^{n} \hat{r}_k^{\pi}, \tag{3.2}$$

where $\hat{r}_k^{\pi} \geq 0$ is the net reward from the k^{th} demand when policy π is in force. Supposing that the k^{th} demand is of class i and arrives at time t, the net reward is defined by

$$\hat{r}_k^{\pi} = 1\{l^t_{\pi(\sigma(t),i)} < c\} \left(r(i) - u(l^t_{\pi(\sigma(t),i)} - b + 1)^+\right). \tag{3.3}$$

Our objective is to find $\max_{\pi} \xi(\pi)$.

Although $\pi(.)$ depends in general on all state and parameter information contained in $\sigma(.)$, a computational approach to optimizing average reward will require some approximation of the policy function, in order to keep the dimension of the problem within tractable limits. For example, in a system with a scheduling horizon of just 9 days and capacity of 100 units per day, the subspace of Σ representing the loads alone would contain 10^{18} points. Possible approaches to approximation include selection of a subset of key elements from $\sigma(.)$, aggregation of sets of state vectors into representative single vectors, or more sophisticated methods, such as functional approximation or neural network training. While the latter often allow greater parameter reduction and/or less information loss, they also entail increasing obscurity in the definition of the function π. We shall restrict the approximation methods used here to selection of subsets from the elements of $\sigma(.)$.

3.4 MARKOV CHAIN ANALYSIS

The dimension of the state space of the problem depends on the capacity of the system and the number of days in the production horizon, while the dimension of the action space depends on the number of customer classes and associated lead times. If these factors are all sufficiently small in magnitude and we fix a policy for lead time promising, the production system constitutes a Markov chain with just a few hundred or thousand states. We may then calculate the average reward from the chain and perform a direct search of the policy space, in order to reveal the optimal policy and corresponding average reward. We develop in this section such a Markov chain model, as a means to give further insight into the problem and to derive a closed-form characterization of the average reward as a function of the policy. The analytic results will provide a benchmark for the performance of reinforcement learning methods that will be used to tackle larger system instances.

3.4.1 BASE CASE: ONE DEMAND CLASS

Suppose the production system faces only a single demand class, so each demand corresponds to potential revenue of $r = 1.0$. Restrict the production horizon to two days, so that the lead time options are just $\phi_1 = 1$ and $\phi_2 = 2$. We will describe this system as a Markov chain $\{X_i \mid i \in \mathbb{N}\}$, where $X_i \in \mathbb{N}$ denotes the number of demands already assigned to day 1 at the start of the i^{th} day of production.

The transitions of the chain $\{X_i\}$ represent periodic observations of the underlying continuous-time Markov process (l_1^t, l_2^t). As noted earlier, $l_y^t \in \mathbb{N}$ is the number of demands assigned to day y after $t \in \mathbb{R}^+$ days have elapsed. At the end of each day, the demands assigned to day 1 go into production, the demands assigned to day 2 become demands assigned to day 1, and a new day 2 enters the scheduling horizon. Thus the number of demands assigned to day 2 at the start of a period is always zero. With a slight abuse of notation, using the integer index i for the corresponding magnitude in $\mathbb{R}$, we can suppress

$l_2^i = 0$ and take the state of the Markov chain to be $X_i = l_1^i$, the number of demands assigned to day 1 at the start of the i^{th} day of production.

In the current context, the number of daily demands is a Poisson random variable L with mean λ. Let $P_\lambda(i)$ be the the p.m.f. of L. The evolution of (l_1^t, l_2^t) between observations X_i and X_{i+1} is conditioned by the lead time policy of the system. Lead time offers are made according to a threshold policy, as follows:

- The lead time policy is defined as an integer $0 \leq \pi \leq c$. A lead time of 1 day will be offered at epoch t if and only if $l_1^t < \pi$.
- If $l_1^t \geq \pi$ but $l_2^t < c$, then a lead time of 2 days will be offered[1]. If $l_1^t \geq \pi$ and $l_2^t = c$, any arriving demand will be lost.
- If a lead time of 2 days is offered, the offer may be rejected with probability h or accepted with probability $a = 1 - h$. If the offer is rejected, the reward from the corresponding demand is zero; otherwise, the reward is $r - u(l_2^t - b + 1)^+$.

The value of π thus determines the average reward $\xi(\pi)$ that the system will generate. Let π^* be the choice of π that maximizes average reward.

In order to give an expression for $\xi(\pi)$, we must determine the corresponding transition probability matrix Q_π for $\{X_i\}$ and the expected immediate reward in each state of the chain. Let the entries of Q_π be q_{jk}^π, with $0 \leq j, k \leq c$. Let the expected immediate reward in state j be $\mathrm{E}[s_j^\pi]$.

For better economy of notation in the following exposition, let $\widehat{jk}$ denote the event that the chain moves from state j to state k, and define $M_{xy} = (x - y)^+$.

When $k = 0$, event $\widehat{j0}$ corresponds to exactly one of the two following scenarios: (a) there are at most $M_{\pi j}$ demands and all receive a lead time of 1 day; (b) there are more than $M_{\pi j}$ demands, but all lead time offers of 2 days are rejected. When $0 < k < c$, the event $\widehat{jk}$

[1]The case of more than π demands on day 1 can only arise if the day begins with more than π demands, *i.e.*, if more than π demands were assigned to day 2 in the preceding period.

corresponds to the following scenario: there are at least $M_{\pi j} + k$ demands, and exactly k lead times offers of 2 days are accepted (after the first $M_{\pi j}$ lead time offers of 1 day). Thus we have

$$q^{\pi}_{jk} = I(k=0) \sum_{i=0}^{M_{\pi j}-1} P_\lambda(i) + \sum_{i=k}^{\infty} P_\lambda(M_{\pi j}+i) \binom{i}{k} a^k h^{i-k}, \quad 0 \le k < c, \tag{3.4}$$

where $I(k=0) = 1$ if $k = 0$ and $I(k=0) = 0$ otherwise.

When $k = c$, we need to recognize that demands may be lost after the load on day 2 reaches c. This can lead to complicated expressions for q^{π}_{jc}. The first two cases are shown below.

- If the event is $\widehat{jc}$ and we have $L = M_{\pi j} + c + 1$, then there are two possible subcases: (a) the first c of the last $c+1$ demands are accepted and the last demand is lost (w.p. 1); (b) one of first c demands rejects its offer and the last demand accepts. These subcases entail

$$q^{\pi}_{jc}|\{L = M_{\pi j} + c + 1\} = \left(\binom{c}{0} h^0 a^c 1^1 + \binom{c}{1} h^1 a^{c-1} a^1 \right) P_\lambda(M_{\pi j} + c + 1).$$

- If the event is $\widehat{jc}$ and we have $L = M_{\pi j} + c + 2$, then there are three subcases: (a) the first c of the last $c+2$ demands accept their offer and the last two are lost (w.p. 1); (b) one of the first c demands is rejects its offer, one of the last two demands accepts, and the other demand from the last two either rejects or is lost; (c) two of the first c demands reject and both of the last two accept. These subcases entail

$$q^{\pi}_{jc}|\{L = M_{\pi j} + c + 2\} = \\ \left(\binom{c}{0} h^0 a^c 1^2 + \binom{c}{1} h^1 a^{c-1} (h^1 a^1 + a^1 1^1) + \binom{c}{2} h^2 a^{c-2} a^2 \right) P_\lambda(M_{\pi j} + c + 2).$$

The expressions for $q^{\pi}_{jc}|\{L = x\}$ proceed analogously when $x > M_{\pi j} + c + 2$. Fortu-

nately, instead of computing q^{π}_{jc} explicitly, we can set

$$q^{\pi}_{jc} = 1 - \sum_{i=0}^{c-1} q^{\pi}_{ji},$$

where the values of q^{π}_{ji} are given by (3.4).

Turning now to the expected immediate rewards, let $\mathrm{E}[s^{\pi}_j]$ be the expected immediate reward in state j. With respect to the possible next state k, let F_{jk} denote the set of all samples[2] that could lead to event $\widehat{jk}$, and let $s(f)$ be the immediate reward resulting from $f \in F_{jk}$. Thus we have

$$\mathrm{E}[s^{\pi}_j] = \sum_k \sum_{f \in F_{jk}} s(f)\, P(f).$$

When $k > 0$, the set F_{jk} has exactly one element: we must have $L \geq M_{\pi j} + k$, so L fills the remaining $M_{\pi j}$ spaces on day 1, then fills k spaces on day 2. Thus $P(f) = q_{jk}$ when $k > 0$, and each of the corresponding values of $s(f)$ can be determined as the inner product of two appropriately defined vectors of dimension $\pi + c$. Specifically,

$$s(f) = \langle v, w(j,k) \rangle\,, \tag{3.5}$$

where v is a vector of possible rewards,

$$\begin{aligned}
v_i &= 1, \quad && 1 \leq i \leq \min(\pi, b), \\
v_i &= 1 - ui, \quad && (\min(\pi, b) + 1) \leq i \leq \pi, \\
v_i &= 1, \quad && \pi + 1 \leq i \leq \pi + b, \\
v_i &= 1 - u(i - \pi - b), \quad && \pi + b + 1 \leq i \leq c,
\end{aligned}$$

[2]A sample that leads to event $\widehat{jk}$ is a given number of demand arrivals and associated acceptances, rejections, or losses.

and $w(j,k)$ is a vector of binary selections,

$$w_i(j,k) = 0, \quad 1 \le i \le j, \tag{3.6}$$

$$w_i(j,k) = 1, \quad j+1 \le i \le \pi, \tag{3.7}$$

$$w_i(j,k) = 1, \quad \pi+1 \le i \le \pi+k, \tag{3.8}$$

$$w_i(j,k) = 0, \quad \pi+k+1 \le i \le c. \tag{3.9}$$

When $k = 0$, the set F_{j0} will usually have more than one element, since the number of demands entering the system is a random variable with range $[0, \pi - j]$. Using

$$S(j,0) \equiv \sum_{f \in F_{j0}} s(f)\, P(f) \tag{3.10}$$

and the triangular number notation

$$T(n) \equiv \sum_{i=1}^{n} i = \frac{n(n+1)}{2},$$

we find:

- If $\pi \le b,$ for $0 \le j < \pi$,

$$S(j,0) = \sum_{i=0}^{\pi-j} iP_\lambda(i) + (\pi - j) \sum_{i=1}^{\infty} P_\lambda(\pi - j + i)h^i,$$

- If $\pi > b,$ for $0 \le j < b$,

$$\begin{aligned} S(j,0) = &\sum_{i=0}^{b-j} iP_\lambda(i) + \sum_{i=b-j+1}^{\pi-j} (i - uT(i+j-b))\, P_\lambda(i) \\ &+ ((\pi - j) - uT(\pi - b)) \sum_{i=1}^{\infty} P_\lambda(\pi - j + i)h^i, \end{aligned}$$

• If $\pi > b,$ for $b \leq j < \pi$,

$$S(j,0) = \sum_{i=0}^{\pi-j} \left(i - uT(i+j-b) + uT(j-b)\right) P_\lambda(i) + \left((\pi-j) - uT(\pi-b) + uT(j-b)\right) \sum_{i=1}^{\infty} P_\lambda(\pi-j+i) h^i,$$

• If $j \geq \pi, S(j,0) = 0.$

The stationary probability vector υ^π for the system satisfies the equation $\upsilon^\pi = \upsilon^\pi Q_\pi$ and the average reward per step is

$$\xi(\pi) = \sum_{j=0}^{c} \upsilon_j^\pi \mathrm{E}[s_j^\pi]. \tag{3.11}$$

The average reward for policy π, as a fraction of possible reward, is thus $\dfrac{\xi(\pi)}{\lambda}$.

3.4.2 EXTENSION: TWO DEMAND CLASSES

Suppose the system faces two demand classes. The potential reward for class 2 demands is $r(2) = \eta$, where $0 < \eta < 1.0$. This case can be analyzed as extension of the single class case, provided we assume that is no overlap of lead times across classes, meaning

- each specific lead time option is available for at most one class,
- if a lead time option for class i is greater (less) than a lead time option for class j, then all lead time options for class i must be greater (less) than all lead time options for class j.

With this assumption, we take the lead time options for current system to be as follows:

- $\phi_{11} = 1$
- $\phi_{12} = 2$
- $\phi_{21} = 3$
- $\phi_{22} = 4$

The scheduling horizon for the system thus extends to four days after the current production day, and the underlying Markov process for the system is $(l_1^t, l_2^t, l_3^t, l_4^t)$, where again $l_y^t \in \mathbb{N}$ is the number of demands assigned to day y after $t \in \mathbb{R}^+$ days have elapsed. The system capacity c, workload penalty threshold b, and unit overtime cost u apply to any of these days in the same manner as before. The lead time policy for the current case is an extension of the threshold concept employed for single class system, as follows.

- The lead time policy π is a vector (π_1, π_2), where $0 \leq \pi_1, \pi_2 \leq c$.
- If an arriving class 1 demand finds $l_1^t < \pi_1$, it receives a lead time offer of 1 day; if $l_1^t \geq \pi_1$ but $l_2^t < c$, the lead time offer is 2 days; if $l_1^t \geq \pi_1$ and $l_2^t = c$, the arriving demand is lost.
- If an arriving class 2 demand finds $l_3^t < \pi_2$, it receives a lead time offer of 3 days; if $l_3^t \geq \pi_2$ but $l_4^t < c$, the lead time offer is 4 days; if $l_3^t \geq \pi_2$ and $l_4^t = c$, the arriving demand is lost.
- For either demand class, if the longer lead time is offered, the offer may be rejected with probability h or accepted with probability $a = 1 - h$. If the offer is rejected, the reward is zero. If the offer is accepted, the reward is the full amount for the class type, minus any overtime penalty.
- The number of daily class 1 demands is a Poisson random variable L with mean λ. $P_\lambda(i)$ is the p.m.f. of L.
- The number of daily class 2 demands is a Poisson random variable K with mean κ. $P_\kappa(i)$ is the p.m.f. of K.

Since $l_4^t = 0$ whenever t is an integer, we proceed in a similar manner as before and describe the system by the Markov chain $\{(X_1, X_2, X_3)_i \mid i \in \mathbb{N}\}$, where $(X_1, X_2, X_3)_i \equiv (l_1^i, l_2^i, l_3^i)$. The system states j and k are thus vectors, $j = (j_1, j_2, j_3)$ and $k = (k_1, k_2, k_3)$.

In order specify consistently the transition probability matrix Q_π, we order system states as follows:

$$\text{If } j_1 > k_1, \text{ then } j > k;$$

$$\text{If } j_1 = k_1 \text{ and } j_2 > k_2, \text{ then } j > k;$$

$$\text{If } j_1 = k_1,\ j_2 = k_2, \text{ and } j_3 > k_3, \text{ then } j > k.$$

The analysis of the single-class system serves as a basis for analysis of the current case. Indeed, after minor modifications to the preceding analysis, we can view each transition in the double-class system as simultaneous, independent transitions with each of two single class subsystems. Specifically, consider the transition from $j = (j_1, j_2, j_3)$ to $k = (k_1, k_2, k_3)$. As far as class 1 demands are concerned, this is the transition $\widehat{j_1 k_1}$ in a single class subsystem, with the additional specification of j_2 demands already assigned to day 2 of the subsystem at the start of the relevant period. For class 2 demands, we have the transition $\widehat{j_3 k_3}$ in a single class subsystem, with the additional specification of k_2 demands assigned to day 1 of the subsystem at the end of the relevant period. Let let $\widehat{j_1 k_1 \,|\, j_2}$ be the class 1 transition just described, and let $q_1(j_1 k_1 \,|\, j_2, \pi_1, \lambda)$ be its probability. Similarly, let $\widehat{j_3 k_3, k_2}$ be the class 2 transition, and let $q_2(j_3 k_3, k_2 \,|\, \pi_2, \kappa)$ be its probability. Then we have for the double class system,

$$q^\pi_{jk} = q_1(j_1 k_1 \,|\, j_2, \pi_1, \lambda) q_2(j_3 k_3, k_2 \,|\, \pi_2, \kappa).$$

We now calculate the transition probabilities $q_1(j_1 k_1 \,|\, j_2, \pi_1, \lambda)$ for the class 1 subsystem. For economy of notation, we employ scalar indices without subscripts, as follows: $j = j_1$, $k = k_1$, $m = j_2$, and $\pi = \pi_1$, with transition probabilities q^π_{jk} for a single class system with demand mean λ.

$$q_1(j, k \,|\, 0, \pi, \lambda) = q^\pi_{jk},$$

$$q_1(j,k\,|\,c,\pi,\lambda) = 1 \text{ for } k = c,$$

$$q_1(j,k\,|\,m,\pi,\lambda) = q^{\pi}_{j(k-m)} \text{ for } 0 < m \le k \le c-1,$$

$$q_1(j,c\,|\,m,\pi,\lambda) = 1 - \sum_{i \ne c} q_1(j,i\,|\,m,\pi,\lambda) = 1 - \sum_{i=m}^{c-1} q^{\pi}_{j(i-m)} \text{ for } 0 < m \le c-1.$$

For the class 2 transition probabilities, $q_2(j_3k_3,k_2\,|\,\pi_2,\kappa)$, we again employ scalar indices without subscripts, as follows: $j = j_3$, $k = k_3$, $m = k_2$, and $\pi = \pi_2$, with transition probabilities q^{π}_{jk} for a single class system with demand mean κ.

If $j \ge \pi,$ then $m = j$ for all $k,$ so

$$q_2(jk,j\,|\,\pi,\kappa) = q^{\pi}_{jk},$$

$$q_2(jk,m\,|\,\pi,\kappa) = 0 \text{ for } j \ne m.$$

If $j < \pi$ and $k > 0,$ then $m = \pi,$ so

$$q_2(jk,\pi\,|\,\pi,\kappa) = q^{\pi}_{jk},$$

$$q_2(jk,m\,|\,\pi,\kappa) = 0 \text{ for } m \ne \pi.$$

If $j < \pi$ and $k = 0,$ then

$$q_2(j0,m\,|\,\pi,\kappa) = 0 \text{ for } m > \pi,$$

$$q_2(j0,m\,|\,\pi,\kappa) = 0 \text{ for } m < j,$$

$$q_2(j0,m\,|\,\pi,\kappa) = P_\kappa(m-j) \text{ for } j \le m < \pi,$$

$$q_2(j0,\pi\,|\,\pi,\kappa) = P_\kappa(\pi-j) + \sum_{i=1}^{\infty} P_\kappa(\pi-j+i)h^i.$$

Given the specification of the double class system as two independent single class subsystems, its immediate rewards are sums of subsystem immediate rewards, and the probability of any given immediate reward is the product of the corresponding subsystem transition probabilities. Thus is is fairly straightforward to calculate the expected immediate reward $\mathrm{E}[s^{\pi}_j]$ in state $j = (j_1,j_2,j_3)$. With respect to the possible next state $k = (k_1,k_2,k_3)$,

let F_{jk} denote the set of samples that entail $\widehat{j_1 k_1 \,|\, j_2}$ within the class 1 subsystem, and let $s(f)$ be the immediate reward resulting from $f \in F_{jk}$. Likewise, let G_{jk} denote the set of samples that entail $\widehat{j_3 k_3, k_2}$ within the class 2 subsystem, and let $s(g)$ be the immediate reward resulting from $g \in G_{jk}$. Then

$$\mathrm{E}[s_j^\pi] = \sum_k \sum_{\substack{f \in F_{jk}, \\ g \in G_{jk}}} (s(f) + s(g))\, P(f) P(g).$$

If the system event $\widehat{jk}$ entails an infeasible event for either subsystem, we have $F_{jk} = G_{jk} = \emptyset$.

In the case of the class 2 subsystem, if the event $\widehat{j_3 k_3, k_2}$ is feasible, then the specification of k_2 means that we know exactly how many demands the subsystem accepts during the transition. Thus $g \in G_{jk}$ is unique, $P(g) = q_2(j_3 k_3, k_2 \,|\, \pi_2, \kappa)$, and the immediate reward $s(g)$ is as follows.

- If $k_3 > 0,$ then $s(g) = \langle v, w(j_3, k_3) \rangle,$ where v and $w(j_3, k_3)$ are defined as in (3.5).
- If $k_3 = 0,$ then $s(g) = \langle \widetilde{v}, \widetilde{w}(j_3, k_2) \rangle,$ where

$$\begin{aligned}
\widetilde{v}_i &= 1, \quad && 1 \le i \le \min(\pi_2, b), \\
\widetilde{v}_i &= 1 - u(i - \min(\pi_2, b)), \quad && (\min(\pi_2, b) + 1) \le i \le \pi_2, \\
\widetilde{w}_i(j_3, k_2) &= 0, \quad && 1 \le i \le j_3, \\
\widetilde{w}_i(j_3, k_2) &= 1, \quad && j_3 + 1 \le i \le k_2, \\
\widetilde{w}_i(j_3, k_2) &= 0, \quad && k_2 + 1 \le i \le \pi_2.
\end{aligned}$$

In the class 1 subsystem, a feasible event $\widehat{j_1 k_1 \,|\, j_2}$ may entail that set F_{jk} have more than one element. The possible cases are as follows:

- If $j_2 = k_1 < c$ then either no demands are offered to the second day of the subsystem, or any offers for the second day are rejected. This is equivalent to the event $\widehat{j0}$ from the

original single class scenario, so we can reuse the analysis of $S(j,0)$ following (3.10) and write

$$\sum_{f \in F_{jk}} s(f)\, P(f) = S(j_1, 0).$$

• If $j_2 = k_1 = c$ then any demands offered to the second day of the subsystem are lost. This is again similar to the event $\widehat{j0}$ from the original single class scenario, except we set $h = 1$ when calculating $S(j_1, 0)$.

• If $j_2 = 0 < k_1$ then F_{jk} has exactly one element, and using (3.5),

$$s(f)\, P(f) = \langle v, w(j_1, k_1) \rangle\, q_1(j_1 k_1 \,|\, 0, \pi_1, \lambda).$$

• If $0 < j_2 < k_1 \leq c$ then F_{jk} has exactly one element and we can adjust (3.5), in order to recognize that the first j_2 allocations on day 2 of the class 1 subsystem are already taken. Specifically, replace (3.8) by two equations,

$$w_i(j_1, k_1) = 0, \quad \pi_1 + 1 \leq i \leq \pi_1 + j_2,$$
$$w_i(j_1, k_1) = 1, \quad \pi_1 + j_2 + 1 \leq i \leq \pi_1 + k_1.$$

This change gives reward $\langle v, w(j_1, k_1 | j_2) \rangle$, and we can write

$$s(f)\, P(f) = \langle v, w(j_1, k_1 | j_2) \rangle\, q_1(j_1 k_1 \,|\, j_2, \pi_1, \lambda).$$

Now that we have specified the entries $q^{\pi}_{jk} = q_1(j_1 k_1 \,|\, j_2, \pi_1, \lambda) q_2(j_3 k_3, k_2 \,|\, \pi_2, \kappa)$ of the transition probability matrix Q_π and the components $\mathrm{E}[s^{\pi}_j]$ of the expected immediate reward vector, we proceed as before. The stationary probability vector υ^π for the system satisfies the equation $\upsilon^\pi = \upsilon^\pi Q_\pi$ and the average reward per step is $\xi(\pi) = \sum_j \upsilon^\pi_j \mathrm{E}[s^\pi_j]$. The average reward for policy π, as a fraction of possible reward, is thus $\dfrac{\xi(\pi)}{\lambda + \eta\kappa}$.

3.4.3 RESULTS

We use MATLAB® to implement the formulations of the preceding two subsections. For the base case of one demand class, the objective function appears to be quasi-concave, so we can locate the optimal input $\pi \in \mathbb{Z}^+$ and the corresponding average reward by trial and error. For example, consider a system facing 10 demands per day, with a nominal capacity of 10 units and a maximum capacity of 15 units. Figure 3.1 plots average reward against π and for various values of the unit overtime cost.

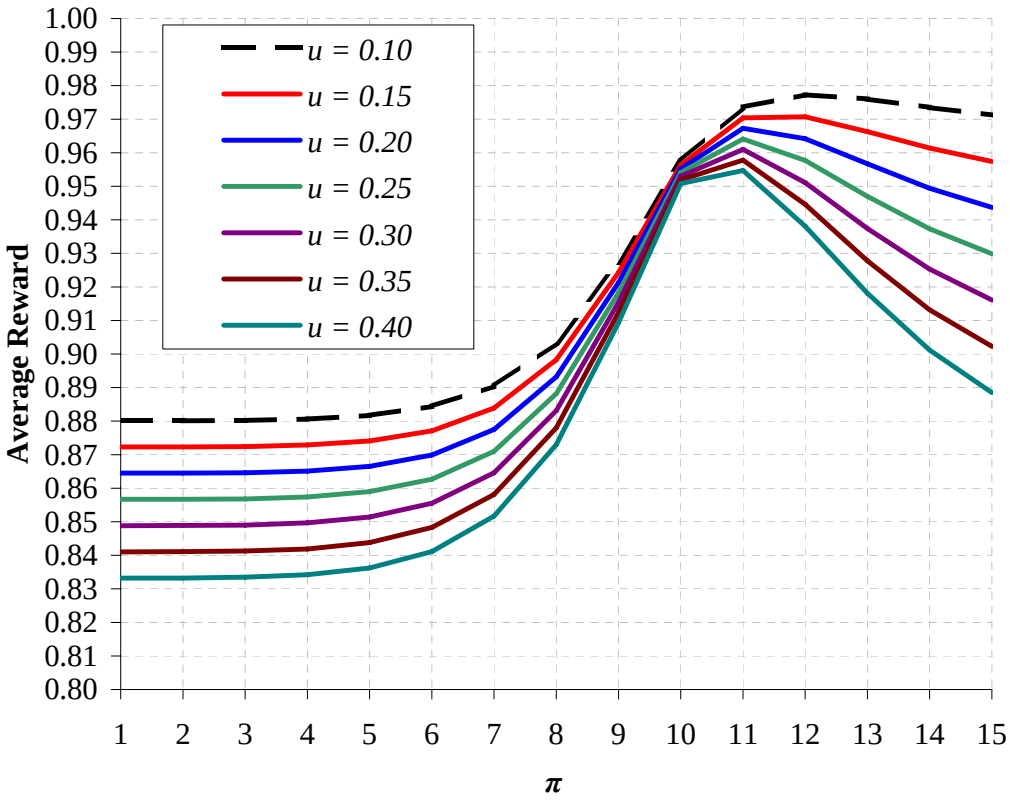


Figure 3.1: Average Reward, Base Case

For the two class case, the length of time required to generate the results is much greater. Indeed, for each input vector $\pi = (\pi_1, \pi_2)$, computation of the average reward requires approximately 75 minutes. Thus we do not give results for different values of the overtime cost. For $u = 0.25$, Figure 3.2 plots average reward against (π_1, π_2).

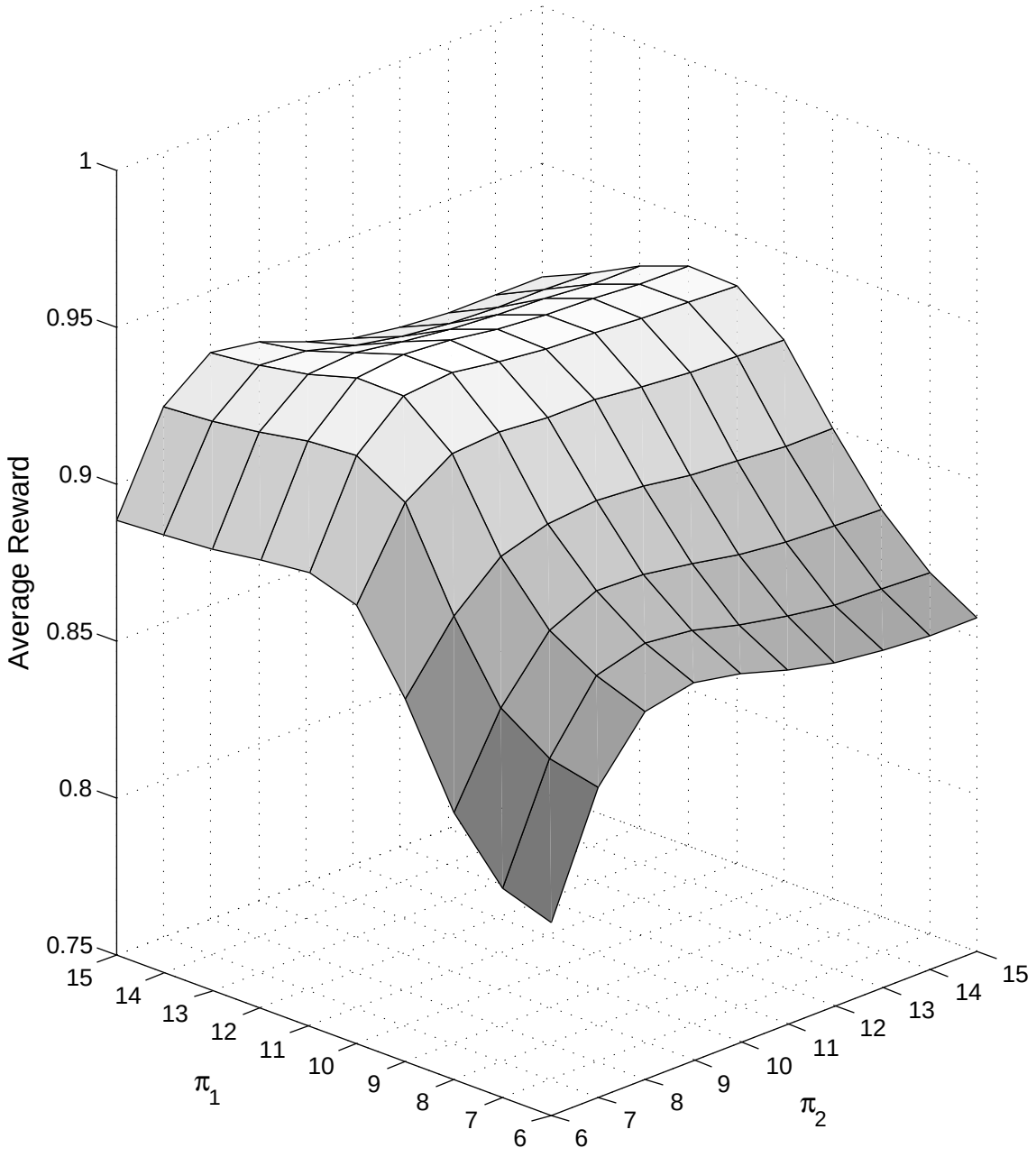


Figure 3.2: Average Reward, Two Classes

Each of the Markov chain outputs shows that the near-optimal policies may compare very well with the optimum. For the base case with $u = 0.25$, we have average reward 0.9641 from $\pi^* = 11$ and average reward 0.9577 from $\pi = 12$. With two classes and $u = 0.25$, average reward is 0.9411 from $\pi^* = (12, 9)$ and 0.9395 from $\pi = (12, 10)$.

3.4.4 FURTHER EXTENSIONS

The principles of the preceding analysis can be extended to cases where three or more demand classes exist and/or to cases where more than two lead times are possible within classes. The only requirement is that the sets of lead time options have no overlaps, as described in the basic model definition (page 54 above). Given this absence of overlap among sets of lead times, the system can be decomposed into independent subsystems.

Although we thus have a rigorous framework for the computation of $\xi(\pi)$, the implementation of the two class case in the previous subsection shows that the approach is of limited practical value. A system with a scheduling horizon of n days and capacity of c units per day has $(c+1)^n$ possible states, entailing $(c+1)^{2n}$ elements each for the transition probability matrix and the transition reward matrix. For $c = 100$ and $n = 4$, allowing 4 bytes per element, we require 76.94PB of storage[3].

Dynamic programming (DP) offers one theoretical means of improvement on this "exhaustive" analysis. DP methods for solution of MDPs were developed in the late 1950s and remain widely applicable today (Bellman, 1957; Bertsekas, 2001). Nevertheless, DP still requires knowledge or good estimates of probabilistic parameters for the system concerned, and the computation of these parameters becomes increasingly difficult for our current problem. One way around this obstacle is to use DP methods in conjunction with a stochastic simulation of the system: optimal decisions can then be progressively approximated (or "learned") from the simulation output. Researchers in the Artificial Intelligence (AI) community first developed these approximate dynamic programming (ADP) techniques, known as either "Temporal Difference" (TD) or "Reinforcement Learning" (RL). Subsequently, researchers in other fields have made significant contributions to the theory, and the name "Neuro-dynamic Programming" (NDP) has also become common.

In the next section, we introduce a specific variant of RL known as *Q*-learning and discuss the adaption of the theory to our lead time promising problem.

[3] 1PB = 1,048,576GB

3.5 REINFORCEMENT LEARNING

Despite multiplicity of names and many further levels of subtly different implementation techniques, the key characteristic of most ADP approaches is that the value function vector of standard DP can be estimated through the output of a discrete event simulation and a stochastic approximation algorithm. The formalization of this insight is particularly evident in *Q*-Learning (Watkins, 1989), one of the best known reinforcement learning methods and the one we shall employ here to derive lead time promising policies. We present below a summary development[4] of *Q*-Learning for average reward problems. Further details of this and many related methods can be found in the texts of Bertsekas and Tsitsiklis (1996), Sutton and Barto (1998), Gosavi (2003), and Powell (2007). For a different perspective on simulation-based DP, using population-based techniques, see Chang et al. (2007).

3.5.1 FROM DYNAMIC PROGRAMMING TO *Q*-LEARNING

Recall the Bellman optimality equation for a finite state, average reward MDP,

$$v^{\pi^*}(i) = \max_{a \in A(i)} \left[\sum_{j=1}^{n} p_{ij}^{a} \left(r_{ij}^{a} + v^{\pi^*}(j) \right) \right] - \rho^{\pi^*}, \tag{3.12}$$

where

n is total number of states in the problem,

$v^{\pi^*}(i)$ is the value function element for state i under the optimal policy π^*,

$A(i)$ is the set of possible actions in state i,

p_{ij}^{a} is the probability of transitioning from state i to state j under action a,

r_{ij}^{a} is the immediate reward for transitioning from state i to state j under action a,

ρ^{π^*} is the average reward under the optimal policy π^*.

[4] Although there are many correspondences, the notation used in subsection 3.5.1 is independent of what is used elsewhere in the chapter.

Define a Q-factor for each possible state-action pair (i, a) as the maximand of the Bellman equation,

$$Q_i^a \equiv \sum_{j=1}^{n} p_{ij}^a \left(r_{ij}^a + v^{\pi^*}(j) \right). \tag{3.13}$$

By substituting (3.13) back into (3.12), we have

$$Q_i^a = \sum_{j=1}^{n} p_{ij}^a \left(r_{ij}^a - \rho^{\pi^*} + \max_{b \in A(j)} Q_j^b \right) \tag{3.14}$$

$$= \mathrm{E}\left[r_{ij}^a - \rho^{\pi^*} + \max_{b \in A(j)} Q_j^b \right]. \tag{3.15}$$

Since (3.15) does not explicitly include any transition probabilities, it allows estimation the Q-factors from the output of a system simulation. This is commonly achieved by means of the stochastic approximation method of Robbins and Monro (1951), which provides an algorithmic approach to updating an estimated mean value, based on new samples of a random variable. Specifically, let s^i is the i^{th} independent sample of a random variable X and define

$$X^m \equiv \frac{\sum_{i=1}^{m} s^i}{m}, \qquad \alpha^m = \frac{1}{m}.$$

Then we have

$$X^{m+1} = (1 - \alpha^{m+1}) X^m + \alpha^{m+1} s^{m+1}.$$

Fitting (3.14) into this algorithmic framework gives

$$\left[Q_i^a\right]^{m+1} = \left(1 - \alpha^{m+1}\right) \left[Q_i^a\right]^m + \alpha^{m+1} \left(r_{ij}^a - \rho^{\pi^*} + \max_{b \in A(j)} \left[Q_j^b\right]^m \right). \tag{3.16}$$

Although ρ^{π^*} is not known in advance, we can distinguish at the outset of the algorithm one Q-factor to use as an estimate of its relative magnitude. This is sufficient to ensure that the Q-factors remained bounded and the algorithm converges.

To start the algorithm, we set the counter m and all Q-factors to zero. Distinguishing then for example Q_1^1 as our estimate of ρ^{π^*}, the complete Q-learning algorithm is

(i) Set $[Q_i^a]^0 = 0$ for all a, i.

(ii) $[Q_i^a]^{m+1} = (1 - \alpha^{m+1})\,[Q_i^a]^m + \alpha^{m+1}\left(r_{ij}^a - [Q_1^1]^{m+1} + \max_{b \in A(j)} \left[Q_j^b\right]^{m+1}\right)$.

After sufficient iterations to estimate the *Q*-factors with a desired degree of precision, the learned policy is read as the action index of the *Q*-factor with greatest magnitude for each state,

$$\pi^*(i) = \arg\max_{a \in A(i)} \left[Q_i^a\right].$$

3.5.2 IMPLEMENTATION OF *Q*-LEARNING

Convergence of approximate dynamic programming algorithms can often be expedited by exploiting any inherent structure of the modeled system, such as renewal cycles (Bertsekas, 2001). In our lead time promising model, one salient structural aspect is the relation between rewards generated by lead time offers that concern the same production day. If our implementation of *Q*-learning explicitly relates a lead time offer to a low class demand for production on a given date to any later offers to higher class demands for production on the same day, approximation of the *Q*-factors may be more direct, yielding faster convergence. Given the potentially large state space for the problem, this exploitation of structure may make a significant difference to the time needed to generate a solution of acceptable quality.

In the real world, relation of demands to specific production dates is accomplished by the unique indexing of the calendar system. In order to capture this information in simulation, let successive positive integers denote different production dates. We extend the indexing system to individual demands by letting d_j^k represent the j^{th} demand to arrive during the k^{th} production day. The expression $d_j^k = i$ indicates that d_j^k is a demand of class i. All demands arriving during production date k will be scheduled for production on some later day, *i.e.*, a day with index in the set $\{k+1,\ k+2,\ \ldots\}$.

Although integer indices allow us to distinguish different production dates, this does not immediately induce any renewal structure. One possible recourse is as follows: a lead time

offer of y days to demand d_j^k is construed as an action a taken for date $k + y$, and because date $k+y$ at that moment is only an instance of a date with y days remaining to production, we can abstractly consider action a to be taken for day y in the production horizon. Since each production date progresses from the furthest day in the production horizon to day 0, relating actions to the time remaining until production entails that each production date is an instance of a renewal cycle: a "scheduling" interval of fixed size, during which lead time offers and associated rewards occur.

Whereas a lead time offer corresponding to day $k + y$ is a range element of the policy function π given by (3.1), the renewal perspective described above entails that y must in some way serve as an argument to π. The form of π must be modified to allow for this. For convenient reference, we will use the terms "demand-centric" for the policy form given in (3.1) and "day-centric" for the policy form that considers actions to be based on days. Retain π to denote the policy corresponding to the demand-centric approach, and let π' be the policy corresponding to the day-centric approach.

In order to implement policy π', we need to distinguish and relate rewards earned for each date k, so π' must prescribe an action for demand d_j^k with respect to date k alone. The action for a single date can only be binary: offer a lead time corresponding to production on that day (action 1), or do not offer it (action 0). Thus, if three or more lead time offers are formally possible for any demand, we need some additional structure to enable the consistent application of π'. This structure follows naturally from the assumption that customers prefer shorter lead times. For example, if the shortest possible lead time offer for demand d_j^k is y days, we can evaluate π' successively for date $k + y$, date $k + y + 1$, and so forth, making the corresponding lead time offer as soon as action 1 is returned. If action 1 is not returned before we reach the date that entails the longest possible lead time, we return lead time offer corresponding to that latter date. Thus, with this approach, the number of policy evaluations for a given demand will always be one fewer than the number of possible lead time offers.

In order to clarify the differences in implementation required by the demand-centric or the day-centric approach, the following example presents an excerpt from the sample path of a small system. Following the specification of the sample path, we show for each approach how the system state may be recorded and the Q-factors updated.

3.5.3 EXAMPLE

Assume we have only two classes of demand and that we offer non-overlapping lead time options, as in the two class Markov chain analyzed above (page 54). We assume that demand within classes occurs according to independent Poisson processes, with means of 1 demand per day for class 1 and 3 demands per day for class 2. The example starts at the beginning of the 9^{th} production day, and we suppose that the loads already scheduled for forthcoming production dates are as follows:

- $k = 10:$ 3 demands of class 2
- $k = 11:$ 2 demands of class 2
- $k = 12:$ 1 demand of class 2
- $k = 13:$ empty.

These loads are represented in the leftmost chart in Figure 3.3 below. The vertical axis is taken to represent the boundary between the current production date (in this case, $k = 9$) and future production dates ($k \geq 10$). The production load for the current date is not specified, since it is considered irrelevant to the question of lead time offering.

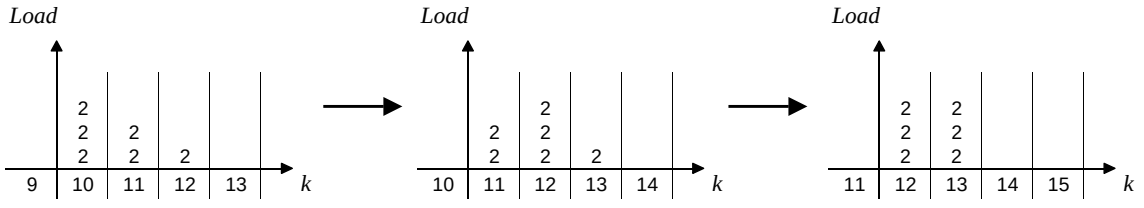


Figure 3.3: System Evolution Example

The system state represented in the leftmost chart in Figure 3.3 evolves through the incidence of each new demand, the associated lead time offer ϕ, and customer response,

C. The lead time offer ϕ implies production on some date k, while $C \in \{A, R\}$ indicates whether the offer is accepted (A) or rejected (R) by the customer. The sample demand path, lead time offers, and customer responses across productions dates 9, 10, and the start of production date 11 are taken to be as follows:

$$k = 9 \begin{cases} \underline{\text{Demand}} & \underline{\phi} & \underline{C} \\ d_1^9 = 2 & 3\ (\Rightarrow k = 12) & \text{A} \\ d_2^9 = 2 & 3\ (\Rightarrow k = 12) & \text{A} \\ d_3^9 = 1 & 1\ (\Rightarrow k = 10) & \text{A} \\ d_4^9 = 2 & 4\ (\Rightarrow k = 13) & \text{A} \end{cases}$$

$$k = 10 \begin{cases} \underline{\text{Demand}} & \underline{\phi} & \underline{C} \\ d_1^{10} = 2 & 3\ (\Rightarrow k = 13) & \text{A} \\ d_2^{10} = 1 & 1\ (\Rightarrow k = 11) & \text{A} \\ d_3^{10} = 1 & 1\ (\Rightarrow k = 11) & \text{A} \\ d_4^{10} = 2 & 4\ (\Rightarrow k = 14) & \text{R} \\ d_5^{10} = 2 & 3\ (\Rightarrow k = 13) & \text{A} \end{cases}$$

$$k = 11 \begin{cases} \underline{\text{Demand}} & \underline{\phi} & \underline{C} \\ d_1^{11} = 1 & 1\ (\Rightarrow k = 12) & \text{A} \\ \vdots & & \end{cases}$$

The remaining two charts in Figure 3.3 show the evolution of the system, given the above sample path. The second chart shows the system loads at the start of day 10, and the third chart shows the state at the start of day 11.

DEMAND-CENTRIC FORMULATION

This approach to lead time promising is oriented around the individual demands as entities: given a new demand of class i, we ask explicitly, what lead time offer should we make?

As in (3.1), our policy function returns a element from the set of lead times available to class i. Although the state vector $\sigma(t)$ seen by the arriving demand is in principle the load on all days in the system horizon, we approximate it here by the projection $\hat{\sigma}(t)$, the load on the possible target days for the demand: $\hat{\sigma}(t) = (l_1^t, l_2^t)$, where $l_n^t \in \mathbb{N}$ denotes the load of the day corresponding to the n^{th} shorthest lead time available to d_j^k. Accordingly, given demand d_j^k of class i, the argument of the policy function is the vector $(i, \hat{\sigma}(t))$. A lead time offer is an action $a \in \{1, 2\}$, where $a = n$ corresponds to an offer of the n^{th} shortest lead time available to d_j^k.

With this formulation, the sample path from above will be recorded as follows (the variable t is suppressed in the example):

$k = 9$

Demand	$(i, \hat{\sigma})$	a	C
$d_1^9 = 2$	(2,1,0)	1	A
$d_2^9 = 2$	(2,2,0)	1	A
$d_3^9 = 1$	(1,3,2)	1	A
$d_4^9 = 2$	(2,3,0)	2	A

$k = 10$

Demand	$(i, \hat{\sigma})$	a	C
$d_1^{10} = 2$	(2,1,0)	1	A
$d_2^{10} = 1$	(1,2,3)	1	A
$d_3^{10} = 1$	(1,3,3)	1	A
$d_4^{10} = 2$	(2,2,0)	2	R
$d_5^{10} = 2$	(2,2,0)	1	A

$k = 11$

Demand	$(i, \hat{\sigma})$	a	C
$d_1^{11} = 1$	(1,3,3)	1	A
$\vdots$			

As usual, we define Q-factors as state-action pairs. The "state" is the policy argument,

so a Q-factor id denoted uniquely by $(i, \hat{\sigma}; a)$. With the demand-centric formulation, two approaches to updating Q-factors are plausible. First, updates could occur where demands are of the same type:

- Updating for class 1 demands: $\ldots \rightarrow (1,3,2;1) \rightarrow (1,2,3;1) \rightarrow (1,3,3;1) \rightarrow (1,3,3;1) \rightarrow \ldots$
- Updating for class 2 demands: $\ldots \rightarrow (2,1,0;1) \rightarrow (2,2,0;1) \rightarrow (2,3,0;2) \rightarrow (2,1,0;1) \rightarrow (2,2,0;2) \rightarrow (2,2,0;1) \rightarrow \ldots$

This will, however, certainly lead to a greedy policy for both demand types.

As a second approach, updates could occur with all state transitions:

- $\ldots \rightarrow (2,1,0;1) \rightarrow (2,2,0;1) \rightarrow (1,3,2;1) \rightarrow (2,3,0;2) \rightarrow (2,1,0;1) \rightarrow (1,2,3;1) \rightarrow (1,3,3;1) \rightarrow (2,2,0;2) \rightarrow (2,2,0;1) \rightarrow (1,3,3;1) \rightarrow \ldots$

In this case, the first and last Q-factors, which concern lead time decisions for the same production day, are separated by eight other Q-factors, all but one of which denote lead time decisions for other production days. The value of these intervening Q-factors may be rather irrelevant for the first and last Q-factors, so their presence will greatly complicate the updating process.

DAY-CENTRIC FORMULATION

This formulation is oriented around the production days as entities: given a new demand of class i, we consider assigning it to the production date corresponding to the shortest possible lead time and ask explicitly, should we make this offer, or not? If we decide not to make the corresponding offer, we then ask the same question for next available production date[5]. Thus, in contrast to (3.1), our policy function must return binary responses. Since the set of lead times available to class i is prescribed, we can replace the class variable in

[5] A demand considered for the last available production date will always receive the corresponding offer.

the policy argument by the index of the day being considered. Also, as in the demand-centric formulation, we utilize the project the state variable $\sigma(t)$ to $\hat{\sigma}(t) = (l_1^t, l_2^t)$, where $l_n^t \in \mathbb{N}$ denotes the load of the day corresponding to the n^{th} shortest lead time available to d_j^k. Accordingly, when a lead time offer of y days is evaluated for demand d_j^k, let $(y, \hat{\sigma}(t))$ be the corresponding argument of the policy function. The policy function returns action $a = (b, k + y)$, where $b \in \{0, 1\}$, indicates whether a lead time offer corresponding to production on date $k + y$ is made ($b = 1$) or not made ($b = 0$).

With this formulation, the sample path from above will be recorded as follows (the variable t is again suppressed).

$k = 9$

Demand	$(y, \hat{\sigma})$	a	C
$d_1^9 = 2$	(3,1,0)	$(1, 12)$	A
$d_2^9 = 2$	(3,2,0)	$(1, 12)$	A
$d_3^9 = 1$	(1,3,2)	$(1, 10)$	A
$d_4^9 = 2$	(3,3,0)	$(0, 12)$	NA
$d_4^9 = 2$	(4,3,0)	$(1, 13)$	A

$k = 10$

Demand	$(y, \hat{\sigma})$	a	C
$d_1^{10} = 2$	(3,1,0)	$(1, 13)$	A
$d_2^{10} = 1$	(1,2,3)	$(1, 11)$	A
$d_3^{10} = 1$	(1,3,3)	$(1, 11)$	A
$d_4^{10} = 2$	(3,2,0)	$(0, 13)$	NA
$d_4^{10} = 2$	(4,2,0)	$(1, 14)$	R
$d_5^{10} = 2$	(3,2,0)	$(1, 13)$	A

$k = 11$

Demand	$(y, \hat{\sigma})$	a	C
$d_1^{11} = 1$	(1,3,3)	$(1, 12)$	A
$\vdots$			

With the day-centric formulation, when recording Q-factors, we suppress the element k in the action vector: the Q-factor for state $(y, \hat{\sigma})$ and action a is represented by $(y, \hat{\sigma}; b)$. This is because we aim to generalize from the specific experience on date k to a general policy for production days indexed solely by y, the number of days remaining until production begins. Nevertheless, the programmatic record of k allows us to update Q-factors in a way that reflects the connection between lead time decisions made with respect to the same production day. Given that we assign an arbitrary terminal state T (and an associated constant terminal reward) to each production day, the updating pattern with the day-centric formulation is:

- $k = 10:$ $\quad \ldots \rightarrow (1, 3, 2; 1) \rightarrow T$
- $k = 11:$ $\quad \ldots \rightarrow (1, 2, 3; 1) \rightarrow (1, 3, 3; 1) \rightarrow T$
- $k = 12:$ $\quad \ldots \rightarrow (3, 1, 0; 1) \rightarrow (3, 2, 0; 1) \rightarrow (3, 3, 0; 0) \rightarrow (1, 3, 3; 1) \rightarrow \ldots$
- $k = 13:$ $\quad \ldots \rightarrow (4, 3, 0; 1) \rightarrow (3, 1, 0; 1) \rightarrow (3, 2, 0; 0) \rightarrow (3, 2, 0; 1) \rightarrow \ldots$
- $k = 14:$ $\quad \ldots \rightarrow (4, 2, 0; 1) \rightarrow \ldots$

Here, the first and last Q-factors shown for $k = 12$, which were separated by eight other Q-factors in the demand-centric formulation, are now only separated by two other Q-factors. This outcome will be true in general, since are fewer demands are expected to be assigned to any one day than to the system as a whole, in any fixed period. The greater proximity between related Q-factors should allow for faster and more accurate approximate of the desired policy.

3.5.4 Additional Considerations

The day-centric formulation has the advantage of allowing a immediately plausible way to reduce the dimension of the state vector. Because we consider assignment of a demand to

each possible production date in turn, the state vector $\sigma(t)$ can be further truncated to the scalar value l_y^t, the current load on the date under consideration. If policy π' dictates that demand d_j^k should not be assigned to date $k+y$, the next policy evaluation will nonetheless include the state information l_{y+1}^t, so action a is still potentially informed by all load information from the possible target days. With the demand-centric formulation, any similar truncation of the load component results in a certain loss of information, since π is evaluated only once to generate action a.

The recognition of rewards is, however, somewhat complicated by the day-centric perspective. In the demand-centric formulation, each reward results by definition from a single policy evaluation, and thus the *Q*-learning framework can be unambiguously applied. Contrastingly, in the day-centric formulation, a reward may result from two or more policy evaluations, each pertaining to a different *Q*-factor. Thus, before finally being accepted for production on a certain day, a demand may have been rejected by one or more earlier days. Since the nature of an optimal policy is such that any initial rejections - as well as ultimate acceptance - should be prescribed, each new demand event in the day-centric formulation must be followed by a *Q*-factor update for *all* days that make a decision with respect to it. In order to reflect the outcome consistently across each of these updates, we divide the relevant immediate reward equally among them. This approach must be further amended for rescheduling of demands, since immediate rewards from rescheduling actions are always negative, so simply dividing them among updates cannot reveal optimal decisions (rewards from rejection would always be greater than rewards from acceptance). Thus immediate rewards for rescheduling actions are taken as the difference between the true reward and the worst possible reward. Each rescheduled demand is followed by a *Q*-factor update for *all* days that make a decision with respect to the rescheduling, with the immediate reward divided equally among them.

3.6 SIMULATION

The simulation is implemented in the C language, using an event scheduling approach (Law and Kelton, 2000). See the appendix to this chapter for implementation notes and listing of the program.

Runtime of the simulation entails two distinct phases:

(i) given a set of input parameters[6], the program engages in learning across a specified number of simulated system days;

(ii) the program evaluates the performance of the learned policy through a set of independent replications, using otherwise the same system parameters that define the learning phase.

Within each simulation phase, two different operational modes arise. In "regular" mode, the system handles newly arriving demands by making lead time offers and receiving non-negative rewards. In "rescheduling" mode, the system handles the effect of inventory supply disruption, by reassigning demands as necessary to later production days and receiving negative rewards.

Figure 3.4 gives a schematic summary of the simulation. Solid lines indicate progression during regular mode, while dashed lines indicate progression during rescheduling mode.

3.6.1 ORGANIZATION OF EXPERIMENTS

Our evaluation of the performance of Q-learning is organized around the assumption that, as in most models, certain elements are more managerially significant than others. Generally, these are variables that a manager can to some extent control, subject of course

[6]Besides the input parameters discussed in this section, there are several simulation inputs that control options such as the duration of the learning phase, the number of replications, the minimum number of visits required before Q-factors are included in the policy determination, *etc.* These other parameters are noted in the appendix.

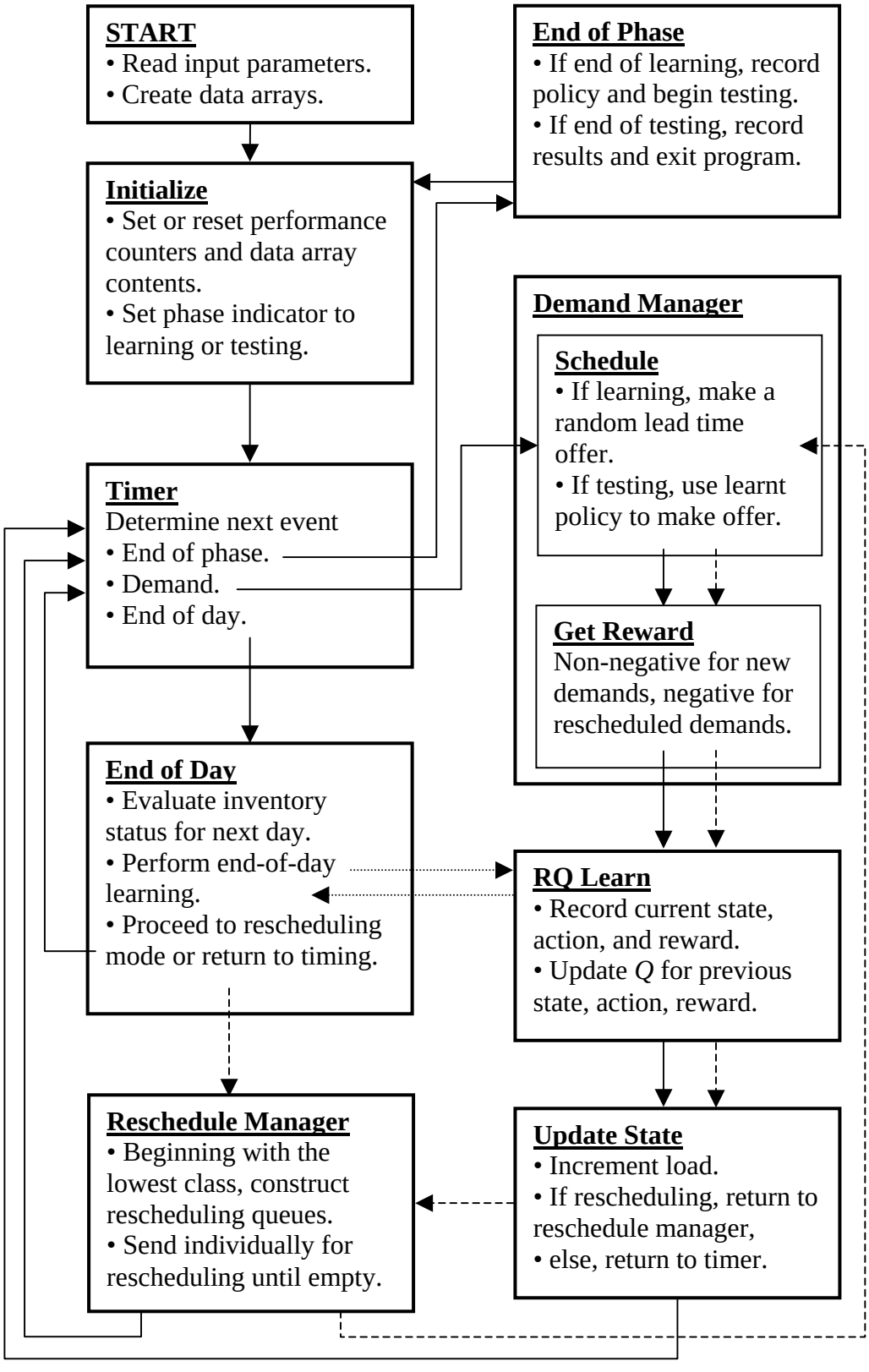


Figure 3.4: Simulation Schematic Summary

to appropriate costs for doing so. For the lead time promising system, the set $\mathcal{V}$ of such variables includes:

- N_i, the number of lead times to offer to each customer class,
- b, the nominal capacity of the system,
- u, the unit cost associated with overtime,
- τ, the probability of supply chain disruption.[7].

In contrast, some elements have more the nature of *parameters* which describe the business conditions faced by the firm, at least in the short run. Here, the set $\mathcal{P}$ of such parameters includes:

- λ, total mean demand,
- n, number of customer classes,
- δ, the distribution of demand among classes (n-vector),
- r, the potential rewards from each class (n-vector),
- h_{ij}, the reject probabilities associated with different lead times for each class,
- c, the maximum capacity of the system.

We therefore proceed by fixing first several distinct sets of values for elements in $\mathcal{P}$. For each possible set of values in $\mathcal{P}$, our simulation experiments then show the performance of Q-learning for different choices of the elements in $\mathcal{V}$.

Within $\mathcal{P}$, we restrict our consideration to cases with $n = 2$ or $n = 3$. With $n = 2$ we take $\delta = (0.3, 0.7)$, meaning the class with highest priority constitutes on average 30% of

[7]While perhaps not directly controllable, this may be important for choosing suppliers or negotiating contracts with them.

demand. With $n = 3$, we take $\delta = (0.1, 0.2, 0.7)$, so here the class with highest priority constitutes on average 10% of demand.

For both possible values of n, we define three different reward structures. These are detailed in Table 3.1. For each structure and value of n, the table shows the vector r, the components of which correspond to the different rewards for each class, and the expected reward under each structure, given the values of δ specified above. In each case, when moving from $n = 2$ to $n = 3$, the definition of the reward structure entails the addition of an intermediate reward level, while the reward for the upper class and the reward for the lower class remain the same. Reward structure A is intended as a base case, with a relatively broad distribution of reward values and intermediate expected reward values. With reward structure B, the distribution of reward values is shifted more toward the top class value, giving higher expected reward values. With reward structure C, the distribution of reward values is shifted toward the lowest class value, giving lower expected reward values.

Structure	$n = 2$	$n = 3$
A	$r = (1.0, 0.50)$ $\Rightarrow \mathrm{E}[r] = 0.650$	$r = (1.0, 0.75, 0.50)$ $\Rightarrow \mathrm{E}[r] = 0.605$
B	$r = (1.0, 0.75)$ $\Rightarrow \mathrm{E}[r] = 0.825$	$r = (1.0, 0.90, 0.75)$ $\Rightarrow \mathrm{E}[r] = 0.805$
C	$r = (1.0, 0.25)$ $\Rightarrow \mathrm{E}[r] = 0.475$	$r = (1.0, 0.50, 0.25)$ $\Rightarrow \mathrm{E}[r] = 0.375$

Table 3.1: Reward Structures and Expected Values

In order to specify total mean demand and maximum capacity as well as the number of classes, we define the four demand scenarios in Table 3.2.

For the reject probabilities h_{ij}, we assume for all classes that the probability of rejection for a lead time offer increases by 10% for each additional day beyond the most favorable

Scenario	n	λ	c
1	2	10	15
2	2	40	50
3	2	100	125
4	3	100	125

Table 3.2: Demand Scenarios

lead time for the class. The probability that a class i demand will reject a lead time of x days is thus $0.1(\phi_{i1} - x)$

Since any of these four scenarios can be evaluated with any one of the three reward structures, we have a total of 12 parameter scenarios from $\mathcal{P}$ to serve as context for evaluation of a set of values from $\mathcal{V}$ (*i.e.*, for given N_i, b, u, and τ). By varying simultaneously at most two variables in $\mathcal{V}$ while holding the others constant, we can visualize their individual impact on the performance of the system. By varying also the reward structure or the demand scenario, we can see whether the performance impact of each variable in $\mathcal{V}$ appears consistent across these contexts.

Turning now to the variables in $\mathcal{V}$, we restrict the values of N_i to a single number N for our evaluations. In particular, we compare the performance with $N = 2$ or $N = 3$ in the simulations. As in the Markov Chain analysis of section 3.4, we assume that the set of all lead times forms an unbroken sequence, $1, 2, \ldots, N, N+1, \ldots, nN$. Base cases for the other elements of $\mathcal{V}$ are $b = \lambda$, $u = 0.25$, and $\tau = 0.100$.

In terms of these different scenarios, variables, and constants, the simulation sets performed are listed in Table 3.3. *All simulations are conducted for all three reward structures, so this aspect is not noted.* The simulation sets are numbered for ease of reference to the

tables and charts of results in subsection 3.6.2.

Set Number	Variables	Constants	Demand Scenarios
1	$N \in \{2, 3\}$, $\tau \in [0.00, 0.150]^a$	$u = 0.25$, $b = \lambda$	1, 2, 3, 4
2	$u \in [0.0, 0.4]^b$	$N = 3$, $b = \lambda$, $\tau = 0.100$	4
3	$b \in [95, 115]^c$	$N = 3$, $b = \lambda$, $u = 0.25$	4

[a] Using subintervals of length 0.025 for τ.
[b] Using subintervals of length 0.01 for $u \in [0.00, 0.05]$ and subintervals of length 0.05 otherwise.
[c] Using subintervals of length 5 for b.

Table 3.3: Simulation Sets

For each distinct combination of parameters and variables in a simulation set, the discussion in sections 3.5.3 and 3.5.4 above suggests three possible policy types to determine through *Q*-learning.

Type 1: Day-centric policy with state approximated by load of day under consideration.

Type 2: Day-centric policy with state approximated by load of all possible target days for given demand class.

Type 3: Demand-centric policy with state approximated by load of all possible target days for given demand class.

While all three policy types are learned for demand scenarios 1 and 2, and for scenario 3 when $N = 2$, only policy type 1 is learned for scenario 3 when $N = 3$ and demand scenario 4. In this latter case, policy types 2 and 3 entail a state space of intractable size[8].

[8] The state space is exponential in N for policy types 2 and 3 (a total of $(c + 1)^N$ states for each class), but linear in N for policy type 1 (a total of $N(c + 1)$ states for each class). Thus, allowing 8 bytes to store

In order to provide a baseline to the policies generated by these three methods, we also use simulation to determine the performance of a heuristic, "policy type 0," which focuses purely on immediate rewards:

> Type 0: offer each demand the first available lead time within its class for which the immediate reward will be positive.

A final practical consideration for our experiments concerns a limitation of the Q-learning approach: the learned policy may be incomplete, since rare events, *i.e.*, system states which are visited with small probability, may have been insufficiently visited during learning to yield a reliable policy[9]. Such a state may nevertheless be visited in the testing phase of an experiment. We thus need to extend our learned policies in some way, in order to handle these eventualities.

One solution to the rare event problem would be simply to choose randomly from the set of feasible actions whenever we reach a state where the learned policy is incomplete. This is unsatisfactory, however, because policies that are based on a larger state space, such as our types 2 and 3, will have relatively fewer visits to each state per unit time, and thus greater incompleteness after a learning period of specified duration. The testing of these policies will then require a greater number of random actions, and the overall results will most likely not reflect the quality of the complete portion of the policy (unless the average reward from the learned policy were very similar to the reward from a policy of random actions). Another possibility would be to increase the length of the learning period for a policy with a larger state space. This may entail exponential increases in simulation time, however, and also makes it difficult to compare efficiency in learning across policies.

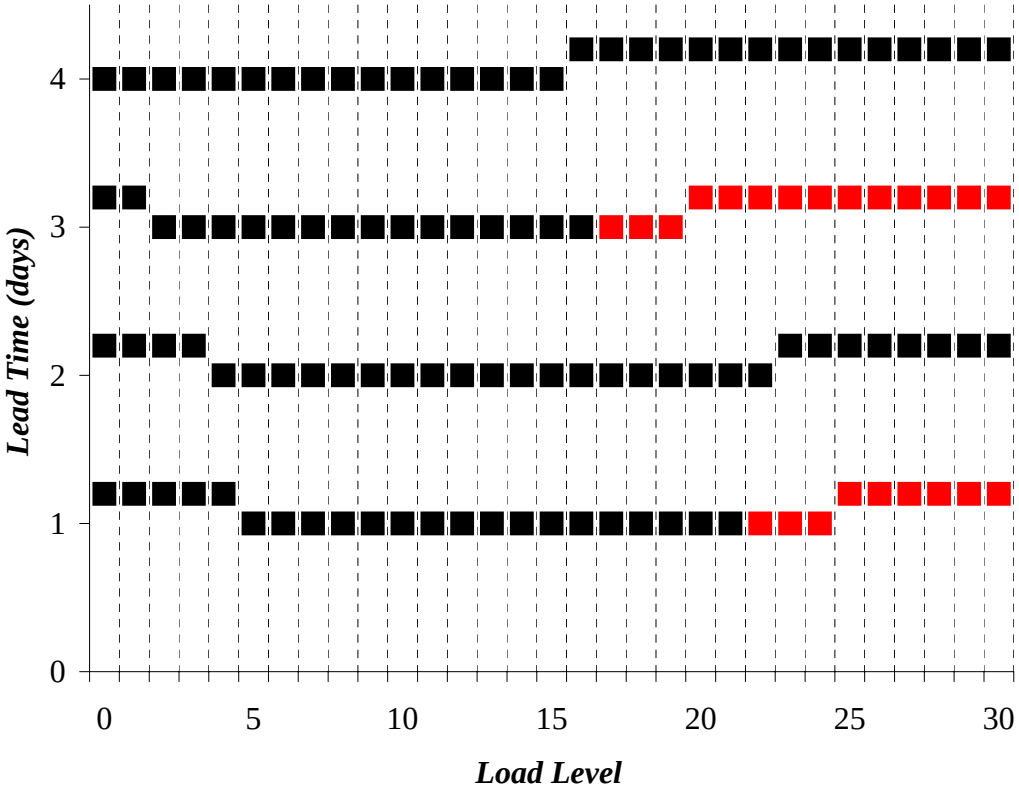
Instead, we implement a two-stage solution: first we extrapolate policy type 1 to cover rare events, then we augment policy types 2 and 3 with type 1 actions wherever they are

each Q-factor as a double precision floating point value, 4 bytes to store each visit counter as an integer, and providing for rescheduling mode as well as regular mode, we would need 2,670,540,192 bytes (2.487GB) of RAM to learn three policies in scenario 4.

[9] In our experiments, we include in the learned policy only states that have been visited at least 100 times during learning.

incomplete. Extrapolation of policy type 1 can be plausibly done, since the state space for each day is ordered. If a learned policy covers a subset of the states for a day and shows a threshold level for acceptance, we can assume that the "reject" action should be extended to *all* states above the threshold, even if the states at the upper extremity were not explicitly decided through learning. Likewise, we can assume that the "accept" action should be extended to *all* states below the threshold, even if the states at the lower extremity were not explicitly decided through learning.

Figure 3.5: Extrapolation of a Learned Type 1 Policy
"Accept" actions are shown by black blocks, and "reject" actions are shown by red blocks. Blocks that are situated on the line indicate the learned policy, while blocks that are slightly raised indicate actions extrapolated from it.



As an illustration, consider Figure 3.5, which shows a type 1 policy chart for a small system: $\lambda = 20$ and $n = N = 2$, so there are four days in the scheduling horizon. The horizontal sequences of blocks in the chart indicate actions to be taken when considering a lead time of x days for a new demand. At each load level, "accept" actions are shown by

black blocks, and "reject" actions are shown by red blocks[10]. For example, the sequence of blocks for $x = 1$ shows that we should accept demands for that day as long as the current load is 21 units or less. Blocks that are situated on the line indicate the learned policy, while blocks that are slightly raised indicate actions extrapolated from it. Extrapolation of the policy here incurs no ambiguity, since the threshold vector is clearly $\pi = (21, 16)$. Type 1 policy thresholds could be ambiguous, but this did not occur in our experiments.

3.6.2 RESULTS

Results were generated on a 1.7 GHz CPU with 512MB of RAM. *Q*-learning for all policies was based on 10^7 simulated days of demand activity, and all policies were tested with 50 independent replications of 10^5 simulated days of demand activity. Common random numbers were employed for the i^{th} replication of each policy test, in order to achieve variance reduction. Examples of total runtime for learning and testing in simulation set 1 are given in Table 3.4. These are average times to generate results for a single value of τ with a single reward structure. In simulation set 2 (*resp.*, set 3), average runtime for given N and u (N and b) is similar to the average runtime of demand scenario 4 in simulation set 1 for given N and τ.

Performance for all policies is stated in the following pages as the average fraction earned from each potential unit of reward. This corresponds to the performance measure derived in the Markov chain analysis of Section 3.4. Stating this measure facilitates policy comparisons across different reward structures, since the average reward from a system with higher expected value per demand could easily be large in absolute magnitude, even though the policy were capturing a relatively small fraction of the potential reward.

Detailed output from the simulations appears in tables on pages 93 - 107. For each of the variable values in each simulation set, the tables show the mean performance of

[10]Since a demand considered for the longest possible lead time in its class will always be accepted (up to capacity c), days 2 and 4 have no red blocks.

Demand Scenario	N	Learning (all policies)	Testing (per policy)
1	2	9.50	1.45
	3	13.06	1.52
2	2	39.86	5.42
	3	60.05	5.82
3	2	108.03	13.41
	3	42.04	12.15
4	2	38.19	12.12
	3	42.82	12.35

Table 3.4: Simulation Set 1, Average Runtimes (minutes)

each policy type, and a 95% confidence interval. The confidence intervals are narrow: for example, across the 182 observations for reward structure C, simulation set 1, the average half-width is 0.08% of the mean value, and the standard deviation of confidence interval half-width for the same observations is 0.06% of the mean value. All other data exhibit similarly small average half-widths and standard deviations across confidence intervals.

In order to facilitate the discussion of results, graphs of the mean performances of policy type 0 versus the best mean performances among learned policies appear on pages 108 - 113. Confidence intervals are omitted from these graphs, due to the general narrowness noted above. While a small number of policy performances are statistically indistinguishable, these cases are also suggested by the proximity of means in the graphs.

SIMULATION SET 1

The objective of this set is to evaluate the performance of policies with respect to variations in N, the number of lead times offered per class, and τ, the probability of supply chain disruption. Within each reward structure, for each trial value of τ, we have results for $N = 2$ and $N = 3$. Color-coding is used in the tables. The result from the best-performing policy across both values of N is given in bold text and highlighted green. The best-performing policy for the alternative value of N is highlighted yellow. In cases where

there are two or more policies with statistically indistinguishable best performance, both are highlighted. This does not occur in evaluations of the best policy across both values of N, however, and it only occurs in three cases[11] of the best policy for the alternative value of N.

For simulation set 1, Tables 3.5 and 3.6 summarize the best outcomes with respect to choice of N and choice of policy type. In the former, instances where $N = 2$ yields the best result are highlighted; in the latter, instances of best performance from policy types 2 or 3 are highlighted respectively yellow and green.

As far as choice of N is concerned, we see a clear trend towards $N = 3$ in the cases of higher mean demand (scenarios 3 and 4) and higher disruption probability, while $N = 2$ tends to perform better in the cases of lower mean demand (scenarios 1 and 2) and lower disruption probability. Of course, increasing disruption probability effectively entails an increase in the arrival rate to the system, since more orders have to be rescheduled. Since the mean and variance of the Poisson distribution are identical, we can suggest that larger values of N alleviate the deterimental impact of increased demand uncertainty. This is the benefit that we would expect from increased lead time flexibility. Contrastingly, when demand uncertainty is low, the advantage of longer lead times is offset by the increased likelihood of customer rejection, and the less flexible policy performs better.

In evaluating the performance of policy types, bear in mind that policy types 2 and 3 are not evaluated in cases where the state space is prohibitively large (see discussion on page 79). Thus, while it is clear that the *Q*-learned policies perform invariably better than the heuristic (policy type 0), we cannot draw an overall conclusion as to which policy type *in principle* performs best. The higher success rate of policy types 2 and 3 in instances of lower mean demand and lower system capacity (scenarios 1 and 2) suggest, as is reasonable, that the extra information afforded by a vectorial representation of loads for these

[11]The three cases are (1) reward structure A, simulation set 1, demand scenario 2, $\tau = 0.150$; (2) reward structure A, simulation set 1, demand scenario 3, $\tau = 0.025$; (3) reward structure B, simulation set 1, demand scenario 3, $\tau = 0.100$.

		τ (disruption probability)							
Demand Scenario	Reward Structure	0.000	0.025	0.050	0.075	0.100	0.125	0.150	Row Average
1	A	2	2	2	2	2	2	2	2.000
	B	3	3	2	3	3	3	3	2.857
	C	3	3	3	3	3	3	3	3.000
2	A	3	3	3	2	3	2	2	2.571
	B	3	3	3	3	3	3	3	3.000
	C	3	2	2	2	3	2	2	2.286
3	A	3	3	3	3	3	3	3	3.000
	B	3	3	3	3	3	3	3	3.000
	C	2	3	3	3	3	3	3	2.857
4	A	2	3	3	3	3	3	3	2.857
	B	3	3	3	3	3	3	3	3.000
	C	3	3	3	3	3	3	3	3.000
Column Average		2.750	2.833	2.750	2.750	2.917	2.750	2.750	

Table 3.5: Performance Summary for $N = 2$ vs. $N = 3$

		τ (disruption probability)							
Demand Scenario	Reward Structure	0.000	0.025	0.050	0.075	0.100	0.125	0.150	Row Average
1	A	1	1	1	1	1	1	1	1.000
	B	3	3	2	3	3	2	3	2.714
	C	1	1	1	1	1	1	1	1.000
2	A	2	2	2	1	2	1	1	1.571
	B	2	2	2	2	2	2	2	2.000
	C	2	1	1	1	1	2	1	1.286
3	A	1	1	1	1	1	1	1	1.000
	B	1	1	1	1	1	1	1	1.000
	C	2	1	1	1	1	1	1	1.143
4	A	1	1	1	1	1	1	1	1.000
	B	1	1	1	1	1	1	1	1.000
	C	1	1	1	1	1	1	1	1.000
Column Average		1.500	1.333	1.250	1.250	1.333	1.250	1.250	

Table 3.6: Performance Summary with Respect to Policy Type

policies is beneficial. The distribution of successes between policy types 2 and 3 within this same region is, however, worth noting. Policy type 2 does exceptionally well on the combination of intermediate demand, intermediate capacity, and greater expected demand value (demand scenario 2, reward structures A and B). Policy type 3 does exceptionally well on the combination of the smallest demand, smallest capacity, and greatest expected demand value (demand scenario 1, reward structure B). Despite the presumable benefit to both policy types of vectorial load information, we may see here a trade-off between the directness of updating in policy type 2 (day-centric formulation) and the burden of adding the "time remaining" variable to the state space of this policy. When demand and capacity are lowest, entailing the smallest number of *Q*-factors, the extra burden of the "time remaining" variable cannot be offset by greater directness in updating. With an intermediate demand, capacity, and *Q*-factor count, the value of more direct updating is worthwhile.

Although it is possible that policy types 2 and 3 could perform better if computational capacity constraints were greatly relaxed, policy type 1 appears to be an indispensable recourse in the current context. Moreover, referring to the detailed tables of results, we see that the performance of policy 1 is usually competitive, even when it is not best. For example, on simulation set 1, demand scenario 1, reward structure B (where policy 3 has the greatest number of successes), the average difference between policy 1 performance and the best policy performance is 0.004152, with a standard deviation of 0.002308. Thus it is possible that performance of policy type 1 on demand scenarios 3 and 4 is also close to the performance that could be obtained from the other policies with greater computational resources.

Turning now to the question of *Q*-learning performance as compared to the performance of the heuristic policy (type 0), we refer primarily to the simulation set graphs on pages 108 - 110. For variables N and τ, these graphs show the performance of the *best Q*-learned policy against the performance of the heuristic policy. The latter is always shown in red color, while the former is always shown in black. Cases with $N = 2$ are shown by dashed

lines, while cases with $N = 3$ are shown by solid lines. Thus, in order to compare *Q*-learning with the heuristic, we must compare solid line with solid line or dashed line with dashed line, not a combination of solid and dashed!

It is immediately clear from the graphs that the performance difference between all policies tends to increase with τ. Considering reward structure A to be a base case, this progressive difference is more marked in the case of reward structure C and less marked in the case of reward structure B. The performance difference between *Q*-learning and the heuristic follows the same pattern: greater in the case of reward structure C and less in the case of reward structure B. Thus we can infer that the *Q*-learned policies are better able to handle a greater level of value variance in the demand stream.

It also apparent from the graphs that in cases where the mean and variance of total demand is high (scenarios 3 and 4), the level of lead time flexibility, as instantiated by N, tends to have greater impact on system performance than the choice of lead time policy[12]. In the graphs for these scenarios, we see that the heuristic policy with $N = 3$ performs distinctly better than the *Q*-learned policy with $N = 2$ in most cases[13]. Comparing performances of *Q*-learned policies alone, we see that increasing lead time flexibility from $N = 2$ to $N = 3$ when total demand is high may augment average reward by several percentage points: for example, the benefit is 3% in the case of demand scenario 4, reward structure A, with $\tau = 0.050$.

SIMULATION SET 2

This set considers the performance of policies with respect to different levels of u, the unit cost applicable to overtime. Under the assumption that all demand scenarios and lead time configurations will react in qualitatively the same manner to variations in u, we conduct experiments for demand scenario 4 and $N = 3$ only.

[12]Of course, truly negligent policies, such as accepting all demands up to capacity, irrespective of overtime costs, could outweigh the choice of N.

[13]The only exception is demand scenario 3 with reward structure C, but even here the performance of the heuristic policy with $N = 3$ lies close to the performance of the *Q*-learned policy with $N = 2$.

Detail of the results appears in Tables 3.21a, 3.22a, and 3.23a. From these we see

- the *Q*-learned policy has better mean performance than the the heuristic policy in all cases except $u = 0.02$ and $u = 0.03$ with reward structure B,
- in no cases does the *Q*-learned policy perform statistically worse than the heuristic policy,
- the *Q*-learned policy performs statistically better than the heuristic policy for $u \geq 0.05$ with reward structure B,
- the *Q*-learned policy always performs statistically better than the heuristic policy with rewards structures A and C.

Referring to the graphs of mean performance in figures 3.9a, 3.10a, and 3.11a, we find a similar pattern across reward sets as we saw in Simulation Set 1. The relative performance of the *Q*-learned policy improves with increasing u, and this improvement increases with the value variance in the demand stream: we see greatest relative improvement with reward structure C and least relative improvement with reward structure B. Again, we can infer that the *Q*-learned policies are better able to handle variance in values across demands.

SIMULATION SET 3

Our third and final simulation set considers the performance of policies with respect to different levels of b, the nominal system capacity. Under the same assumption of qualitative similarity across demand scenarios and lead time configurations, we conduct experiments for demand scenario 4 and $N = 3$ only.

The mean demand for scenario 4 is 100 units per day. While it is perhaps immediately evident that having a nominal capacity less than mean demand is undesirable, we evaluate the case of $b = 95$, in order to get an idea of the impact of such onerous constraints.

Detail of the results appears in Tables 3.21b, 3.22b, and 3.23b. From these we see that the performance of the *Q*-learned policy is statistically superior to the performance of

the heuristic in all cases, although the performance difference is of course decreasing in b, and on the order of 0.01 or smaller for $b = 115$. Nevertheless, the same pattern across reward structures emerges: greatest benefit from *Q*-learning with reward structure C and least benefit with reward structure B.

COMPARISON WITH MARKOV CHAIN ANALYSIS

A final perspective on the results of the simulation study is provided by the Markov chain analysis of section 3.4. For all reward structures, when $\tau = 0$, we can use the two-class Markov chain implementation to search for the optimal policy and average reward of demand scenario 1. These results are given in the Table 3.7.

The rows and columns in Table 3.7 indicate respectively the threshold level π_1 for class 1 demands and π_2 for class 2 demands. The optimal average reward for each case is shown by bold type, and the reward corresponding to the policy determined by *Q*-learning is highlighted green. Any suboptimal average rewards that are higher than the *Q*-learning average reward are highlighted yellow. On the assumption that the objective function for the problem is quasi-concave, the extent of the tables is restricted to the policies bordering the yellow region (*i.e.*, we assume that no policies outside the ones shown perform as well or better than the Q-learning policy.)

The results in Table 3.7 indicate that although *Q*-learning does not provide an optimal threshold policy, nor even the next-to-optimal policy, the results are not far removed from optimality. The maximum deviation from optimality in the table is 0.009156, in the case of reward structure B. In problems with three or fewer classes, a local search of the policy space around the *Q*-learning solution may reveal the true optimum.

(a) Reward Structure A, Demand Scenario 1, $\tau = 0$

		π_2				
		6	7	8	9	10
π_1	10	0.927066	0.940882	0.941255	0.935364	0.929299
	11	0.930686	0.946589	**0.949946**	0.944399	0.938268
	12	0.925734	0.940910	0.944053	0.939182	0.933668

(b) Reward Structure B, Demand Scenario 1, $\tau = 0$

		π_2						
		6	7	8	9	10	11	12
π_1	9	0.909733	0.924767	0.932220	0.931016	0.928615	0.925301	0.922721
	10	0.927405	0.944468	0.949696	0.947494	0.943827	0.941975	0.939723
	11	0.930240	0.948721	**0.956267**	0.954330	0.950783	0.948024	0.946339
	12	0.926325	0.944247	0.951624	0.950219	0.947111	0.944762	0.943139
	13	0.921396	0.939263	0.945158	0.941930	0.937834	0.935347	0.934100

(c) Reward Structure C, Demand Scenario 1, $\tau = 0$

		π_2			
		6	7	8	9
π_1	9	0.895800	0.901411	0.897324	0.886695
	10	0.926478	0.935499	0.927554	0.914801
	11	0.931463	**0.942886**	0.938967	0.926795
	12	0.924708	0.935115	0.930903	0.919769

Table 3.7: Markov Chain Analysis of Demand Scenario 1
Optimal average reward for each case is shown by bold type; reward for the policy determined by Q -learning is highlighted green.

3.7 CONCLUSIONS

We have provided several perspectives on the lead time promising problem for a production operation facing multiple customer classes. For small systems, an optimal threshold policy for lead times can be derived through the combination of a Markov chain model and direct search of the policy space. This analysis is contingent, however, on the assumption of non-overlapping lead times. Moreover, since transition and reward matrices grow exponentially with system capacity, the approach serves only to provide benchmark results for

the performance of other techniques on small systems.

Q-learning allows us to mitigate the problem of dimensionality, but large state spaces still pose a challenge for computational resources. Also, the updating pattern of the algorithm is complicated by the presence of multiple customer classes, since directly related lead time actions are unlikely occur in direct sequence. We address both of these problems through the single device of reorienting lead time offers to focus on production days, rather than on individual demands.

Our results from Q-learning suggest that, in most cases, it offers a distinct improvement over a plausible heuristic approach to lead time promising. In particular, the relative advantage of learned policies is only small when the business conditions faced by the system are easier: more homogeneity across demand classes, less likelihood of supply chain disruptions, higher nominal production capacity, or lower overtime costs. As any one of these conditions becomes less favorable, the relative benefit from Q-learning increases.

Nevertheless, in light of the benchmark results established by our Markov chain model, we can be almost certain that our Q-learning results still fall short of optimal lead time policies. Further work is thus needed, in order to establish benchmarks for larger systems and improve the output of our algorithms. Several possibilities for the latter are readily apparent. We might employ other value function approximation techniques, such as neural network training, either as a complement or as an alternative to our day-centric formulation; adjustments to the learning rate of algorithm may prevent it "getting stuck" in a near-optimal policy; or an iterative approach, using two or more rounds of Q-learning with information from one taken as input to the next. We could also investigate solution methods other than Q-learning. In particular, since our day-centric formulation can readily be interpreted as a stochastic shortest path problem, and since we are only concerned with policies that are consistently improving (*i.e.*, we do not anticipate ever making a lead time offer that would entail a net cost if accepted), solution through a stochastic adaptation of a label correcting algorithm is a possibility (Bertsekas, 2001).

Besides the need for better algorithm performance, we have shown that inherent system flexibility is an important concern for the lead time promising problem. When variance of total demand is high, a heuristic approach that allows a greater range of lead time offers may outperform an otherwise optimal lead time policy. When variance of total demand is low, however, additional flexibility may add no benefit or even incur some net loss of revenue. Due to assumptions made for the sake of tractability, these contrasts are outlined here in broad strokes. In a more general framework, a more nuanced approach to the question of lead time flexibility may be essential. How many lead times should be available for each class? What advantage may be gained is we allow lead times to overlap across customer classes? Our Q-learning implementation is not bound by an assumption of non-overlapping lead times, although the system instances evaluated here do have that property.

Other conceptual extensions of the current research are readily apparent. We might consider the possibility of non-stationary demand processes: for example, total demand may exhibit autoregressive properties or be described by a non-homogeneous Poisson process. A non-stationary lead time policy would be needed to handle such scenarios. The business model of the firm could also be enhanced: the reward from each successful lead time offer might be a deterministic function of the length of lead time, or the probability of rejection might be reduced by offering a customer a discount along with a longer lead time offer. Everything else equal, what reward or discount structure would entail an increase in the firm's average reward?

τ	N	Policy	CI, Lower	Mean	CI, Upper
0.000	2	0	0.929488	0.929622	0.929755
		1	**0.943935**	**0.944007**	**0.944079**
		2	0.916462	0.916550	0.916639
		3	0.942180	0.942294	0.942409
	3	0	0.933000	0.933128	0.933257
		1	0.939177	0.939252	0.939327
		2	0.940531	0.940602	0.940673
		3	0.943362	0.943466	0.943569
0.025	2	0	0.894381	0.894638	0.894894
		1	**0.911871**	**0.912086**	**0.912301**
		2	0.885664	0.885886	0.886108
		3	0.908288	0.908524	0.908760
	3	0	0.897938	0.898186	0.898434
		1	0.907246	0.907460	0.907673
		2	0.900251	0.900454	0.900656
		3	0.909780	0.910001	0.910222
0.050	2	0	0.858861	0.859170	0.859480
		1	**0.879347**	**0.879612**	**0.879878**
		2	0.852056	0.852327	0.852597
		3	0.867006	0.867322	0.867638
	3	0	0.862471	0.862794	0.863117
		1	0.875050	0.875315	0.875581
		2	0.868083	0.868345	0.868607
		3	0.875819	0.876101	0.876383
0.075	2	0	0.822991	0.823348	0.823705
		1	**0.846152**	**0.846467**	**0.846782**
		2	0.826091	0.826409	0.826728
		3	0.821099	0.821479	0.821859
0.075	3	0	0.826537	0.826903	0.827270
		1	0.842371	0.842680	0.842989
		2	0.831585	0.831872	0.832160
		3	0.841468	0.841802	0.842136
0.100	2	0	0.786543	0.786969	0.787395
		1	**0.812288**	**0.812676**	**0.813065**
		2	0.788625	0.789025	0.789426
		3	0.787566	0.788005	0.788444
	3	0	0.790158	0.790589	0.791020
		1	0.809206	0.809576	0.809947
		2	0.800704	0.801057	0.801411
		3	0.806770	0.807172	0.807573
0.125	2	0	0.749467	0.749912	0.750357
		1	**0.777585**	**0.778000**	**0.778416**
		2	0.747767	0.748174	0.748581
		3	0.750607	0.751046	0.751486
	3	0	0.753191	0.753651	0.754112
		1	0.775506	0.775894	0.776283
		2	0.762218	0.762605	0.762993
		3	0.715540	0.715980	0.716421
0.150	2	0	0.711746	0.712225	0.712705
		1	**0.742044**	**0.742496**	**0.742948**
		2	0.709397	0.709856	0.710316
		3	0.712240	0.712723	0.713207
	3	0	0.715520	0.716020	0.716519
		1	0.741079	0.741506	0.741933
		2	0.726807	0.727242	0.727677
		3	0.735360	0.735819	0.736278

Table 3.8: Reward Structure A, Simulation Set 1, Demand Scenario 1

τ	N	Policy	CI, Lower	Mean	CI, Upper
0.000	2	0	0.943624	0.943730	0.943836
		1	0.946979	0.947060	0.947141
		2	0.939811	0.939877	0.939943
		3	0.953618	0.953710	0.953802
	3	0	0.946385	0.946487	0.946590
		1	0.947376	0.947454	0.947532
		2	0.948075	0.948137	0.948199
		3	**0.954553**	**0.954632**	**0.954712**
0.025	2	0	0.915711	0.915914	0.916118
		1	0.920763	0.920953	0.921143
		2	0.914264	0.914442	0.914619
		3	0.921854	0.922064	0.922274
	3	0	0.918507	0.918705	0.918903
		1	0.921206	0.921386	0.921566
		2	0.922183	0.922349	0.922515
		3	**0.923083**	**0.923280**	**0.923476**
0.050	2	0	0.887461	0.887708	0.887956
		1	0.894175	0.894401	0.894627
		2	**0.899106**	**0.899314**	**0.899523**
		3	0.893922	0.894173	0.894425
	3	0	0.890290	0.890548	0.890805
		1	0.894646	0.894871	0.895096
		2	0.895723	0.895930	0.896136
		3	0.883760	0.884024	0.884287
0.075	2	0	0.858915	0.859200	0.859485
		1	0.867186	0.867457	0.867728
		2	0.868056	0.868306	0.868556
		3	0.857098	0.857404	0.857711
0.075	3	0	0.861703	0.861995	0.862288
		1	0.867708	0.867973	0.868239
		2	0.869664	0.869917	0.870169
		3	**0.873426**	**0.873693**	**0.873961**
0.100	2	0	0.829904	0.830244	0.830583
		1	0.839728	0.840044	0.840359
		2	0.845442	0.845751	0.846060
		3	0.820822	0.821187	0.821553
	3	0	0.832742	0.833085	0.833427
		1	0.840281	0.840592	0.840904
		2	0.841200	0.841498	0.841796
		3	**0.845789**	**0.846113**	**0.846437**
0.125	2	0	0.800396	0.800751	0.801106
		1	0.811708	0.812036	0.812365
		2	0.810128	0.810455	0.810781
		3	0.801401	0.801756	0.802111
	3	0	0.803312	0.803677	0.804043
		1	0.810448	0.810779	0.811110
		2	**0.815657**	**0.815969**	**0.816281**
		3	0.729941	0.730297	0.730652
0.150	2	0	0.770364	0.770744	0.771124
		1	0.783074	0.783429	0.783783
		2	0.778292	0.778649	0.779005
		3	0.769520	0.769901	0.770283
	3	0	0.773309	0.773704	0.774099
		1	0.786440	0.786785	0.787129
		2	0.787959	0.788302	0.788646
		3	**0.788933**	**0.789289**	**0.789646**

Table 3.9: Reward Structure B, Simulation Set 1, Demand Scenario 1

τ	N	Policy	CI, Lower	Mean	CI, Upper
0.000	2	0	0.904933	0.905117	0.905301
		1	0.934881	0.934983	0.935086
		2	0.902344	0.902456	0.902567
		3	0.928858	0.929000	0.929141
	3	0	0.909751	0.909925	0.910100
		1	**0.936968**	**0.937055**	**0.937141**
		2	0.902142	0.902232	0.902321
		3	0.923931	0.924073	0.924216
0.025	2	0	0.857333	0.857683	0.858033
		1	0.890923	0.891216	0.891509
		2	0.855947	0.856253	0.856558
		3	0.876369	0.876692	0.877015
	3	0	0.862210	0.862548	0.862885
		1	**0.894457**	**0.894737**	**0.895017**
		2	0.862011	0.862287	0.862563
		3	0.831562	0.831866	0.832170
0.050	2	0	0.809183	0.809603	0.810024
		1	0.846255	0.846621	0.846988
		2	0.810169	0.810559	0.810950
		3	0.820268	0.820704	0.821139
	3	0	0.814149	0.814589	0.815028
		1	**0.851607**	**0.851962**	**0.852317**
		2	0.819202	0.819551	0.819899
		3	0.832616	0.833005	0.833395
0.075	2	0	0.760594	0.761077	0.761561
		1	0.800640	0.801073	0.801506
		2	0.764950	0.765418	0.765885
		3	0.757793	0.758301	0.758809
0.075	3	0	0.765455	0.765953	0.766451
		1	**0.808147**	**0.808539**	**0.808931**
		2	0.773873	0.774263	0.774652
		3	0.785848	0.786310	0.786771
0.100	2	0	0.711227	0.711806	0.712386
		1	0.754039	0.754568	0.755096
		2	0.716159	0.716711	0.717264
		3	0.719268	0.719853	0.720438
	3	0	0.716191	0.716780	0.717368
		1	**0.763858**	**0.764346**	**0.764835**
		2	0.734357	0.734851	0.735345
		3	0.739054	0.739573	0.740091
0.125	2	0	0.661005	0.661611	0.662216
		1	0.706341	0.706917	0.707494
		2	0.666612	0.667190	0.667767
		3	0.669733	0.670325	0.670916
	3	0	0.666132	0.666763	0.667393
		1	**0.718880**	**0.719398**	**0.719916**
		2	0.687497	0.688033	0.688569
		3	0.618706	0.619311	0.619916
0.150	2	0	0.609929	0.610586	0.611242
		1	0.657437	0.658060	0.658683
		2	0.614890	0.615540	0.616189
		3	0.613247	0.613909	0.614572
	3	0	0.615144	0.615829	0.616514
		1	**0.674136**	**0.674722**	**0.675307**
		2	0.642736	0.643317	0.643899
		3	0.642561	0.643185	0.643808

Table 3.10: Reward Structure C, Simulation Set 1, Demand Scenario 1

τ	N	Policy	CI, Lower	Mean	CI, Upper
0.000	2	0	0.955650	0.955780	0.955909
		1	0.964852	0.964937	0.965023
		2	0.965264	0.965359	0.965455
		3	0.963044	0.963150	0.963255
	3	0	0.962283	0.962391	0.962499
		1	0.966668	0.966740	0.966813
		2	**0.969637**	**0.969700**	**0.969762**
		3	0.943324	0.943417	0.943509
0.025	2	0	0.916432	0.916737	0.917043
		1	0.934178	0.934416	0.934653
		2	0.927205	0.927476	0.927748
		3	0.916674	0.917004	0.917334
	3	0	0.925815	0.926101	0.926387
		1	0.934796	0.935030	0.935263
		2	**0.936395**	**0.936625**	**0.936856**
		3	0.925427	0.925677	0.925928
0.050	2	0	0.876729	0.877085	0.877441
		1	0.904509	0.904783	0.905058
		2	0.892108	0.892431	0.892754
		3	0.869987	0.870371	0.870755
	3	0	0.888363	0.888706	0.889048
		1	0.903972	0.904256	0.904539
		2	**0.906206**	**0.906474**	**0.906743**
		3	0.883853	0.884165	0.884477
0.075	**2**	0	0.836216	0.836625	0.837033
		1	**0.870920**	**0.871248**	**0.871575**
		2	0.869291	0.869621	0.869950
		3	0.821811	0.822270	0.822729
0.075	3	0	0.849881	0.850273	0.850664
		1	0.869055	0.869375	0.869695
		2	0.871126	0.871446	0.871767
		3	0.830492	0.830829	0.831166
0.100	2	0	0.795018	0.795540	0.796062
		1	0.836257	0.836697	0.837137
		2	0.813607	0.814106	0.814605
		3	0.773012	0.773589	0.774166
	3	0	0.810355	0.810848	0.811341
		1	0.837112	0.837526	0.837940
		2	**0.838289**	**0.838702**	**0.839116**
		3	0.822150	0.822562	0.822974
0.125	**2**	0	0.753333	0.753847	0.754362
		1	**0.803994**	**0.804446**	**0.804897**
		2	0.777143	0.777636	0.778129
		3	0.725621	0.726197	0.726773
	3	0	0.769999	0.770507	0.771015
		1	0.798794	0.799241	0.799688
		2	0.799647	0.800086	0.800525
		3	0.784752	0.785200	0.785648
0.150	**2**	0	0.710796	0.711335	0.711875
		1	**0.764766**	**0.765280**	**0.765795**
		2	0.733517	0.734041	0.734565
		3	0.676303	0.676916	0.677528
	3	0	0.728538	0.729094	0.729649
		1	0.762853	0.763347	0.763841
		2	0.762606	0.763100	0.763594
		3	0.727571	0.728077	0.728583

Table 3.11: Reward Structure A, Simulation Set 1, Demand Scenario 2

τ	N	Policy	CI, Lower	Mean	CI, Upper
0.000	2	0	0.965032	0.965134	0.965236
		1	0.969627	0.969707	0.969788
		2	0.972466	0.972542	0.972618
		3	0.970208	0.970296	0.970384
	3	0	0.970258	0.970343	0.970428
		1	0.972909	0.972967	0.973026
		2	**0.973485**	**0.973536**	**0.973588**
		3	0.941247	0.941298	0.941349
0.025	2	0	0.933935	0.934177	0.934420
		1	0.942142	0.942357	0.942572
		2	0.941187	0.941401	0.941614
		3	0.934146	0.934406	0.934666
	3	0	0.941327	0.941554	0.941781
		1	0.946273	0.946477	0.946680
		2	**0.947049**	**0.947240**	**0.947431**
		3	0.937015	0.937217	0.937420
0.050	2	0	0.902441	0.902723	0.903005
		1	0.914800	0.915049	0.915297
		2	0.912096	0.912358	0.912620
		3	0.897234	0.897542	0.897850
	3	0	0.911607	0.911879	0.912151
		1	0.918171	0.918416	0.918661
		2	**0.922437**	**0.922661**	**0.922886**
		3	0.907153	0.907388	0.907624
0.075	2	0	0.870297	0.870621	0.870945
		1	0.887895	0.888176	0.888457
		2	0.882800	0.883112	0.883424
		3	0.859195	0.859560	0.859926
0.075	3	0	0.881063	0.881374	0.881684
		1	0.890441	0.890722	0.891003
		2	**0.893942**	**0.894200**	**0.894457**
		3	0.887095	0.887368	0.887641
0.100	2	0	0.837594	0.838008	0.838422
		1	0.858518	0.858895	0.859272
		2	0.850518	0.850922	0.851327
		3	0.820008	0.820466	0.820925
	3	0	0.849680	0.850071	0.850462
		1	0.860986	0.861349	0.861711
		2	**0.867273**	**0.867606**	**0.867939**
		3	0.858495	0.858832	0.859169
0.125	2	0	0.804493	0.804902	0.805311
		1	0.831596	0.831968	0.832340
		2	0.817488	0.817887	0.818286
		3	0.782366	0.782827	0.783288
	3	0	0.817622	0.818025	0.818428
		1	0.830932	0.831318	0.831703
		2	**0.837220**	**0.837580**	**0.837939**
		3	0.821676	0.822042	0.822407
0.150	2	0	0.770704	0.771133	0.771562
		1	0.796471	0.796886	0.797301
		2	0.785341	0.785759	0.786177
		3	0.743460	0.743941	0.744422
	3	0	0.784679	0.785120	0.785562
		1	0.800925	0.801343	0.801762
		2	**0.806757**	**0.807160**	**0.807564**
		3	0.796496	0.796905	0.797313

Table 3.12: Reward Structure B, Simulation Set 1, Demand Scenario 2

τ	N	Policy	CI, Lower	Mean	CI, Upper
0.000	2	0	0.939355	0.939532	0.939709
		1	0.962173	0.962278	0.962382
		2	0.955485	0.955599	0.955713
		3	0.950505	0.950643	0.950782
	3	0	0.948432	0.948579	0.948726
		1	0.957330	0.957417	0.957504
		2	**0.963174**	**0.963257**	**0.963341**
		3	0.923921	0.923979	0.924037
0.025	2	0	0.886031	0.886446	0.886861
		1	**0.925871**	**0.926167**	**0.926463**
		2	0.908942	0.909292	0.909642
		3	0.846373	0.846816	0.847258
	3	0	0.898871	0.899260	0.899648
		1	0.920420	0.920701	0.920982
		2	0.921040	0.921331	0.921621
		3	0.880157	0.880461	0.880766
0.050	2	0	0.921040	0.832554	0.833039
		1	**0.883331**	**0.883683**	**0.884035**
		2	0.876607	0.876977	0.877348
		3	0.788616	0.789125	0.789634
	3	0	0.847990	0.848456	0.848922
		1	0.880677	0.881020	0.881363
		2	0.878140	0.878495	0.878850
		3	0.834595	0.834987	0.835379
0.075	2	0	0.777022	0.777578	0.778133
		1	**0.840701**	**0.841128**	**0.841554**
		2	0.838128	0.838567	0.839006
		3	0.748146	0.748750	0.749354
0.075	3	0	0.795721	0.796253	0.796785
		1	0.830344	0.830771	0.831198
		2	0.827059	0.827499	0.827939
		3	0.787297	0.787742	0.788187
0.100	2	0	0.721066	0.721776	0.722487
		1	0.795144	0.795713	0.796281
		2	0.790447	0.791034	0.791620
		3	0.687710	0.688449	0.689188
	3	0	0.742051	0.742722	0.743392
		1	**0.796641**	**0.797155**	**0.797669**
		2	0.788622	0.789165	0.789709
		3	0.726139	0.726772	0.727404
0.125	2	0	0.664472	0.665170	0.665869
		1	0.747554	0.748142	0.748730
		2	**0.750926**	**0.751515**	**0.752103**
		3	0.627233	0.628029	0.628825
	3	0	0.687280	0.687972	0.688663
		1	0.739243	0.739822	0.740401
		2	0.735020	0.735607	0.736193
		3	0.672187	0.672822	0.673456
0.150	2	0	0.606743	0.607473	0.608203
		1	**0.694892**	**0.695586**	**0.696279**
		2	0.657074	0.657761	0.658448
		3	0.563266	0.564083	0.564901
	3	0	0.631029	0.631781	0.632533
		1	0.691522	0.692168	0.692815
		2	0.686400	0.687040	0.687681
		3	0.625033	0.625738	0.626444

Table 3.13: Reward Structure C, Simulation Set 1, Demand Scenario 2

τ	N	Policy	CI, Lower	Mean	CI, Upper
0.000	2	0	0.971397	0.971492	0.971588
		1	0.975360	0.975443	0.975527
		2	0.978082	0.978132	0.978182
		3	0.972315	0.972367	0.972419
	3	0	0.976579	0.976648	0.976718
		1	**0.978639**	**0.978691**	**0.978743**
0.025	2	0	0.916342	0.916723	0.917104
		1	0.927860	0.928193	0.928526
		2	0.927967	0.928314	0.928662
		3	0.922590	0.922944	0.923297
	3	0	0.934046	0.934351	0.934657
		1	**0.939666**	**0.939945**	**0.940224**
0.050	2	0	0.859319	0.859815	0.860311
		1	0.877218	0.877667	0.878116
		2	0.876307	0.876761	0.877215
		3	0.876106	0.876531	0.876955
	3	0	0.887484	0.887887	0.888291
		1	**0.891891**	**0.892270**	**0.892649**
0.075	2	0	0.800267	0.800848	0.801430
		1	0.821672	0.822208	0.822745
		2	0.825564	0.826105	0.826646
		3	0.826795	0.827298	0.827800
	3	0	0.837132	0.837649	0.838167
		1	**0.857087**	**0.857532**	**0.857978**
0.100	2	0	0.739327	0.740073	0.740818
		1	0.766031	0.766715	0.767400
		2	0.766512	0.767211	0.767911
		3	0.775191	0.775842	0.776493
	3	0	0.783653	0.784332	0.785012
		1	**0.813199**	**0.813765**	**0.814331**
0.125	2	0	0.676999	0.677800	0.678601
		1	0.715122	0.715855	0.716589
		2	0.699679	0.700438	0.701197
		3	0.675081	0.675853	0.676625
	3	0	0.727634	0.728354	0.729073
		1	**0.759885**	**0.760522**	**0.761159**
0.150	2	0	0.613222	0.614059	0.614896
		1	0.649114	0.649901	0.650688
		2	0.638253	0.639072	0.639892
		3	0.656132	0.656877	0.657621
	3	0	0.668977	0.669771	0.670566
		1	**0.694843**	**0.695617**	**0.696391**

Table 3.14: Reward Structure A, Simulation Set 1, Demand Scenario 3

τ	N	*Policy*	*CI, Lower*	*Mean*	*CI, Upper*
0.000	2	0	0.977464	0.977539	0.977614
		1	0.980469	0.980537	0.980605
		2	0.979812	0.979852	0.979892
		3	0.969853	0.969916	0.969979
	3	0	0.981547	0.981601	0.981656
		1	**0.983177**	**0.983218**	**0.983259**
0.025	2	0	0.933903	0.934205	0.934506
		1	0.941735	0.942011	0.942287
		2	0.942625	0.942900	0.943175
		3	0.937556	0.937844	0.938132
	3	0	0.947852	0.948093	0.948335
		1	**0.952742**	**0.952957**	**0.953172**
0.050	2	0	0.888779	0.889171	0.889563
		1	0.901508	0.901863	0.902219
		2	0.902336	0.902691	0.903046
		3	0.888017	0.888400	0.888782
	3	0	0.910969	0.911289	0.911608
		1	**0.912554**	**0.912859**	**0.913163**
0.075	2	0	0.842040	0.842501	0.842961
		1	0.858760	0.859187	0.859615
		2	0.860708	0.861142	0.861577
		3	0.839689	0.840138	0.840587
	3	0	0.871086	0.871496	0.871905
		1	**0.880336**	**0.880722**	**0.881108**

τ	N	*Policy*	*CI, Lower*	*Mean*	*CI, Upper*
0.100	2	0	0.793800	0.794389	0.794979
		1	0.816422	0.816962	0.817502
		2	0.815819	0.816369	0.816919
		3	0.795601	0.796177	0.796752
	3	0	0.828723	0.829261	0.829799
		1	**0.849549**	**0.850010**	**0.850472**
0.125	2	0	0.744448	0.745082	0.745716
		1	0.767941	0.768537	0.769133
		2	0.761706	0.762312	0.762919
		3	0.742815	0.743429	0.744044
	3	0	0.784343	0.784913	0.785483
		1	**0.801683**	**0.802209**	**0.802735**
0.150	2	0	0.693938	0.694602	0.695265
		1	0.713855	0.714504	0.715153
		2	0.713305	0.713947	0.714590
		3	0.685398	0.686054	0.686711
	3	0	0.737867	0.738497	0.739128
		1	**0.750367**	**0.750988**	**0.751608**

Table 3.15: Reward Structure B, Simulation Set 1, Demand Scenario 3

τ	N	*Policy*	*CI, Lower*	*Mean*	*CI, Upper*
0.000	2	0	0.960859	0.960990	0.961121
		1	0.961533	0.961671	0.961809
		2	**0.975976**	**0.976043**	**0.976109**
		3	0.970317	0.970389	0.970461
	3	0	0.967950	0.968045	0.968141
		1	0.970837	0.970908	0.970978
0.025	2	0	0.885841	0.886361	0.886880
		1	0.903804	0.904224	0.904644
		2	0.902382	0.902851	0.903320
		3	0.911772	0.912178	0.912584
	3	0	0.910066	0.910483	0.910899
		1	**0.922345**	**0.922690**	**0.923035**
0.050	2	0	0.808151	0.808827	0.809504
		1	0.847112	0.847647	0.848182
		2	0.833416	0.834024	0.834633
		3	0.855270	0.855786	0.856302
	3	0	0.846694	0.847242	0.847791
		1	**0.865397**	**0.865866**	**0.866335**
0.075	2	0	0.727714	0.728505	0.729295
		1	0.765464	0.766172	0.766881
		2	0.761663	0.762393	0.763122
		3	0.788170	0.788820	0.789471
	3	0	0.778159	0.778863	0.779568
		1	**0.816124**	**0.816683**	**0.817242**

τ	N	*Policy*	*CI, Lower*	*Mean*	*CI, Upper*
0.100	2	0	0.644716	0.645733	0.646750
		1	0.687288	0.688211	0.689134
		2	0.684848	0.685797	0.686746
		3	0.713472	0.714306	0.715140
	3	0	0.705372	0.706297	0.707223
		1	**0.756038**	**0.756774**	**0.757509**
0.125	2	0	0.559850	0.560942	0.562033
		1	0.617279	0.618238	0.619198
		2	0.591412	0.592453	0.593493
		3	0.654122	0.655033	0.655945
	3	0	0.629139	0.630119	0.631098
		1	**0.693750**	**0.694529**	**0.695308**
0.150	2	0	0.473030	0.474168	0.475306
		1	0.539337	0.540375	0.541413
		2	0.512885	0.513996	0.515107
		3	0.553345	0.554302	0.555260
	3	0	0.549324	0.550404	0.551485
		1	**0.619905**	**0.620829**	**0.621753**

Table 3.16: Reward Structure C, Simulation Set 1, Demand Scenario 3

τ	N	Policy	CI, Lower	Mean	CI, Upper
0.000	**2**	0	0.957817	0.957932	0.958046
		1	**0.972741**	**0.972823**	**0.972905**
	3	0	0.969164	0.969260	0.969355
		1	0.971267	0.971346	0.971424
0.025	2	0	0.901125	0.901517	0.901908
		1	0.919454	0.919820	0.920186
	3	0	0.924248	0.924561	0.924874
		1	**0.931389**	**0.931670**	**0.931952**
0.050	2	0	0.842407	0.842894	0.843380
		1	0.864963	0.865419	0.865876
	3	0	0.876516	0.876912	0.877307
		1	**0.891298**	**0.891660**	**0.892022**
0.075	2	0	0.781392	0.781971	0.782550
		1	0.809673	0.810228	0.810782
	3	0	0.825698	0.826196	0.826693
		1	**0.845756**	**0.846200**	**0.846644**

τ	N	Policy	CI, Lower	Mean	CI, Upper
0.100	2	0	0.718336	0.719079	0.719822
		1	0.749065	0.749789	0.750513
	3	0	0.772093	0.772722	0.773351
		1	**0.794852**	**0.795447**	**0.796042**
0.125	2	0	0.653339	0.654141	0.654943
		1	0.685907	0.686687	0.687467
	3	0	0.715705	0.716401	0.717098
		1	**0.741570**	**0.742214**	**0.742858**
0.150	2	0	0.586386	0.587253	0.588121
		1	0.620334	0.621205	0.622075
	3	0	0.656363	0.657154	0.657945
		1	**0.685854**	**0.686631**	**0.687408**

Table 3.17: Reward Structure A, Simulation Set 1, Demand Scenario 4

τ	N	*Policy*	*CI, Lower*	*Mean*	*CI, Upper*
0.000	2	0	0.968371	0.968457	0.968543
		1	0.974934	0.975003	0.975073
	3	0	0.976826	0.976897	0.976969
		1	**0.977755**	**0.977818**	**0.977882**
0.025	2	0	0.925736	0.926030	0.926324
		1	0.937316	0.937594	0.937872
	3	0	0.942837	0.943073	0.943309
		1	**0.946355**	**0.946576**	**0.946797**
0.050	2	0	0.881580	0.881945	0.882311
		1	0.895810	0.896155	0.896500
	3	0	0.906731	0.907030	0.907329
		1	**0.911415**	**0.911695**	**0.911975**
0.075	2	0	0.835695	0.836130	0.836566
		1	0.853673	0.854094	0.854515
	3	0	0.868305	0.868681	0.869057
		1	**0.873785**	**0.874138**	**0.874492**

τ	N	*Policy*	*CI, Lower*	*Mean*	*CI, Upper*
0.100	2	0	0.788275	0.788834	0.789393
		1	0.807624	0.808169	0.808714
	3	0	0.827782	0.828258	0.828733
		1	**0.834798**	**0.835264**	**0.835731**
0.125	2	0	0.739389	0.739992	0.740596
		1	0.759526	0.760120	0.760714
	3	0	0.785164	0.785690	0.786217
		1	**0.797871**	**0.798375**	**0.798879**
0.150	2	0	0.689020	0.689673	0.690326
		1	0.712064	0.712726	0.713387
	3	0	0.740319	0.740917	0.741515
		1	**0.756298**	**0.756893**	**0.757489**

Table 3.18: Reward Structure B, Simulation Set 1, Demand Scenario 4

τ	N	*Policy*	*CI, Lower*	*Mean*	*CI, Upper*
0.000	2	0	0.933181	0.933363	0.933545
		1	0.962767	0.962901	0.963034
	3	0	0.951346	0.951497	0.951648
		1	**0.963814**	**0.963903**	**0.963992**
0.025	2	0	0.843564	0.844183	0.844803
		1	0.882764	0.883318	0.883873
	3	0	0.881039	0.881529	0.882020
		1	**0.904524**	**0.904944**	**0.905363**
0.050	2	0	0.750724	0.751495	0.752265
		1	0.797396	0.798102	0.798809
	3	0	0.806271	0.806890	0.807510
		1	**0.843515**	**0.844045**	**0.844576**
0.075	2	0	0.654245	0.655160	0.656075
		1	0.706907	0.707773	0.708640
	3	0	0.726616	0.727396	0.728176
		1	**0.771204**	**0.771883**	**0.772561**
0.100	2	0	0.554529	0.555705	0.556880
		1	0.612123	0.613250	0.614377
	3	0	0.642545	0.643532	0.644518
		1	**0.694206**	**0.695123**	**0.696040**
0.125	2	0	0.451755	0.453023	0.454290
		1	0.521857	0.523058	0.524259
	3	0	0.554077	0.555170	0.556262
		1	**0.624853**	**0.625830**	**0.626807**
0.150	2	0	0.345915	0.347286	0.348657
		1	0.421003	0.422352	0.423701
	3	0	0.460944	0.462185	0.463426
		1	**0.533438**	**0.534640**	**0.535842**

Table 3.19: Reward Structure C, Simulation Set 1, Demand Scenario 4

Table 3.20: Reward Structure A, Simulation Sets 2 and 3

(a) Simulation Set 2, Demand Scenario 4.

	Policy 0			*Policy 1*		
u	***CI, Lower***	***Mean***	***CI, Upper***	***CI, Lower***	***Mean***	***CI, Upper***
0.00	0.893571	0.893858	0.894146	0.895596	0.895869	0.896143
0.01	0.882111	0.882416	0.882722	0.883687	0.883985	0.884282
0.02	0.871240	0.871569	0.871898	0.871901	0.872219	0.872538
0.03	0.863467	0.863810	0.864152	0.864237	0.864569	0.864901
0.04	0.857747	0.858107	0.858467	0.859287	0.859628	0.859970
0.05	0.853208	0.853580	0.853952	0.854323	0.854675	0.855027
0.10	0.828455	0.828891	0.829326	0.837263	0.837678	0.838094
0.15	0.809668	0.810168	0.810667	0.819990	0.820468	0.820946
0.20	0.790881	0.791445	0.792008	0.806245	0.806784	0.807324
0.25	0.772093	0.772722	0.773351	0.794852	0.795447	0.796042
0.30	0.753304	0.753999	0.754694	0.783062	0.783704	0.784346
0.35	0.734515	0.735276	0.736037	0.771742	0.772442	0.773142
0.40	0.715725	0.716553	0.717381	0.759479	0.760244	0.761009

(b) Simulation Set 3, Demand Scenario 4.

	Policy 0			*Policy 1*		
b	***CI, Lower***	***Mean***	***CI, Upper***	***CI, Lower***	***Mean***	***CI, Upper***
95	0.673485	0.674256	0.675028	0.707862	0.708600	0.709338
100	0.772093	0.772722	0.773351	0.794852	0.795447	0.796042
105	0.826735	0.827237	0.827739	0.843454	0.843918	0.844383
110	0.851576	0.852010	0.852443	0.859148	0.859550	0.859952
115	0.861917	0.862319	0.862721	0.864780	0.865164	0.865549

Table 3.21: Reward Structure B, Simulation Sets 2 and 3

(a) Simulation Set 2, Demand Scenario 4.

	Policy 0			*Policy 1*		
u	***CI, Lower***	***Mean***	***CI, Upper***	***CI, Lower***	***Mean***	***CI, Upper***
0.00	0.920174	0.920390	0.920606	0.921345	0.921552	0.921760
0.01	0.910827	0.911059	0.911291	0.912128	0.912353	0.912577
0.02	0.903494	0.903739	0.903985	0.903441	0.903682	0.903923
0.03	0.896456	0.896718	0.896979	0.896412	0.896666	0.896921
0.04	0.892119	0.892391	0.892663	0.892246	0.892511	0.892776
0.05	0.888501	0.888784	0.889067	0.890440	0.890713	0.890985
0.10	0.869791	0.870123	0.870454	0.875077	0.875394	0.875711
0.15	0.855789	0.856168	0.856546	0.862774	0.863139	0.863504
0.20	0.841786	0.842213	0.842640	0.847470	0.847892	0.848314
0.25	0.827782	0.828258	0.828733	0.834798	0.835264	0.835731
0.30	0.813778	0.814303	0.814827	0.826134	0.826648	0.827162
0.35	0.799774	0.800348	0.800922	0.818012	0.818552	0.819092
0.40	0.785770	0.786393	0.787016	0.809542	0.810129	0.810716

(b) Simulation Set 3, Demand Scenario 4.

	Policy 0			*Policy 1*		
b	***CI, Lower***	***Mean***	***CI, Upper***	***CI, Lower***	***Mean***	***CI, Upper***
95	0.753137	0.753720	0.754303	0.770055	0.770620	0.771184
100	0.827782	0.828258	0.828733	0.834798	0.835264	0.835731
105	0.869152	0.869532	0.869912	0.877219	0.877579	0.877940
110	0.888059	0.888387	0.888715	0.890782	0.891097	0.891413
115	0.896029	0.896333	0.896636	0.897551	0.897845	0.898139

Table 3.22: Reward Structure C, Simulation Sets 2 and 3

(a) Simulation Set 2, Demand Scenario 4.

	Policy 0			*Policy 1*		
u	***CI, Lower***	***Mean***	***CI, Upper***	***CI, Lower***	***Mean***	***CI, Upper***
0.00	0.831205	0.831660	0.832114	0.835695	0.836123	0.836551
0.01	0.815824	0.816299	0.816773	0.817166	0.817628	0.818090
0.02	0.799511	0.800023	0.800534	0.805335	0.805818	0.806302
0.03	0.787645	0.788179	0.788712	0.793549	0.794055	0.794562
0.04	0.779087	0.779643	0.780199	0.787189	0.787709	0.788230
0.05	0.771993	0.772569	0.773146	0.778674	0.779209	0.779745
0.10	0.732725	0.733402	0.734080	0.755826	0.756447	0.757067
0.15	0.702666	0.703445	0.704225	0.735838	0.736556	0.737274
0.20	0.672606	0.673489	0.674371	0.717459	0.718272	0.719086
0.25	0.642545	0.643532	0.644518	0.694206	0.695123	0.696040
0.30	0.612482	0.613575	0.614667	0.681516	0.682517	0.683518
0.35	0.582419	0.583618	0.584816	0.663315	0.664420	0.665525
0.40	0.552356	0.553661	0.554965	0.639706	0.640932	0.642158

(b) Simulation Set 3, Demand Scenario 4.

	Policy 0			*Policy 1*		
b	***CI, Lower***	***Mean***	***CI, Upper***	***CI, Lower***	***Mean***	***CI, Upper***
95	0.488586	0.489795	0.491005	0.565180	0.566321	0.567461
100	0.642545	0.643532	0.644518	0.694206	0.695123	0.696040
105	0.727897	0.728683	0.729469	0.764116	0.764821	0.765527
110	0.766414	0.767094	0.767774	0.785536	0.786147	0.786758
115	0.782159	0.782792	0.783424	0.792942	0.793526	0.794110

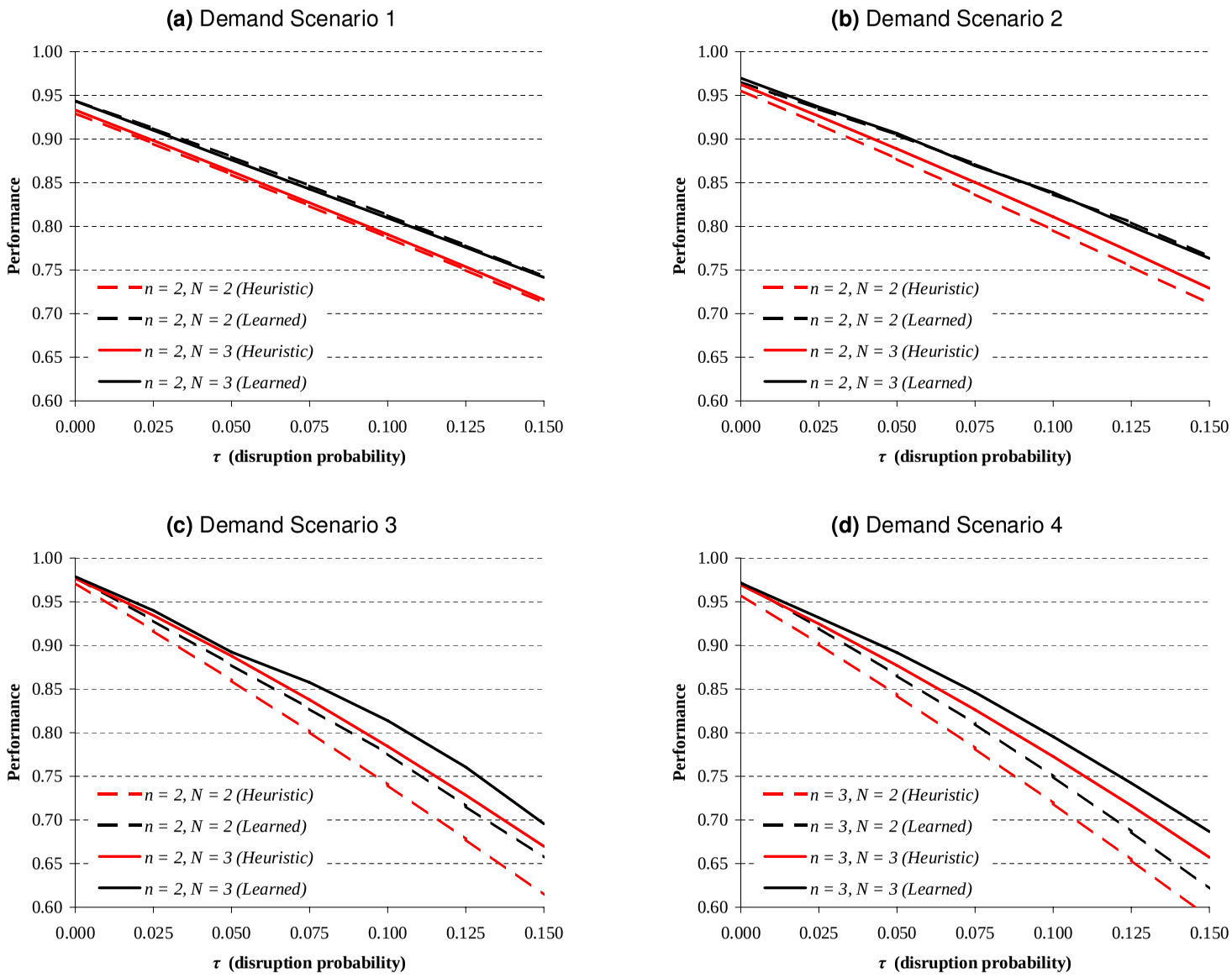


Figure 3.6: Reward Structure A, Simulation Set 1

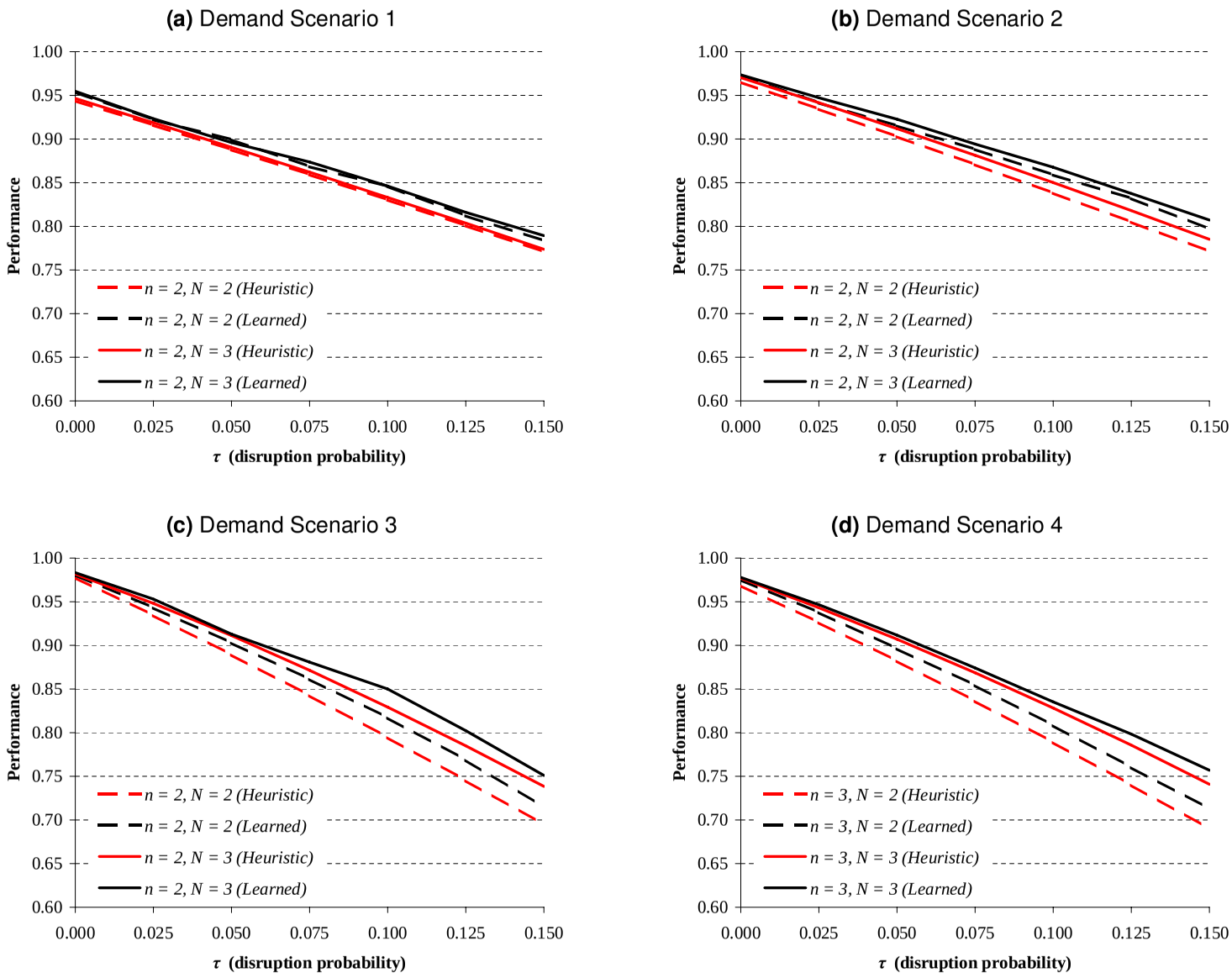


Figure 3.7: Reward Structure B, Simulation Set 1

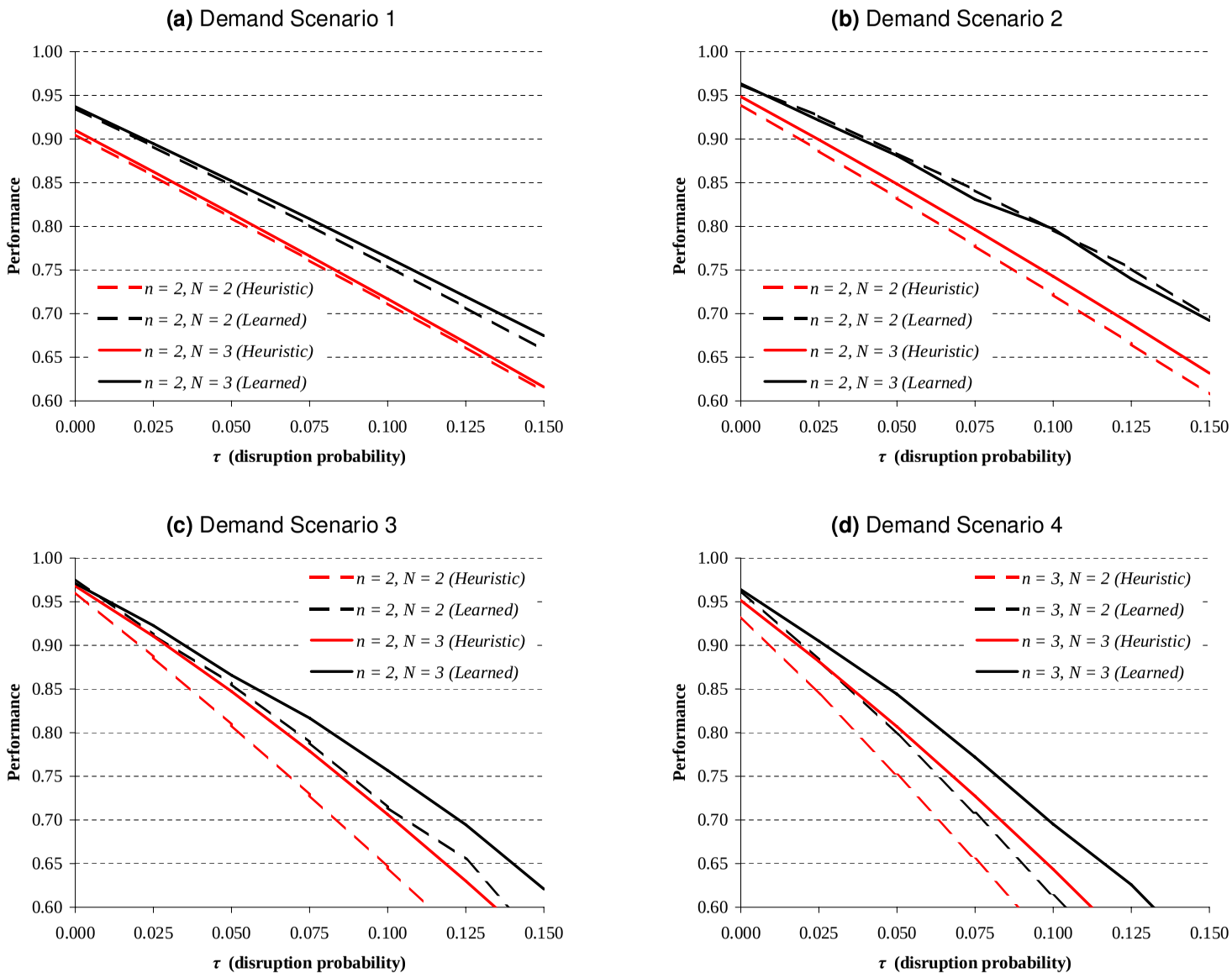


Figure 3.8: Reward Structure C, Simulation Set 1

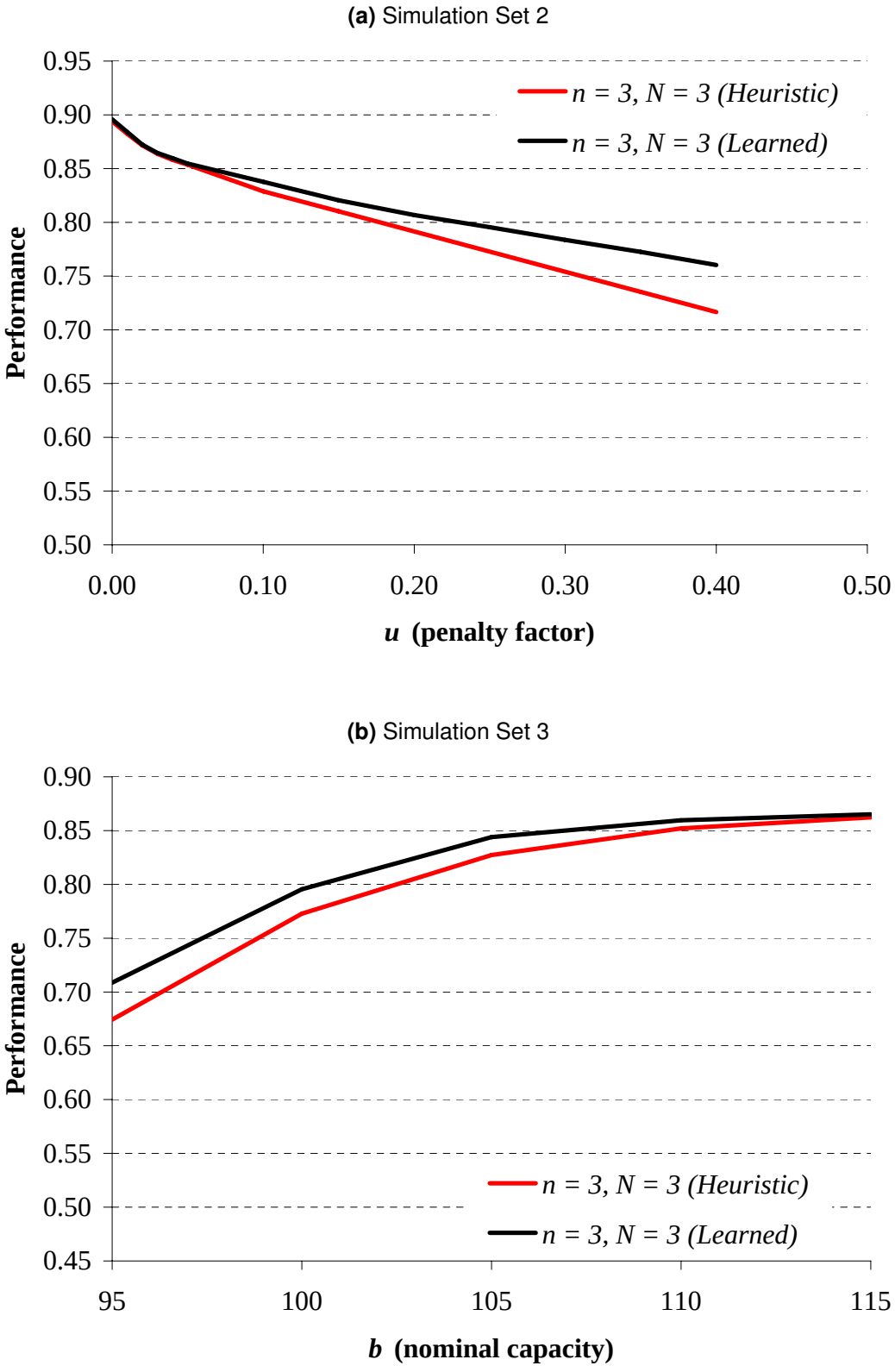


Figure 3.9: Reward Structure A, Simulation Sets 2 & 3

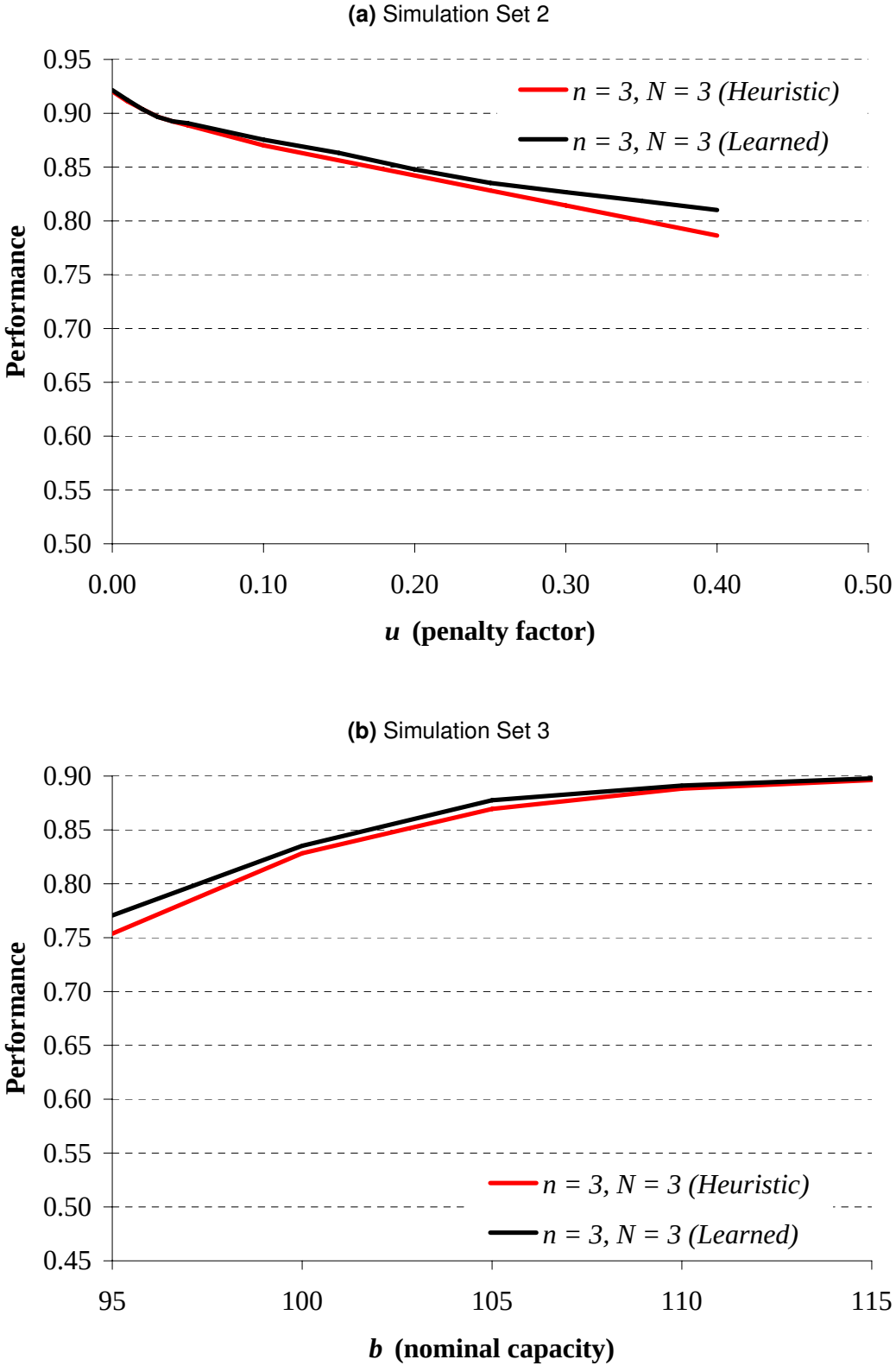


Figure 3.10: Reward Structure B, Simulation Sets 2 & 3

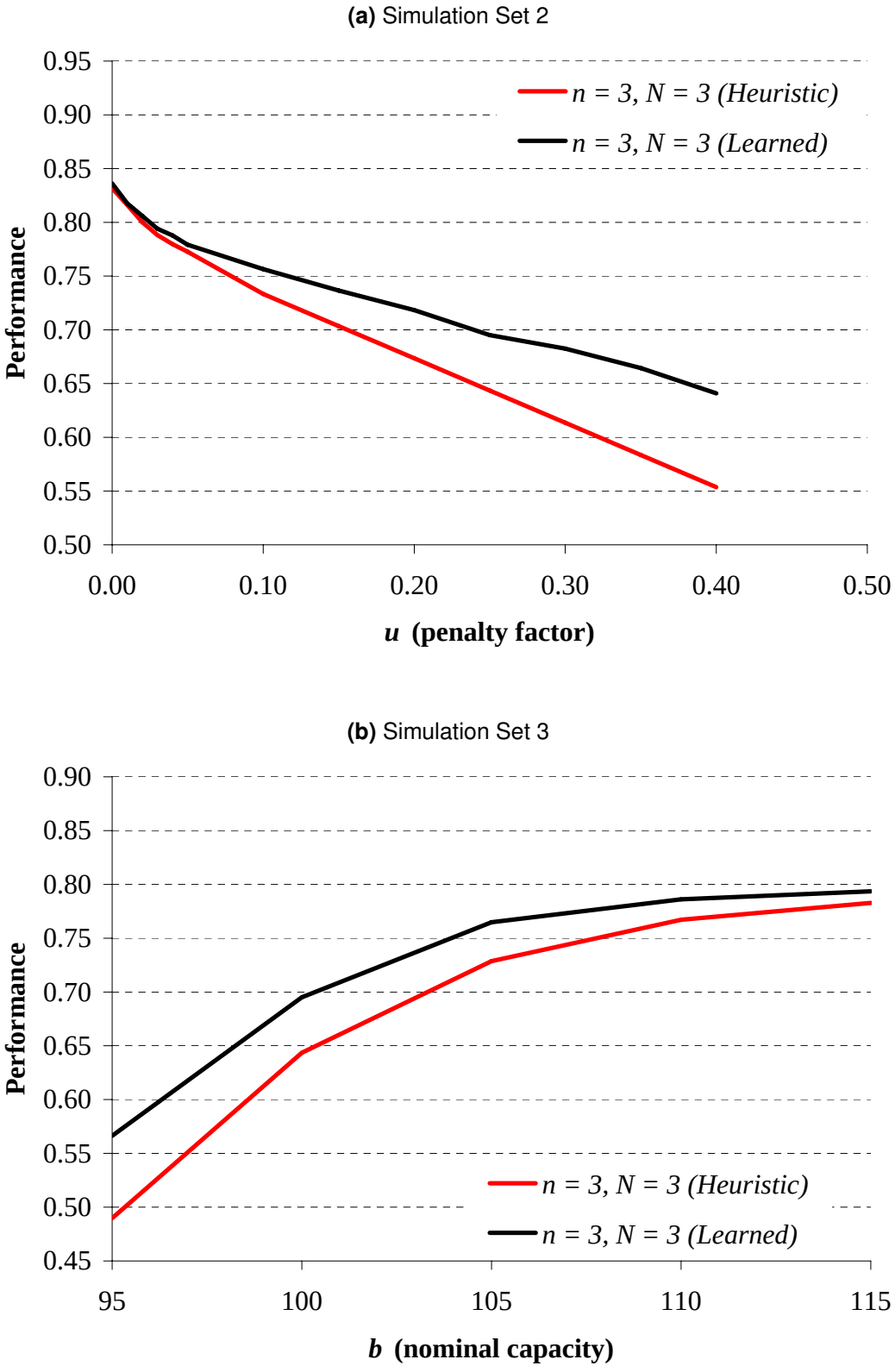


Figure 3.11: Reward Structure C, Simulation Sets 2 & 3

CHAPTER 4

AUTOMOTIVE LAUNCH PRODUCTIVITY

4.1 MOTIVATION

The transition from development and testing to full-scale manufacturing is a critical phase in the life of any new product. Even if initial market demand is strong, the firm may be hard pressed to meet output targets, due to problems resulting from the "ramp up" of production volume (Terwiesch and Bohn, 2001). Typical sources of difficulty include previously unseen flaws in the specification of production processes, breakdowns of machinery, or even aspects of the product design.

The potential impact of manufacturing hitches during launch is particularly serious for automotive companies: a new car or truck may represent the culmination of years of investment in design and engineering. Considering also the need to prepare a plant for the

new vehicle, train workers, and coordinate with component suppliers, it is clear that a company's operating costs may be significantly affected by the product launch process. Indeed, an article appearing in Wards Automotive remarks that "whenever an auto maker launches major products, there is a fear that the launch will drain productivity due to plant downtime, which inevitably drags down productivity scores. There is the increased time needed to build a newly installed vehicle at a plant, as well" (Stoll, 2004). Automotive companies must therefore be concerned to avoid - or at least to mitigate - these adverse effects on productivity during the initial manufacturing of a new product.

Japanese manufacturers have shown more success in this respect than their American counterparts: "Detroit auto makers still lag Toyota, Honda and Nissan in several critical areas: the time it takes to launch new vehicles and plant flexibility" (Winter, 2006). Thus it is not surprising that Frank Ewasyshyn, EVP of manufacturing at Chrysler, should envision better performance during launch as a key element for overall productivity improvement: "we'll see very short ramp-ups. We've run the pilots online. We'll see short drops for a matter of days, then right back up again. So the effect is very minimal compared to what's happened in the past" (Mayne, 2006). Of course, speed of ramp-up is only one factor in the productivity equation. We will show here how productivity levels during launch relate to a range of manufacturers' production decisions and the characteristics of the plants concerned.

The productivity effects of the launch process vary greatly between different automotive plants. For example, Figure 4.1 below shows results from five different plants that hosted a launch in 2001. Year 0 establishes a baseline: productivity levels in the year before launch. Relative levels of productivity during the launch year are shown at Year 1, and relative levels of productivity during the year after launch are shown at Year 2. Although we see that productivity at all plants declines during the launch year and improves again during the year after launch, no other patterns are immediately apparent. The magnitude of productivity change observed in the year after launch seems independent of the magnitude

of change in the launch year.

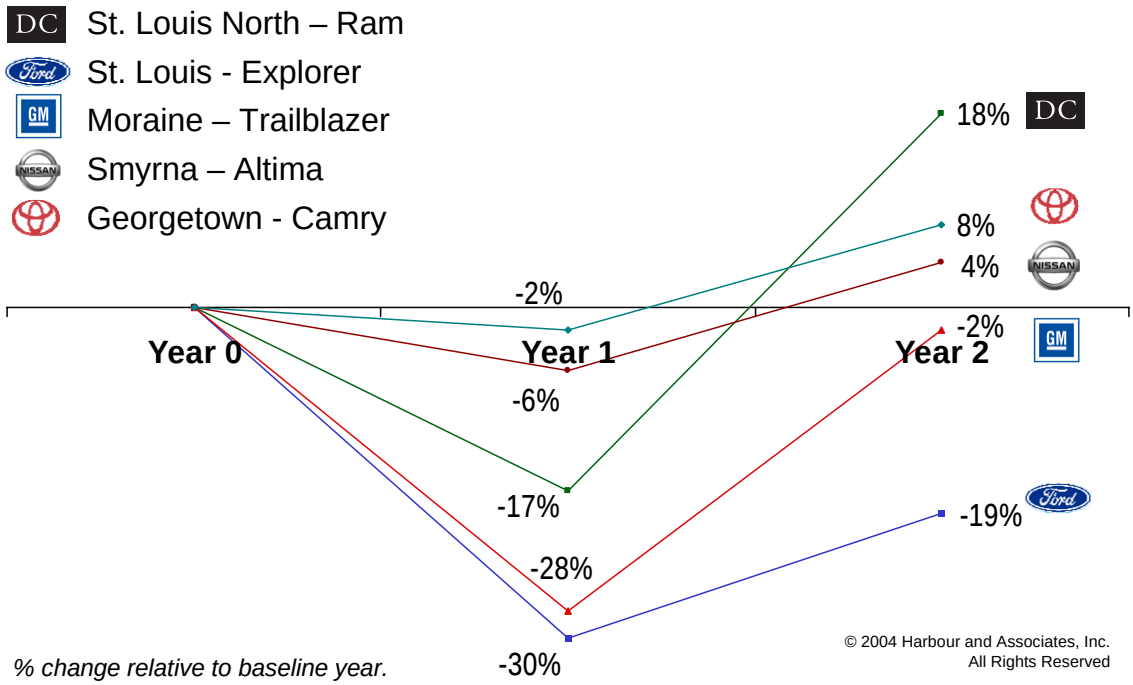


Figure 4.1: Productivity Changes During Launch

Our research questions here concern two key aspects of the product launch process. First, we analyze the "launch location decision" of manufacturers. Are plants that are likely to be chosen to host a new product launch distinguishable in advance from others? For instance, we might expect that highly productive plants or plants with experience on similar vehicles to the launch model would be more likely chosen to host the launch. Second, we examine the productivity effects of the launch, in light of quantitative plant data. What factors are able to explain the variation in plant performance, as exemplified in Figure 1 above? Although interesting in themselves, the answers to these two questions should illuminate a third: to what extent do factors that influence choice of launch location correspond to the factors that characterize better productivity during launch?

Using data that are publicly available from Ward's Automotive and the Harbour Consulting Group, we find support for two hypotheses concerning launch location: a plant's

likelihood of hosting a new product launch is significantly increased if it has prior experience with one of the forthcoming launch platforms, or if it employs flexible production methods. Correspondingly, we find that prior experience or flexible production methods are significant factors for productivity during the first year of the launch process: plants with prior experience or with flexible production tend to show better productivity figures than other plants. When we consider productivity changes from the year before launch to the year after launch (Year 0 to Year 2 in Figure 1), however, neither experience nor flexibility proves to be significant. Instead, only plant utilization appears important for productivity levels across the two-year period.

4.2 LITERATURE REVIEW

This study has commonalities with several streams of literature. Our analysis is set in the automotive industry, but our research questions relate conceptually to work on product development or manufacturing productivity, each of which occurs in numerous contexts.

With focus on the automotive industry, Clark and Fujimoto (1991) provide a comprehensive survey of the processes that lead from new product design to the point of launch, but they do not address the operational aspects of the subsequent transition to manufacturing. Substantial work on the subject of productivity in automotive manufacture has come from researchers affiliated with MIT's International Motor Vehicle Program (IMVP). Krafcik (1988) initiates the analysis of productivity in terms of HPV[1] measures and notes the superiority of Japanese plants from this perspective. Lieberman (1990) shows how productivity improvements may be achieved through more efficient labor utilization. MacDuffie et al. (1996) find that the increased complexity of parts involved in production of an automobile tends to have a detrimental effect on productivity. Similarly, Fisher and Ittner (1999) find that increased variability in vehicle options has a negative impact on produc-

[1] Hours-per-Vehicle: see below, p126

tivity and quality. Lieberman and Demeester (1999) show a strong relationship between higher productivity and inventory reduction. Nevertheless, none of these papers is specifically concerned with productivity during product launch.

Some studies focusing on manufacturing strategies for new products have appeared recently, but these have mostly been set in the context of high-tech industries, where the product life cycles and prior knowledge are generally much lower than in the case of automobile manufacture. For example, Terwiesch and Xu (2004) describe an optimal ramp-up strategy for electronic component manufacture that intentionally delays process change. Within the automotive industry, a study by the Office for the Study of Automotive Transportation (OSAT) at the University of Michigan seems to be the only one addressing productivity during launch. In a series of six articles for Automotive Design and Production magazine, OSAT researchers Smith et al. (1998) examine manufacturing performance during launch at thirty North American automobile plants in the period 1992 to 1998. These authors consider only single-product, single-plant launches, however, and the "series analyzes a launch event in only the most basic form: the time it takes a facility to return to capacity in terms of production" (part 2). Also, the authors' treatment is primarily a discursive comparison of different manufacturers' performance with respect to return to capacity; no statistical analyses are offered. They do, however, suggest the importance of at least one factor that proves to be significant in our study: "General Motors' body shops are currently much more flexible than a decade ago. Such flexibility will enable quicker and likely smoother launches in the future" (part 2). Likewise, "Ford has progressively moved away from hard automation, and increased flexibility in its body shops" (part 3).

As is well known from the work of Womack et al. (1990), flexible production methods were a key component in the early success of Japanese automobile makers, but many subsequent studies of flexibility are not directly concerned with its impact on productivity. Some, such as Jordan and Graves (1995) or Chod and Rudi (2005), show how flexibility is a useful means for hedging against demand uncertainty, particularly when demand correla-

tion between products is low. Others, such as Fine and Pappu (1990) or Röller and Tombak (1990) consider the strategic aspects of flexibility, concluding that it intensifies competition and depresses prices. Goyal and Netessine (2007) model flexibility adoption decisions subject simultaneously to competition and demand uncertainty, finding that the benefits of flexibility to any given firm will usually be tempered by the degree of its competitors' investments in flexible production methods.

Automotive industry research has only recently begun to consider the significance of flexible production methods for productivity. Goyal et al. (2006) consider the actual deployment of flexible production methods in North American automobile plants and assess consistency between the data and hypotheses drawn from current theoretical models. When considering performance implications, they find that flexibility appears to have an adverse effect on productivity. Van Biesebroeck (2007) arrives at a similar conclusion, but shows also the existence of significant interactions between flexibility and other manufacturing variables, which can lead flexibility to be beneficial for productivity in certain contexts. Nevertheless, neither of these studies is concerned with productivity during launch, and our research here thus addresses an apparently open question. We show what value flexible production methods and prior knowledge, among other variables, may have for manufacturing productivity during the ramp-up stage of product development.

4.3 RESEARCH HYPOTHESES

The preceding discussion already suggests that some measure of manufacturing flexibility should be a candidate for influence in the second of our two research concerns, productivity during launch. Of course, we can also surmise that the purported benefits of flexibility may make manufacturers more likely to launch new products in flexible plants. Indeed, any circumstance offering a productivity advantage at a plant is a possible influence on production location decisions. There may be constraints on these decisions that do not

necessarily align with productivity goals, however, and some factors affecting productivity may only become manifest during the period of production ramp-up, long after the location decision had been made.

Based on insights derived from works already noted and others, we propose below a total of nine research hypotheses. These concern key factors in manufacturers' strategic production planning, product life cycles and sales trends, past productivity performance, speed of production ramp-up, flexible production methods, and existing organizational knowledge. We discuss first our hypotheses relating to launch location decisions, then hypotheses about productivity during launch. Without suggesting causality, we would like to evaluate consistency between the hypotheses and our automotive industry data.

Regarding strategic production planning, a recent study by Fleischmann et al. (2006) of practices at BMW indicates the enormous scale of planning involved in producing a new automotive product. The time horizon for the company's production model extends to twelve years, thereby covering the full life cycle of products starting in the next five years. Once new product choices and accompanying sales forecasts have been made, planners begin working on production location decisions. The preeminent aim in making these allocations is to ensure that capacity at each plant closely fits production needs, given the forecast evolution in volume for each model that will be produced there over the twelve-year planning period. Accurate provision of capacity at each plant is a critical factor for the manufacturer, since expanding facilities after product launch can be very expensive, but low utilization tends to harm profitability.

The need to tie total production volume closely to plant capacity entails that new loads resulting from a product launch can compensate for decreased volume on one or more other product lines. If production planning must simultaneously cover many different plants, the complexity of modern automobile production processes and of the accompanying supply chains mean that this compensation for one product by another cannot be optimally decided by a one-dimensional look at unit volumes (Fleischmann et al., 2006). Nevertheless, the ba-

sic element in the decision process remains the trade-off between models, and since we are not interested here in the optimal solution to the problem, only in empirically demonstrable correspondences, it suffices for us to observe them at this fairly coarse level of detail.

Production volume for a specific model may decline steadily over a period of years or stop quite abruptly at some point. In the former case, the change may be part of the manufacturer's strategic plan, or it may follow from market conditions. Strategic planning considerations may specify a life cycle for the product, towards the end of which it will be intentionally phased out; market conditions would most obviously be declining sales. In the case of an abrupt termination of production, strategic reasons are more likely than market conditions.

Our first two hypotheses therefore address factors in a plant's current production portfolio that may lead to it being selected as the host site for a new product launch:

H1: Plants with one or more models nearing the end of their life cycle are more likely to host a new product launch.

H2: Plants with one or more models for which sales are declining are more likely to host a new launch.

With successful forecasting and planning, capacity utilization at an automotive plant may remain high, despite variation in the constituent products and their respective production volumes. In a less ideal situation, however, a manufacturer may have to confront unexpected low utilization at a particular plant. In this case, unless the prevailing causes of low utilization are expected to pass, the plant may become more attractive as a launch site. Thus, as complement to *H1* and *H2*, which address utilization concerns indirectly, we add an explicit hypothesis relating low utilization to a plant's likelihood of hosting a new product launch.

H3: Plants with low utilization are more likely to host a new product launch.

As noted in our introduction, it is commonly believed within the automotive industry that product launches have a negative effect on productivity. Terwiesch and Xu (2004)

indicate that manufacturers of high-tech products also often experience problems with yield losses during production ramp-up, while Pisano (1995) discusses the same phenomenon within pharmaceutical industry. Carrillo and Gaimon (2000) show analytically that even an optimal approach to change in manufacturing processes will entail some short-term losses in productivity. In light of the attendant risks, automotive executives may face a dilemma: launching a new car or truck at a plant that already has low productivity could make matters even worse there, while launching at a plant with high productivity could create an entirely new point of concern. A "rule of thumb" for this situation is by no means clear, but we assume provisionally that loss of existing profitability poses a lesser threat.

H4: Plants with higher productivity are more likely to host a new product launch.

Referring to research where the impact of new or changed processes on manufacturing productivity is analyzed in detail, we can discern a key explanatory variable: an organization's level of knowledge about the production process. In general, production ramp-up is a phase characterized by a low level of understanding of the new product and its associated production processes; but as the firm works through initial difficulties and learns how to manufacture the product as efficiently as possible, its level of knowledge increases (Terwiesch and Bohn, 2001). Early models of the learning curve claim that variable costs fall with the logarithm of cumulative production (Wright, 1936; Levy, 1965). More recent studies also see knowledge as advantageous for production yields, however, and suggest that knowledge levels can be improved by deliberate measures such as preparation and training (Hatch and Mowery, 1998; Carrillo and Gaimon, 2000).

In addition to "direct" augmentations of knowledge through experience or training, Levitt and March (1988) argue that one organization can benefit indirectly from the experience of another. Epple et al. (1995) show that the same phenomenon can take place between units entirely within the firm: almost all knowledge gained by the first shift in an automobile plant is transferred to the second shift [2]. These findings suggest that perfor-

[2]The authors do not explain why the knowledge level at the plant is initially low. In particular, no reference is made to implemented process change or product launch.

mance during the production ramp-up phase may benefit from organizational knowledge of processes used for similar products, and that such benefits could be greatest when this relevant organizational knowledge has been developed at the very plant and workforce where the product launch will occur. In automotive production, a significant degree of similarity can obtain between two nominally distinct products when they share the same platform, which may be defined as a subassembly unit that is common to an entire product family (Muffatto and Roveda, 2002). Plants with manufacturing experience on the platform that will be used in a new product should therefore have an advantage over others.

H5: Plants with experience on the launch platform are more likely to host the launch.

Turning now to flexible production methods, we find a large body of literature advocating their value. Hayes and Wheelwright (1984) were perhaps the first to point out the competitive importance of manufacturing flexibility, and the work of Womack et al. (1990) greatly raised awareness about Japanese automakers' advantage in this respect. More recent research recognizes different types of manufacturing flexibility, distinguishing in particular volume flexibility from product flexibility (Parker and Wirth, 1999). We focus on the latter type, which is arguably the more important for the automotive industry (Goyal and Netessine, 2007; Jordan and Graves, 1995). Product flexibility means here that a given plant is capable of producing more than one type of automobile on one or more of its production lines. More precisely, we are interested in demonstrated product flexibility, since it is essentially impossible - and perhaps not even practically relevant - to quantify a plant's potential for flexible production[3].

Most studies of flexibility to date have not, however, been directly concerned with its impact on productivity. Instead, flexibility has widely been evaluated as a means to mitigate

[3]As noted by Goyal et al. (2006), a automobile plant can in principle manufacture multiple types of vehicle, but the cost of converting, replacing, or augmenting the plant's equipment and human resources in order to do so may in many cases be prohibitive. A well-known illustration of this point is provided by Chrysler's experience with the launch of its PT Cruiser. The company intended to manufacture the vehicle at a plant that already produced the Dodge Neon, but the PT cruiser needed a slightly larger paint bath (Boudette, 2006). Replacing this equipment would have been so costly that Chrysler preferred to switch production to a different plant.

the negative effects of demand uncertainty, particularly when low demand correlation exists between products. See, for example, Fine and Freund (1990), Van Mieghem (1998), and Chod and Rudi (2005). Competitive equilibria in the adoption of flexibility are studied by Fine and Pappu (1990), Röller and Tombak (1990), and Goyal and Netessine (2007).

Recent work by Van Biesebroeck (2007) does specifically examine the value of flexibility for productivity, and in the context of the automobile industry. Its potential benefits are shown to depend significantly on the degree of variety among products manufactured. All else equal, flexibility tends to decrease productivity at a plant, but the penalty is itself decreasing in the total number of chassis configurations and body styles produced at the plant. Moreover, the value of flexibility is enhanced when plants "insource" rather than outsource ancillary production activities.

In light of these broadly positive findings, we expect flexibility also to provide benefit during the ramp-up period of product launch.

H6: Plants using flexible production methods are more likely to host a launch.

Unlike *H1 - H3*, which address circumstantial reasons why a plant might be chosen to host a launch, *H5* and *H6* concern attributes of the plant itself - platform experience or use of flexible production methods, respectively - that may entail productivity benefits during launch, and thus influence manufacturers' decisions. The question of whether these possible benefits tend to be realized during a launch period therefore provides us immediately with two hypotheses about the productivity impact of the launch process. Indeed, a plant's experience with a launch platform appears essentially to be a variation of the case of deliberate preparation and training, the benefits of which are suggested by Hatch and Mowery (1998) and Carrillo and Gaimon (2000).

H7: Plants with experience on the launch platform will show better productivity during launch.

A hypothesis about the value of flexible production methods during launch is a natural counterpart to the large research literature on flexibility. As noted earlier, however, the

study made by Smith et al. (1998) suggests that flexibility should be beneficial for automotive product launches, so our treatment of it here provides the first empirical verification of their work.

H8: Plants using flexible production methods will show better productivity during launch.

Our final hypothesis brings the speed of the ramp-up processes into the productivity picture. The ramp-up phase is regarded as lasting from the start of production until the attainment of full capacity utilization. "Faster" ramp-up therefore corresponds to a relatively shorter time to reach full capacity utilization. As noted by Terwiesch and Bohn (2001), in a situation where the nuances of a manufacturing process are new and poorly understood, there may be a trade-off in this period between production rate and yields. As managers apply pressure to accelerate production, the number of errors and defects requiring rework may rise.

H9: Plants with shorter ramp-up phases will be relatively less productive.

In the following section we describe the data and derived variables that we use to test the hypotheses just developed.

4.4 DATA & DERIVED VARIABLES

Our data are compiled from two sources: annual reports of the Harbour Consulting Group and the Reference Center of Ward's Automotive. The Harbour reports provide detailed sets of production data and performance measures for the North American automotive industry. Most automotive manufacturers voluntarily provide data to Michigan-based Harbour, which audits the information provided by means of plant visits. From Ward's Automotive, we obtain supplementary detail of monthly sales and and production volume, to the specificity of individual models and plant.

The data used for our study here are based on eight Harbour reports, covering produc-

tion in the years 1999 to 2006. Although the reports include data from all active automotive plants in North America, new product launches occur at only a small number of plants each year. From eight years of data, we obtain a total of 333 plant records, of which 63 are launch instances. Of these 63 records, 3 are excluded because of partially missing data[4].

For each plant in each year, the Harbour reports indicate an inverse measure of productivity, Hours-per-Vehicle (HPV). This variable is constructed by dividing the total number of working hours expended at the plant in the year by the number of vehicles produced in the same year. The measure is an inverse in the sense that a greater value for HPV indicates lower productivity, and vice-versa. HPV is accepted in the industry as a measure of productivity and has been used in several other research studies (MacDuffie et al., 1996; Goyal et al., 2006; Van Biesebroeck, 2007).

In addition to the variables explicitly reported by Harbour, we construct 23 others for use in the analysis. These additional variables are described in the next few paragraphs.

Flexibility. We provide four measures of flexible production methods at plant. Following the approach taken by other researchers, flexibility is construed as a relation between the number of assembly lines or body shops at each plant and the number of platforms that are produced at the same plant. If the number of platforms produced exceeds the number of assembly lines, we consider the plant to be "assembly line flexible." Otherwise, we consider it to be inflexible (Goyal et al., 2006; Van Biesebroeck, 2007). Similarly, if the number of platforms produced exceeds the number of body shops at the plant, we consider the plant to be "body shop flexible." Each of these two flexibility measures is recorded as a binary variable (flexible vs. inflexible) and as an explicit ratio of the underlying measures (number of platforms produced vs. number of assembly lines or body shops). By the criteria described above, 13 of the 60 launch records in our data are "assembly line flexible," while 11 plants are "body shop flexible."

Experience. Two binary variables are constructed to indicate whether a plant has prior

[4]The plants in question did not exist prior to the product launch year, so productivity changes cannot be assessed.

experience with a platform used to launch a car or truck in a given launch year. Two variables are needed here because we want to distinguish between the effect of experience on the launch location decision and the effect of experience on productivity during launch. In the context of our launch location analysis, we say that a plant that has prior experience in a given year if it manufactures vehicles on at least one of the platforms that will be used to launch new products in the succeeding year (this might be termed "potential experience in launch"). Plants that do not manufacture on any of the platforms that will be used for a launch in the succeeding year are noted as having no potential experience in launch. Since we cannot presume that experience in this sense entails that a plant will host a launch, we recode the experience variable when we look at productivity during launch. In this latter context, saying that a plant hosting a launch has prior experience (which might be termed "realized experience in launch") is equivalent to saying that it did in fact manufacture vehicles on the launch platform in the year preceding launch. Clearly, if a launch plant has "realized experience in launch," it must have had "potential experience in launch." Nevertheless, if a plant does not have "realized experience in launch," it still may have had "potential experience in launch," since a plant with prior experience on one or more of the platforms that will be used for launch may nevertheless host a launch on a platform that lies outside its set of experience.

The "potential" and "realized" experience variables are particular respectively to the launch location analysis and productivity analysis, so we henceforth refer to them both simply as "experience." Context indicates which variable we mean.

Production Ramp-up. Productivity during launch may be affected by the rapidity of production ramp-up, so we add variables to measure the latter. Production ramp-up is defined to be the period between the end of product development and full capacity production (Terwiesch and Bohn, 2001). Manufacturers' detail of production capacity for launch products is not available to us, so we estimate the ramp-up period from the data in four different ways: the time taken for weekly production in the launch year to reach its maximum, its

90th percentile, or the mean or the median of its weekly levels in the second half of the year. Each of the four resulting variables is included in the analysis.

Sales Trends. In order to capture possible influence of sales trends on the decision for launch location, we regress monthly sales data against time for each model produced at each plant that hosts a launch. The regressions are run for 1, 2, 3, and 4 years prior to the launch year (cases for which insufficient data exist are omitted). Parameter estimates from the regressions are then matched to the launch plant records. Plants that produce more than one product will receive several parameter estimates, which may present conflicting sales trends. We address such conflicts by taking the minimum, maximum, and mean parameters estimates from all regressions with the time frame. Thus we have a total of twelve aggregate measures for each plant. The minimum coefficient from regression across two years proves to be most significant for our models, however, so results with this measure are reported.

End of life-cycle. Our product data allows us to see when models are phased out of production. At the plant level, we add an indicator variable in each year, to show whether the plant manufactures a model that is within the last two years of its life cycle.

Control Variables. In order to control for possible trends across companies or time in our analyses, we utilize indicator variables for manufacturer and year.

The full set of variables used in the study is described at the start of the appendix to this chapter. Summary statistics appear in Table C.1. Sample correlations for the launch location data are in Table C.2, while sample correlations for the productivity data are in Table C.3. Data on shift patterns and plant capacity are available to us but ultimately not used in the analysis. The great majority (82%) of launch plants in our data operate with two shifts, and no statistically significant difference between the mean productivity of this group and the mean of groups with other shift patterns (one, three, or four shifts) is discernable. Plant capacity did not prove to be a significant factor for either of our research questions.

4.5 ANALYSIS

We present the analysis of our data in four parts. First, we consider the general impact of launch events on productivity. Notwithstanding anecdotal evidence cited earlier (see section 4.1), we show from the data that a productivity penalty appears to be associated with launch activity. Next, we model manufacturers' launch location decisions by means of logistic regression. This shows which variables in our dataset tend to contribute to the probability of a launch occurring at a plant in a given year. Third, we give the productivity analysis itself, showing which variables may mitiage the producitivity penalty that often accompanies a launch. Finally, we address the possibility that our results on productivity could be biased by manufacturers' choice of launch location. In order to support our findings on the productivity penalty associated with launch, we use propensity scoring and other methods to view productivity performance at launch plants in light of performance at sets of comparable non-launch plants. In order to argue that the results of the productivity analysis are unbiased, we estimate a Heckman correction model (Heckman, 1979).

4.5.1 LAUNCH IMPACT

OLS regression on our data provides an initial perspective on the productivity impact of a launch event. Considering first a model that includes launch and non-launch plants alike, but excludes the binary launch indicator variable, we find that significant predictors of productivity at any plant in any year are plant utilization, the number of chassis configurations on the assembly lines, and the productivity level in the preceding year (see "base model," Table C.4). When added to this base model, the binary launch indicator variable takes a significant and positive parameter estimate. Thus launch tends to entail higher HPV figures, *i.e.*, lower productivity (see "launch model," Table C.4). The regression is on the logarithmic scale, so the parameter estimate of 0.150 for the launch indicator implies that launch plants tend on average to see 16.2% greater HPV than non-launch plants.

4.5.2 LAUNCH LOCATION

In estimating a location model, we restrict our data to variables that provide data from before launch, such as prior year capacity, utilization, HPV, and the regression estimates of sales trends discussed earlier. We include the indicator variables for each company[5] and each production year, in order to control for any significant differences in these areas.

Since some plants in the data have multiple observations (across different production years), tests of significance for each coefficient are computed using robust standard errors, clustered by plant. Tests without clustered standard errors may bias the tests in favor of accepting the null hypothesis.

The results of logistic regression on all relevant variables are shown in Table C.5. We see that prior experience with the launch platform and body shop flexibility[6] are significant predictors of launch location, supporting hypotheses 5 and 6. As expected, the parameter estimates for these variables are both positive: having prior experience or flexible production methods increases a plant's chance of hosting a launch. The parameter estimate for prior-year HPV is also significant, but negative: poorer productivity (higher HPV) decreases a plant's probability of hosting a launch. This entails support for hypothesis 4.

The regression does not give any support to our hypotheses about product life cycles, sales, or plant utilization (hypotheses 1, 2, and 3). The estimates for the late model indicator, sales trend, and utilization variables are not statistically significant. Since table C.1 indicates that mean prior utilization is close to 90%, we check further to see whether our finding on hypothesis 3 may result from generally high plant utilization in our data. Of 147 plants with below mean utilization in a given year, however, only 28 were chosen to host a product launch in the following year.

Among control variables, only DCX and 2004 are significant: the proportion of plants

[5]The indicator for GM is suppressed in the estimation.

[6]Due to high correlation among the four available flexibility measures, the model was estimated with the inclusion of only one of them at a time. The parameter estimate for body shop flexibility proved to be the most significant.

hosting a launch is higher for Daimler-Chrysler than for other manufacturers, while the overall number of launches in 2004 is proportionately higher than in other years.

4.5.3 LAUNCH PRODUCTIVITY

The results of our launch location analysis suggest relevant experience, flexible production methods, and superior productivity at a given plant all contribute positively to the likelihood that a manufacturer will choose to host a launch there. The question of whether these factors correspond significantly to any productivity advantage during launch itself is thus highly material (hypotheses 7 and 8).

We use OLS regression to model productivity changes during the first year and second years following a new vehicle launch. Specifically, the effects modeled are the change in HPV from the year preceding launch to the year of launch, which we call *Y01effect*; and the change from the year preceding launch to the second year after launch, which we call *Y02effect*. We do not present an analysis of the change in HPV between the first and second years after launch (which we would call *Y12effect*), since the parameter estimates for this model can generate ambiguities: a plant that shows a great improvement in HPV during the second year after launch may be performing well over both years, or merely recovering from a particularly great loss in productivity in the first year of launch.

The changes in HPV given by *Y01effect* and *Y02effect* can be measured as absolute differences or as percentage measures. We estimate all four possible models, and find greater statistical significance for our independent variables when using the percentage measure. We present these more significant results. The parameter estimates for models of *Y01effect* and *Y02effect* are given in tables C.6, C.7, and C.8. In order to highlight the import of the experience and flexibility variables in the analysis, we estimate first a base model, from which they are excluded. Alongside these estimates we present estimates for an extended model, which includes the experience and flexibility variables.

Considering first the model for *Y01effect* shown in Table C.6, we note the following:

(i) The estimate for the intercept is statistically significant at an HPV increase of 71.39% with respect to productivity before launch. Any statistically significant negative parameter estimates therefore correspond to variables that tend to lower this increase.

(ii) Flexibility of the body shop with respect to platforms significantly mitigates increase in HPV. Moreover, since Flexibility is a continuous measure of flexibility, a greater degree of flexibility in the body shop entails a greater mitigation of HPV increases.

(iii) Also significant to the *Y01effect* is prior manufacturing experience on the platform used for the launch. The estimate is again negative, suggesting, as we would expect, that prior experience tends to result in better productivity during the launch year.

(iv) The estimate for plant utilization during the first year of launch (*Utilization*) is significant. In general, plants with higher levels of utilization show better productivity.

(v) The estimate for standard deviation of sales during the first year of launch (*stdDev(Sales)*) is significant. In general, plants with models for which sales are more variable show lower productivity. This appears consistent with findings of Bresnahan and Ramey (1994), who study of output fluctuations in the U.S. automotive industry. They note that the shift-based manufacturing environment tends to make production more volatile than sales, and that productivity often suffers when output levels are increased, because workers on the newly added shifts are likely to be to relatively inexperienced.

(vi) The estimate for the duration of ramp-up is not significant.

(vii) The estimates for the control variables (company and year) are not significant.

Thus flexibility and prior experience with the launch platform tend to entail a productivity advantage gained during launch., in accordance with hypotheses 7 and 8. Plants with these characteristics may enjoy an economic advantage during launch. This result amplifies

the relevance of the earlier findings that these factors contribute positively to the likelihood of a plant hosting a launch.

Surprisingly, the regression of *Y01effect* offers no support for hypothesis 9. None of the parameter estimates for the four measures[7] of ramp-up duration is significant. This outcome changes, however, if replace the experience indicator by its negation (*i.e.*, an indicator for lack of experience) and include an interaction term between this and ramp-up duration. These results are shown in Table C.7. While the significance, magnitude, and sign of other the estimates are much the same here as in the regression without the interaction term, the coefficient estimate for ramp-up duration is now significant, the coefficient estimate for the interaction term is close to significant (p-value of 0.145), the magnitudes of the estimates are comparable, but their signs are opposite. The overall implication is that ramp-up duration is not very important for plants without experience, but for plants with experience, shorter ramp-ups correspond to better productivity, and vice versa. From this perspective, ramp-up duration appears more akin to a performance measure than an explanatory variable (*cf.* Terwiesch et al., 2001). Indeed, the correlation between ramp-up duration and *Y01effect* is only 0.072 for plants without experience, while it is 0.397 for plants with experience.

Turning to the regression of *Y02effect*, we see from the estimations in Table C.8 that neither prior experience on the launch platform nor flexibility are significant factors for productivity changes over the period of two years following launch. In the case of plants that show improvement over the period (Table C.9), utilization is significant (*avgUtilization*). Since the parameter estimate is negative, we infer in this limited case that higher utilization tends to entail better productivity.

[7] As with the four flexibility measures, there is high correlation among the four available measures of ramp-up duration, so the model was estimated with the inclusion of only one of them at a time. Results were similar for all, but the time to reach mean productivity level gave best overall model significance.

4.5.4 SELECTION BIAS

In order to account for the possibility that the results in Table C.4 may be biased by the selection of launch plants, we benchmark against productivity changes at comparable non-launch plants. In particular, for each launch plant, we first identify the set (possibly empty) of plants within the same company that are active in the launch period, but do not host a launch. From these sets, we determine a comparative non-launch productivity change, according to each of the following criteria:

(i) HPV change at the single non-launch plant that includes the same product segment classification as the launch model, and has the closest HPV to the launch plant prior to launch.

(ii) Mean HPV change at all non-launch plants that include the same product segment classification as the launch model.

(iii) HPV change at the single non-launch plant with the closest propensity score to the launch plant.

The propensity scoring (PS) that we employ for criterion (iii) offers a theoretical basis for matching subjects within a non-treatment group (e.g., plants that do not host a launch) to statistically equivalent subjects within a treatment group (e.g., plants hosting a launch). Introduced by Rosenbaum and Rubin (1983), PS relies on the conditional probability that a subject will be in one group rather than in another. From the PS perspective, two subjects are similar, irrespective of their *de facto* group membership, to the extent that their propensity scores for group membership are *a priori* similar. PS has been employed by many researchers who, while studying the effect of some treatment, need to select an unbiased control group against which to benchmark (see Luellen et al., 2005, for an introduction). For example, Lechner (2002) uses the PS method to evaluate the effectiveness of different job training programs in Switzerland: PS enables *a priori* matching of individuals, even

though their *de facto* selection into a particular job training program may not be random. Logistic regression is a common means for determining propensity scores, so we can immediately apply the results from our launch location analysis for our propensity scoring. The non-launch plant with the closest propensity score to a given launch plant is the one within the same company and launch period that appeared *ex ante* to have the most similar chance as the launch plant to be chosen to host the launch.

Given matches of a non-launch plant to each launch plant, using in turn criteria (i) - (iii) above, we calculate the adjusted mean productivity changes due to launch activity, *i.e.*, the HPV change at each launch plant minus the HPV change at the matched non-launch plant. These results appear in Table 4.1.

Criterion	Adjusted Mean HPV Change	95% Confidence Interval
(i)	+19.06%	(+13.66%, +24.46%)
(ii)	+15.71%	(+10.29%, +21.14%)
(iii)	+19.36%	(+14.44%, +24.30%)

Table 4.1: Adjusted Mean Changes in Productivity

From these figures we see that the mean productivity penalty associated with launch activity is likely to be slightly greater than suggested by the OLS regression in Table C.4. This is consistent with the assumption that launch plants are selected in order to provide the best possible productivity performance during the launch period.

As also observed at the outset of this section, the selection of launch plants may influence productivity performance. Thus we may also ask whether the parameter estimates in our regressions for launch performance are biased by the launch plant selection. In order to address this point, we estimate a Heckman correction model (Heckman, 1979) for launch

location and *Y01effect* (see Table C.10). Under the assumption that the error terms of the selection model and the regression model are jointly normal, the Heckman model estimates ρ, the correlation between them. Viewed as conditional on the selection model, the regression sample is random; thus the results can be considered unbiased if the estimate of ρ is zero. The likelihood ratio test at the end of Table C.10 indicates that we cannot reject the null hypothesis of $\rho = 0$, so so we can take the estimates from the OLS productivity analysis to be unbiased. Indeed, comparing the coefficient estimates of the OLS model in Table C.10 to the coefficient estimates in Table C.6, we see that they are highly similar[8]. Thus we infer that the parameter estimates in our OLS regressions may be extrapolated to cover automotive plants in general, rather than just the set of plants in our data that were observed to host a launch.

4.6 CONCLUSIONS

New product launches tend to have a negative impact on productivity at North American automotive plants, but this effect may be mitigated by careful choice of the plant that will host the launch. Specifically, our analysis suggests that plants with prior product experience and/or flexible manufacturing capabilities may show better productivity performance during launch. Moreover, it is body shop flexibility that we find to be significant, in accordance with conclusions reached heuristically in the study by Smith et al. (1998).

Consonant with our findings, product experience and flexibility appear to increase a plant's likelihood of being chosen to host a launch. Indeed, besides prior productivity performance, product experience and flexibility are the only variables in our dataset that are significant predictors of launch location. This seems to present a contrast with the work of Fleischmann et al. (2006), which suggests that balancing loads across plants is a key element of manufacturers' production planning.

[8]The standard deviation of sales, *stdDev(Sales)*, is not used in the Heckman model, since it is defined as the standard deviation of sales of the launch product.

OLS regression of productivity change during the first year of launch shows that flexibility or experience yield, respectively, a 10% or 15% savings in HPV. Average productivity for all plants in our dataset is 28.5 HPV, so we might suggest a savings of 3 or 4 HPV from flexibility or experience. Average annual production at plants is 194,000. In 2006 a typical UAW-represented assembler at GM earned $27.81 per hour[9], so implied savings at the average launch plant are $16.2MM for flexibility or $21.6MM for experience.

Surprisingly, we find no evidence that Japanese automobile firms are more productive during launch than their U.S. counterparts. References to "the time it takes to launch new vehicles" in the trade press may be intended to include product development time, however, which is not recognized in our study. We also find no evidence of change in general launch productivity levels across the eight years covered by our data.

[9]UAW website.

APPENDIX A

PROOFS FOR CHAPTER 2

Since much of the following discussion considers the relationship between the two sides of the optimality equation (2.11), the following definitions will simplify notation:

$$\xi(S) \equiv \frac{S(S-1) - k_0 a}{S^2 + k_0 a},$$

$$\psi(S, \gamma) \equiv \frac{1 - S^{-\gamma}}{\gamma}.$$

Thus we can see

$$\xi(0) = -1, \qquad \xi(1) = \frac{-k_0 a}{1 + k_0 a} \tag{A.1}$$

$$\psi(0, \gamma_n) = \frac{1}{\gamma_n}, \qquad \lim_{S \downarrow 0} \psi(S, \gamma_p) = {}^{1} - \infty \tag{A.2}$$

$$\psi(1, \gamma) = 0. \tag{A.3}$$

[1] $\lim_{\gamma \downarrow 0} \psi(S, \gamma) = \ln(S)$

Lemma 1. *If* $1 + 4k_0a > 0$ *then* $\xi(S)$ *is concave on* $(\sqrt{-k_0a}, 1]$.

Proof. The second derivative of $\xi(S)$ is

$$\xi''(S) = \frac{-2S^3 - 12S^2k_0a + 6Sk_0a + 4(k_0a)^2}{(S^2 + k_0a)^3}.$$

Define $N(S)$ and $D(S)$ as the numerator and denominator of $\xi''(S)$,

$$N(S) = -2S^3 - 12S^2k_0a + 6Sk_0a + 4(k_0a)^2,$$
$$D(S) = (S^2 + k_0a)^3.$$

The sign of $N(\sqrt{-k_0a})$ changes according to whether $1 + 4k_0a < 0$ or $1 + 4k_0a > 0$, since

$$\begin{aligned} N(\sqrt{-k_0a}) &= -2(-k_0a)^{\frac{3}{2}} + 12(k_0a)^2 - 6(-k_0a)^{\frac{3}{2}} + 4(k_0a)^2 \\ &= (-k_0a)^{\frac{3}{2}}(16\sqrt{-k_0a} - 8). \end{aligned}$$

The multiplicand $(-k_0a)^{\frac{3}{2}}$ is positive by assumption, while $(16\sqrt{-k_0a} - 8)$ is negative if $1 + 4k_0a > 0$ and positive if $1 + 4k_0a < 0$. Thus $N(\sqrt{-k_0a}) > 0$ if $1 + 4k_0a < 0$ and $N(\sqrt{-k_0a}) < 0$ if $1 + 4k_0a > 0$.

Considering further the case $1+4k_0a > 0$, we note that $N'(S) = -6S^2-24Sk_0a+6k_0a$. The discriminant of $N'(S)$ is $\sqrt{4k_0a(1 + 4k_0a)}$. Again, $-k_0a$ is positive by assumption, so $N'(S)$ does not change sign when $1 + 4k_0a > 0$. Since $N'(0) = 6k_0a < 0$, we have $N'(S) < 0$ for all $S \in [0, 1]$. This and $N(\sqrt{-k_0a}) < 0$, established above, imply that $N(S) < 0$ for all $S \in (\sqrt{-k_0a}, 1]$. Because $D(S) > 0$ for all $S \in (\sqrt{-k_0a}, 1]$, we have $\xi''(S) < 0$ on $(\sqrt{-k_0a}, 1]$ and therefore $\xi(S)$ is concave on the interval. □

Lemma 2. $\psi(S, \gamma_n)$ *is concave when* $\gamma_n \in (-1, 0)$ *and convex when* $\gamma_n < -1$.

Proof. Follows immediately from

$$\psi_{SS}(S,\gamma) = -(\gamma+1)S^{-(\gamma+2)}.$$

□

Proposition 1. *If $1 + 4k_0a > 0$, there is a unique solution S_n to (2.11) on $(\sqrt{-k_0a}, 1]$ for each solution γ_n of (2.12).*

Proof: (All claims in the proof are made for $S \in (\sqrt{-k_0a}, 1]$.)
The endpoints of $\xi(S)$ and $\psi(S,\gamma)$ are

$$\lim_{S\downarrow\sqrt{-k_0a}} \xi(S) = -\infty < \psi(\sqrt{-k_0a})$$
$$\xi(1) > \frac{1}{3} > \psi(1,\gamma) = 0.$$

Since the functions are continuous, the conditions on the endpoints ensure that the set $\{s\}$ of points where $\xi(s) - \psi(s,\gamma_n) = 0$ is non-empty. Let $s_m = \min\{s\}$. If $\psi(S,\gamma_n)$ is convex then the result is immediate, since

$$\begin{aligned}\psi(\lambda s_m + (1-\lambda)1, \gamma_n) &< \lambda\psi(s_m,\gamma_n) + (1-\lambda)\psi(1,\gamma_n)\\ &< \lambda\xi(s_m) + (1-\lambda)\xi(1)\\ &< \xi(\lambda s_m + (1-\lambda)1).\end{aligned}$$

If $\psi(S,\gamma_n)$ is concave, then s_m is unique if $\xi'(S) - \psi_S(S,\gamma) = 0$ has at most one solution; equivalently, if there is at most one solution to

$$\frac{\xi'(S)}{\psi_S(S,\gamma)} = 1.$$

We have $\xi'(\sqrt{-k_0 a}) = \infty$ and $\psi_S(\sqrt{-k_0 a}, \gamma)$ finite, while $\xi'(1) = \frac{1+3k_0 a}{(1+k_0 a)^2}$ (finite) and $\psi_S(1, \gamma) = 1$. Thus if $\frac{\xi'(S)}{\psi_S(S,\gamma)}$ is monotone decreasing, all solutions will be unique.

Thus we require

$$\frac{\partial}{\partial S}\left(\frac{\xi'(S)}{\psi_S(S,\gamma)}\right) < 0$$

$$\Leftrightarrow \quad \frac{\psi_S(S,\gamma)\xi''(S) - \xi'(S)\psi_{SS}(S,\gamma)}{(\psi_S(S,\gamma))^2} < 0.$$

$$\Leftrightarrow \quad \psi_S(S,\gamma)\xi''(S) - \xi'(S)\psi_{SS}(S,\gamma) < 0$$

$$\Leftrightarrow \quad \frac{\psi_S(S,\gamma)}{\psi_{SS}(S,\gamma)} < \frac{\xi'(S)}{\xi''(S)}$$

$$\Leftrightarrow \quad \frac{S}{\gamma} + \frac{(S^2 + k_0 a(4S-1))(S^2 + k_0 a)}{2S^3 + 12S^2 k_0 a - 6Sk_0 a - 4(k_0 a)^2} < 0$$

$$\Leftrightarrow \quad -S + \frac{(S^2 + k_0 a(4S-1))(S^2 + k_0 a)}{2S^3 + 12S^2 k_0 a - 6Sk_0 a - 4(k_0 a)^2} < 0.$$

The two multiplicands in the numerator of the right hand expression are positive, as is the denominator. Thus we now require

$$(S^2 + k_0 a(4S-1))(S^2 + k_0 a) - 2S^4 - 12S^3 k_0 a + 6S^2 k_0 a + 4S(k_0 a)^2 < 0$$

$$\Leftrightarrow \quad q(S) \equiv -S^4 - 8S^3 k_0 a + 6S^2 k_0 a + 8S(k_0 a)^2 - (k_0 a)^2 < 0.$$

Checking the left endpoint shows

$$q(\sqrt{-k_0 a}) = 8(k_0 a)^2(2\sqrt{-k_0 a} - 1) < 0,$$

which is true because $\sqrt{-k_0 a} < \frac{1}{2}$. Similarly,

$$q(1) = -1 - 2k_0 a + 7(k_0 a)^2 < -\frac{1}{2} + \frac{7}{16} < 0.$$

Nevertheless, $q(S)$ is concave, as can be seen by calculating

$$q''(S) = -12(S^2 + k_0 a(4S - 1)) < 0.$$

Thus we need to check the possibility that $q(S)$ takes its maximum at some point $\bar{S} \in (\sqrt{-k_0 a}, 1]$.

The first order condition is

$$q'(\bar{S}) = \bar{S}^3 + 6\bar{S}^2 k_0 a - 3\bar{S} k_0 a - 2(k_0 a)^2 = 0$$

$$\Rightarrow q(\bar{S}) = q(\bar{S}) + (\bar{S} + 2k_0 a)q'(\bar{S})$$

$$= (3\bar{S}^2 k_0 a - (k_0 a)^2)(1 + 4k_0 a) < 0.$$

We conclude that $q(S) < 0$ for all $S \in (\sqrt{-k_0 a}, 1]$, so $\xi(S) - \psi(S, \gamma)$ has a unique zero on the interval. $\diamondsuit$

Proposition 2. *The solution $S_n \in (\sqrt{-k_0 a}, 1]$ established by Proposition 1 is strictly de-*

creasing in k_0. Moreover, holding $a < 0$ fixed, we have

$$1 + 4k_0a \uparrow 1 \Rightarrow S_n \uparrow 1,$$
$$1 + 4k_0a \downarrow 0 \Rightarrow S_n \downarrow \frac{1}{2}.$$

Proof: Note first that for fixed $\bar{S} \in (\sqrt{-k_0a}, 1]$, taking $k_0^* > k_0$ along with $a < 0$ implies

$$\bar{S}(\bar{S} - 1) - k_0a < \bar{S}(\bar{S} - 1) - k_0^*a,$$
$$\bar{S}^2 + k_0a > \bar{S}^2 + k_0^*a.$$

Thus, rewriting $\xi(S)$ as a function of S and k_0, we see that

$$\xi^*(S, k_0) = \frac{S(S-1) - k_0a}{S^2 + k_0a}$$

is strictly increasing with k_0 when S is held fixed.

Now for some $\epsilon > 0$, consider the point $S_n - \epsilon \in (\sqrt{-k_0a}, 1]$. Since $\xi'(S) > 0$ and $\psi_(S, \gamma_n) > 0$ for all $S \in (\sqrt{-k_0a}, 1]$, we have $\xi(S_n - \epsilon) < \xi(S_n)$ and $\psi(S_n - \epsilon, \gamma_n) < \psi(S_n, \gamma_n)$. Moreover, since S_n is the unique point of equality between these two functions, $\xi(S_n - \epsilon) < \psi(S_n - \epsilon, \gamma_n)$. If we replace $\xi(S_n - \epsilon)$ by $\xi^*(S_n - \epsilon, k_0)$, Proposition 1 guarantees that the existence of some $k_0^* > k_0$ such that $\xi^*(S_n - \epsilon, k_0^*) = \psi(S_n - \epsilon, \gamma_n)$. Thus S_n is strictly decreasing as k_0 increases.

The upper limit for S_n follows because $\xi^*(1, 0) = 0 = \psi(1, \gamma_n)$.

In order to obtain the lower limit for S_n, we show first that $\psi(\frac{1}{2}, \gamma_n) > -1$. Specifically, we have

$$\frac{1 - (\frac{1}{2})^{-\gamma_n}}{\gamma_n} \geq -1$$
$$\Leftrightarrow 1 + \gamma_n \leq 2^{\gamma_n},$$

with equality obtaining only if $\gamma_n = 0$. Thus $\psi(\frac{1}{2}, \gamma_n) > -1$ because

$$1 = \frac{d}{d\gamma_n}(1 + \gamma_n) > \frac{d}{d\gamma_n}(2^{\gamma_n}) = 2^{\gamma_n} \ln 2.$$

Nevertheless, $\xi(\frac{1}{2}) = -1$. Thus for $\epsilon > 0$ we can choose k_0 such that $\xi^*(\frac{1}{2} + \epsilon, k_0) = \psi(\frac{1}{2} + \epsilon, \gamma_n)$. Taking $\epsilon \downarrow 0$ requires $1 + 4k_0a \downarrow 0$ in order to maintain equality.

Thus $1 + 4k_0a \downarrow 0 \Rightarrow S \downarrow \frac{1}{2}$. ♢

Proposition 3. *If* $1+4k_0a < 0$*, there is no solution to (2.11) on* $(\sqrt{-k_0a}, 1]$ *for any solution* γ_n *of (2.12).*

Proof: Since $0 < -k_0a < 1$, we have $\xi(1) > 0$. The discriminant of the numerator of $\xi(S)$ is negative on the interval, so $\xi(S) > 0$. Nevertheless, $\max \psi(S) = \psi(1) = 0$. ♢

APPENDIX B

SIMULATION CODE FOR CHAPTER 3

Simulation code appears in the "listing" portion of this appendix section. The listing works with the BORLAND® C++ compiler (version 5.5) running on a WINDOWS® operating system. Key parameters for the simulation are read at runtime from a text file in the same directory as the executable. If the parameter file is, for example, *run01.txt*, then the executable expects a command line parameter "run01" that enables it to locate the input data. The policy simulation results are appended to the file *results.txt* (generated if needed) and the learned policy is written to *run01_policy.txt*.

A sample input file is given below, followed by explantory notes for the content. (Note the the consecutive integers given in the margin are line numbers, and should not be included in the file itself.) Where appropriate, the corresponding variable names from sections 3.3 and 3.5 are given in parantheses.

```
 1  NC  3
 2  NL  3
 3  ND  100000
 4  learn                1
 5  minV                 100
 6  learnMultiple        100
 7  totalReps            50
 8  capacity             125
 9  penaltyThreshold     100
10  invDistPr            0.100
11  interdemandMean      0.01
12  learnRate            1.00
```

```
13 penalty              0.25
14 classDist            0.10   0.20   0.70
15 potentialRewards     1.00   0.90   0.75
16 rejectPr             0.00   0.10   0.20
17 leadTimes            1  2  3  4  5  6  7  8  9
18 userPolicy           100  100  999  100  100  999  100  100  999
```

- *NC* - Number of demand classes (n).
- *BL* - Number of lead times per demand class (N).
- *ND* - Number of production days simulated for a single policy test.
- *learn* - Learning mode: 0, 1, or 2. See comments in listing below, lines 306 - 316.
- *minV* - Number of visits needed to a state before its *Q*-factors are counted in the learned policy.
- *learnMultiple* - Length of learning phase is *learnMultiple*ND*.
- *totalReps* - Number of replications for each policy test.
- *capacity* - Maximum system capacity (c).
- *penaltyThreshold* - Nominal system capacity (b).
- *invDistPr* - Probability of supply chain disruption (τ).
- *interdemandMean* - Inverse of total daily demand ($1/\lambda$).
- *learnRate* - Multiplier of α^m in the Robbins-Monro algorithm.
- *penalty* - Unit overtime cost (u).
- *classDist* - Distribution of demand across classes (δ_i).
- *potentialRewards* - Potential rewards per class ($r(i)$).
- *rejectPr* - Probabilities of rejection for lead times, ordered longest to shortest.

- *leadTimes* - Possible lead times for the system, ordered from highest class and from shortest length within each class.

- *userPolicy* - Fixed threshold levels for system days, ordered in correspondence to possible lead times. To be tested as a user-defined policy when *learn* equals 0.

PROGRAM LISTING

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <math.h>
4  #include <string.h>
5  #include <time.h>
6
7  /* Parameters for random number generator and learning. */
8  #define a (double) 16807.0
9  #define m (double) 2147483647.0
10 #define q (double) 127773.0
11 #define r (double) 2836.0
12 #define SMALL (double) -1e6
13
14 /******** global variables **************
15   NP  Number of policy types.
16   NC  Number of classes.
17   NL  Number of lead time possibilities days for each class.  By
         assumption, NL > 1!
18   Ax  Resulting size of the array: NC*NL+1.
19   NS  Size of state space with respect to each decision (simple
         threshold policy will not use all states).
20   NM  Number of scheduling modes.
21   NE  Number of system events.
22   ND  Length of simulation (days).
23 */
24
25 int NP, NC, NL, Ax, NS, NM, NE, ND, capacity, penaltyThreshold,
     simMode, minV, learnMultiple, totalReps, nextEventType,
     demandType, currentDay, currentDayIndex, repDemands,
     repSuccesses, repRejects, repLosses, repReschedules,
     repRescheduleLosses, totalDemands, totalSuccesses,
     totalRejects, totalLosses, totalRescheduled,
     totalRescheduleLosses, totalExcesses, intLearnDuration, *
     refState, **leadTimes, **userPolicy;
26
27 float *demandDist, *rejectPr;
```

```
28
29 double simTime, interdemandMean, invDistPr, timeNextEvent[5],
       penalty, learnRate, repReward, repPotentialReward, *repData, *
       potentialRewards;
30
31 static double zrng[6];
32
33 FILE *fileInput, *filePolicy, *fileResults;
34
35 time_t timeStart;
36 /******** end global variables ************/
37
38 /************ typedefs ************
39 The dynamically allocated arrays for the structs entail ....
40
41 typedef struct {
42 int state[NP][NC+7][Ax];
43 double reward[NP][Ax];
44 }stcState;
45
46 The final index in the arrays denotes the day (rolling index)
47   The [NC + 7] indices are (for the day referenced by the second)
48   0 to NC-1: number of scheduled type 0 to type NC-1 orders
49   NC    03: currentLoad (sum of scheduled orders across all types
         )
50   NC+1  04: date index (integer)
51   NC+2  05: currentStateA (days remaining to production)
52   NC+3  06: prevStateA (days remaining to production at time of
         previous action)
53   NC+4  07: prevStateB (generalized load at time of previous
         action)
54   NC+5  08: prevAction (previous action)
55   NC+6  09: prevArrivalType (previous arrival type, i.e., new or
         reschedule)
56   NC+7  10: currentStateB (generalized load at time of current
         action, i.e., load across one or more days)
57   reward: prevReward (for the day referenced by the second index
         in state)
58
59 typedef struct {
60 double Q[NP][NC*NL][NS][NM][NL+1]; The extra space in the last
       index of Q is to store the learnt policy.
61 int V[NP][NC*NL][NS][NM][NL];    The penultimate index is to
       distinguish between standard scheduling and rescheduling
       events (NM is always binary).
62 }stcStat;
63 */
```

```
64
65 typedef struct {
66 int ***state;
67 double **reward;
68 }stcState;
69
70 typedef struct {
71 double *****Q;
72 int *****V;
73 int prevClass, prevArrival, prevState, prevAct, currentState,
      currentAction;
74 double prevReward, currentReward;
75 }stcStat;
76 /************end typedefs***********/
77
78 double round(double x);
79 double exponVariate(int stream, double mean);
80 double uniformVariate(int stream, float lower, float upper);
81 double lcgrand(int stream);
82 double getReward(stcState*, int policyIndex, int intDecision, int
     intOfferIndex, int intLTO, int intNewArrival);
83 void  initialize(stcState*, stcStat*, int rep);
84 void  timing(void);
85 void  demand(stcState*, stcStat*, int intNewArrival, int
      policyIndex);  /* intNewArrival = 0 implies rescheduling event
      */
86 void  end_day(stcState*, stcStat*, int rep);
87 void  decodeState(int state, int* stateLevels, int policyIndex);
88 void  conclude(stcStat*, int rep);
89 void  updateState(stcState*, int policyIndex, int intOfferIndex);
90 void  learnRecord(stcState*, stcStat*, int policyIndex, int
      intOffer, int intNewArrival, int intAction, double dblReward);
91 void  rqlearn(stcState*, stcStat*, int intNewArrival, int
      intRecordIndex, int terminal, int policyIndex);
92 void  reschedule(stcState*, stcStat*, int intToReschedule, int
      policyIndex);
93 int   schedule(stcState*, stcStat*, int intNewArrival, int
      policyIndex);
94 int   nextDemandType(void);
95 int   makeOffer(int intOffer);
96
97 int main(int argc, char** argv){
98
99   int i, j, k, l, learn, start, end, rep, maxRep;
100  char *inputSummary, *pch, *pathUltimate, *pathPenultimate, *
       fileInputStr, *filePolicyStr;
101  stcState stateVars;
```

```
102   stcStat stat;
103
104   if(argc == 1){
105     printf ("Job␣input␣parameters␣are␣needed!\n");
106     exit(1);
107   }
108   else if(argc > 2){
109     printf ("Too␣many␣input␣parameters!\n");
110     exit(1);
111   }
112   else{
113     pathUltimate = "x";
114     pch = strtok (argv[0],"\\.");
115     while (pch != NULL){
116       pathPenultimate = pathUltimate;
117       pathUltimate = pch;
118       pch = strtok (NULL, "\\.");
119     }
120     inputSummary = argv[1];  /* Short, contiguous string
          summarizing key input parameters. */
121
122     fileInputStr = calloc(strlen(pathPenultimate) + strlen(
          inputSummary) + 5, sizeof(char)); /* Add 'inputSummary'
          and ".txt\0" */
123     strcat(fileInputStr, pathPenultimate);
124     strcat(fileInputStr, inputSummary);
125     strcat(fileInputStr, ".txt");
126
127     filePolicyStr = calloc(strlen(pathPenultimate) + strlen(
          inputSummary) + 12, sizeof(char)); /* Add 'inputSummary'
          and "_policy.txt\0" */
128     strcat(filePolicyStr, pathPenultimate);
129     strcat(filePolicyStr, inputSummary);
130     strcat(filePolicyStr, "_policy.txt");
131   }
132   free(inputSummary);
133   free(pch);
134   free(pathUltimate);
135   free(pathPenultimate);
136
137   /* Open input file. */
138
139   fileInput = fopen(fileInputStr, "r");
140   if (fileInput == NULL){
141     printf("Input␣file␣missing␣or␣name␣not␣matched!");
142     exit(1);
143   }
```

```
144     free(fileInputStr);
145
146     /* Looks as if we have what we need; start the clock. */
147     timeStart = time (NULL);
148
149     /* Read some input parameters. */
150
151     fscanf(fileInput, "NC %d \
152 ␣␣␣␣␣␣␣␣␣␣NL %d \
153 ␣␣␣␣␣␣␣␣␣␣ND %d \
154 ␣␣␣␣␣␣␣␣␣␣learn %d \
155 ␣␣␣␣␣␣␣␣␣␣minV %d \
156 ␣␣␣␣␣␣␣␣␣␣learnMultiple %d \
157 ␣␣␣␣␣␣␣␣␣␣totalReps %d \
158 ␣␣␣␣␣␣␣␣␣␣capacity %d \
159 ␣␣␣␣␣␣␣␣␣␣penaltyThreshold %d \
160 ␣␣␣␣␣␣␣␣␣␣invDistPr %lf \
161 ␣␣␣␣␣␣␣␣␣␣interdemandMean %lf \
162 ␣␣␣␣␣␣␣␣␣␣learnRate %lf \
163 ␣␣␣␣␣␣␣␣␣␣penalty %lf ",
164             &NC, &NL, &ND, &learn, &minV, &learnMultiple, &
                totalReps, &capacity, &penaltyThreshold, &invDistPr,
                 &interdemandMean, &learnRate, &penalty);
165
166     /* Specify the number of policy types, number of events for the
          timing function, and number of scheduling modes. */
167     NE = 4;
168     NM = 2;
169     NS = (capacity+1);
170     Ax = NC*NL + 1;
171
172     if(learn > 1){  /* Conditional thresholds will be learnt and
          tested. */
173       NP = 3;
174       for (i = 1; i < NL; i++)
175         NS = NS*(capacity+1);
176     }
177     else  /* Only the unconditional threshold will be learnt and
          tested. */
178       NP = 1;
179
180     if(learn > 0){
181       filePolicy = fopen(filePolicyStr, "w");
182       free(filePolicyStr);
183     }
184
185     /* Create arrays necessary to read the remaining input
```

```
      parameters. */
186
187   demandDist = calloc(NC, sizeof(float));
188     if (demandDist == NULL) exit(1);
189
190   potentialRewards = calloc(NC, sizeof(double));
191     if (potentialRewards == NULL) exit(1);
192
193   rejectPr = calloc(NL, sizeof(float));
194     if (rejectPr == NULL) exit(1);
195
196   leadTimes = malloc(NC*sizeof(int*));
197     if (leadTimes == NULL) exit(1);
198   for (i = 0; i < NC; i++)
199     leadTimes[i] = malloc(NL*sizeof(int));
200
201   userPolicy = malloc(NC*sizeof(int*));
202     if (userPolicy == NULL) exit(1);
203   for (i = 0; i < NC; i++)
204     userPolicy[i] = malloc(NL*sizeof(int));
205
206   /* Read remaining input parameters. */
207
208   fscanf(fileInput, "␣classDist␣");
209   for (i = 0; i < NC; i++){
210     fscanf(fileInput, "␣%f␣", &demandDist[i]);
211   }
212
213   fscanf(fileInput, "␣potentialRewards␣");
214   for (i = 0; i < NC; i++){
215     fscanf(fileInput, "␣%lf␣", &potentialRewards[i]);
216   }
217
218   fscanf(fileInput, "␣rejectPr␣");
219   for (i = 0; i < NL; i++){
220     fscanf(fileInput, "␣%f␣", &rejectPr[i]);
221   }
222
223   fscanf(fileInput, "␣leadTimes␣");
224   for (i = 0; i < NC; i++){
225     for (j = 0; j < NL; j++){
226     fscanf(fileInput, "␣%d␣", &leadTimes[i][j]);
227     }
228   }
229
230   fscanf(fileInput, "␣userPolicy␣");
231   for (i = 0; i < NC; i++){
```

```
232       for (j = 0; j < NL; j++){
233       fscanf(fileInput, " %d ", &userPolicy[i][j]);
234       }
235     }
236     fclose(fileInput);
237
238     /* Create rest of the arrays. */
239
240     stateVars.state = malloc(NP*sizeof(int**));
241       if (stateVars.state == NULL) exit(1);
242     for (i = 0; i < NP; i++){
243       stateVars.state[i] = malloc((NC+8)*sizeof(int*));
244       if (stateVars.state[i] == NULL) exit(1);
245       for (j = 0; j < NC+8; j++)
246         stateVars.state[i][j] = malloc(Ax*sizeof(int));
247     }
248
249     stateVars.reward = malloc(NP*sizeof(double*));
250       if(stateVars.reward == NULL) exit (1);
251     for (i = 0; i < NP; i++)
252       stateVars.reward[i] = malloc(Ax*sizeof(double));
253
254     stat.Q = malloc(NP*sizeof(double****));
255       if (stat.Q == NULL) exit(1);
256     for (i = 0; i < NP; i++){
257       stat.Q[i] = malloc((NC*NL)*sizeof(double***));
258       if (stat.Q[i] == NULL) exit(1);
259       for (j = 0; j < NC*NL; j++){
260         stat.Q[i][j] = malloc(NS*sizeof(double**));
261         if (stat.Q[i][j] == NULL) exit(1);
262         for (k = 0; k < NS; k++){
263           stat.Q[i][j][k] = malloc(NM*sizeof(double*));
264           if (stat.Q[i][j][k] == NULL) exit(1);
265           for (l = 0; l < NM; l++)
266             stat.Q[i][j][k][l] = malloc((NL+1)*sizeof(double));
267         }
268       }
269     }
270
271     stat.V = malloc(NP*sizeof(double****));
272       if (stat.V == NULL) exit(1);
273     for (i = 0; i < NP; i++){
274       stat.V[i] = malloc((NC*NL)*sizeof(double***));
275       if (stat.V[i] == NULL) exit(1);
276       for (j = 0; j < NC*NL; j++){
277         stat.V[i][j] = malloc(NS*sizeof(double**));
278         if (stat.V[i][j] == NULL) exit(1);
```

```
279         for (k = 0; k < NS; k++){
280           stat.V[i][j][k] = malloc(NM*sizeof(double*));
281           if (stat.V[i][j][k] == NULL) exit(1);
282           for (l = 0; l < NM; l++)
283             stat.V[i][j][k][l] = malloc(NL*sizeof(double));
284         }
285       }
286     }
287
288     repData = calloc(totalReps, sizeof(double));
289       if (repData == NULL) exit(1);
290
291     refState = malloc(NP*sizeof(int));
292       if (refState == NULL) exit(1);
293
294     refState[0] = (int) round((double) demandDist[NC-1]/(NL*
          interdemandMean));
295     if(NP > 1){
296       refState[1] = (int) round((double) ((capacity+2)*demandDist[
            NC-1])/(NL*interdemandMean));
297       refState[2] = refState[1];
298     }
299     for (i = 0; i < NP; i++)
300       printf("refState␣%d:␣%d\n", i, refState[i]);
301
302     /* Set simulation cycle. */
303
304     if (learn == 1){  /* Learn and test the unconditional threshold
          policy. */
305       start = 0;      /* Also test a fixed policy specified by
            userPolicy. */
306       end = 2;
307     }
308     else if (learn == 2){  /* Learn and test all policies (
          unconditional and conditional thresholds). */
309       start = 0;           /* Also test a fixed policy specified by
            userPolicy. */
310       end = 4;
311     }
312     else{          /* (i.e., if learn == 0) test only a fixed
          policy specified by userPolicy. */
313       start = 1;
314       end = 1;
315     }
316
317     for (simMode = start; simMode <=end; simMode++){
318
```

```
319       if(simMode == 0)
320         maxRep = 1;
321       else
322         maxRep = totalReps;
323
324       for(rep = 0; rep < maxRep; rep++){
325
326         /* Initialize the simulation. */
327         initialize(&stateVars, &stat, rep);
328
329         /* Run the simulation until it terminates after an end-
               simulation event (type 4) occurs. */
330         do {
331           timing();
332
333           /* Invoke the appropriate event function. */
334           switch (nextEventType) {
335             case 1:
336               printf("Case 1\n");
337               break;
338             case 2:
339               if(simMode == 0){
340                 for (i = 0; i < NP; i++)
341                   demand(&stateVars, &stat, 1, i);
342               }
343               else
344                 demand(&stateVars, &stat, 1, simMode - 2);
345               timeNextEvent[2] = simTime + exponVariate(1,
                     interdemandMean);
346               demandType = nextDemandType();
347               break;
348             case 3:
349               end_day(&stateVars, &stat, rep);
350               break;
351             case 4:
352               conclude(&stat, rep);
353               break;
354           }
355         } while (nextEventType != 4);
356       }
357     }
358     return 0;
359   }
360
361   void initialize(stcState* stateVars, stcStat* stat, int rep){  /*
          Initialization function. */
362
```

```
363   int i, j, k, classDay, load, scheduleMode, action, policy;
364   double offset;
365
366   offset = 1e7;
367
368   /* Initialize the RNG. */
369   if(simMode == 0){
370     zrng[0] = 1383670351;  /* Rejection evaluation. */
371     zrng[1] = 1985072130;  /* Interdemand times. */
372     zrng[2] = 748932582;  /* Scheduling event. */
373     zrng[3] = 1631331038;  /* Demand types. */
374     zrng[4] = 443952721;  /* Inventory disruption event. */
375     zrng[5] = 1848090842;  /* Magnitude of inventory disruption.
          */
376   }
377   else{
378     zrng[0] = 1848090842 + rep*offset;  /* Rejection evaluation.
          */
379     zrng[1] = 443952721 + rep*offset;  /* Interdemand times. */
380     zrng[2] = 1631331038 + rep*offset;  /* Scheduling event. */
381     zrng[3] = 748932582 + rep*offset;  /* Demand types. */
382     zrng[4] = 1985072130 + rep*offset;  /* Inventory disruption
          event. */
383     zrng[5] = 1383670351 + rep*offset;  /* Magnitude of inventory
           disruption. */
384   }
385
386   /* Initialize the simulation clock. */
387   simTime = 0.0;
388
389   /* Initialize the state variables. */
390   for (k = 0; k < NP; k++){
391     for (j = 0; j < Ax; j++){
392       for (i = 0; i <= NC; i++){
393         stateVars->state[k][i][j] = 0;
394       }
395       stateVars->state[k][NC+1][j] = j;
396       stateVars->state[k][NC+2][j] = j;
397       stateVars->state[k][NC+3][j] = 0;
398       stateVars->state[k][NC+4][j] = 0;
399       stateVars->state[k][NC+5][j] = 0;
400       stateVars->state[k][NC+6][j] = 0;
401       stateVars->state[k][NC+7][j] = 0;
402       stateVars->reward[k][j] = 0;
403     }
404   }
405
```

```
406   if (simMode == 0){
407     /* Initialize learning variables. */
408     for(load=0;load < NS;load++){  /* Only the first (capacity +
          1) entries will be used for policy type 0. */
409       for(scheduleMode = 0; scheduleMode < NM; scheduleMode++){
410         for(classDay=0;classDay<NC*NL;classDay++){  /* Only the
              first NC entries will be used for policy type 2. */
411           for(policy = 0; policy < NP; policy++){
412             for(action = 0; action < NL; action++){
413               stat->Q[policy][classDay][load][scheduleMode][
                    action]=0;
414               stat->V[policy][classDay][load][scheduleMode][
                    action]=0;
415             }
416             stat->Q[policy][classDay][load][scheduleMode][NL
                  ]=-1.0;  /* These are the elements that will hold
                  the learnt policies */
417           }
418         }
419         /* Go back and the Q-factor for reject to SMALL in policy
              types 0 and 1, in order to ensure it is not the
             policy conclusion. */
420         for(classDay = 0; classDay < NC*NL; classDay++){
421           for(policy = 0; policy < (NP > 1 ? 2 : 1); policy++){
422             if((classDay+1)%NL == 0)  /* The final day for each
                    class has only one action: accept. */
423               stat->Q[policy][classDay][load][scheduleMode][0]=
                      SMALL;
424           }
425         }
426       }
427     }
428   }
429
430   /* Initialize global statistical variables.
431   The first group are initialized for each replication. */
432   currentDay              = 0;
433   currentDayIndex           = 0;
434   repDemands              = 0;
435   repSuccesses            = 0;
436   repRejects              = 0;
437   repLosses             = 0;
438   repReschedules            = 0;
439   repRescheduleLosses         = 0;
440   repReward             = 0.0;
441   repPotentialReward        = 0.0;
442   stat->prevClass           = -1;
```

```
443     stat->prevArrival          = -1;
444     stat->prevState            = -1;
445     stat->prevAct            = -1;
446     stat->currentState          = -1;
447     stat->currentAction          = -1;
448     stat->currentReward          = -1.0;
449     stat->prevReward          = -1.0;
450
451     /* The second group are summed across all replications. */
452     if (rep == 0){
453       totalDemands          = 0;
454       totalSuccesses          = 0;
455       totalRejects          = 0;
456       totalLosses            = 0;
457       totalRescheduled        = 0;
458       totalRescheduleLosses      = 0;
459       totalExcesses          = 0;
460     }
461
462     /* Initialize the event list. */
463     timeNextEvent[1] = 1.0e+30;     /* Next order arrival time. */
464     timeNextEvent[2] = simTime + exponVariate(1, interdemandMean);
          /* Next demand arrival time ... */
465     demandType = nextDemandType();              /* ... and type. */
466     timeNextEvent[3] = 1.0;       /* End of first day. */
467     if(simMode > 0)
468       timeNextEvent[4] = (double) ND;     /* End of simulation run.
          */
469     else
470       timeNextEvent[4] = (double) ND * learnMultiple;     /* End of
          simulation run. */
471 }
472
473 void timing(){  /* Timing function. */
474
475   int i;
476   double min_timeNextEvent = 1.0e+29;
477
478   nextEventType = 0;
479
480   /* Determine the next event type. */
481   for (i = 1; i <= NE; i++){
482     if (timeNextEvent[i] < min_timeNextEvent){
483       min_timeNextEvent = timeNextEvent[i];
484       nextEventType = i;
485     }
486   }
```

```
487
488   /* Check whether the event list is empty. */
489   if (nextEventType == 0) {
490     /* The event list is empty, so stop the simulation */
491     printf("\nEvent list empty at time %f\n", simTime);
492     exit(1);
493   }
494
495   /* The event list is not empty, so advance the simulation clock
        . */
496   simTime = min_timeNextEvent;
497 }
498
499 int schedule(stcState* stateVars, stcStat* stat, int
      intNewArrival, int policyIndex){
500
501   int i, multiplier, baseLT, load, stateVal, returnVal;
502   double var, incr;
503
504   returnVal = NL - 1;  /* Set the return value initially at the
        maximum lead time. */
505   baseLT = leadTimes[demandType][0];
506   if(policyIndex > 0){
507     stateVal = 0;
508     multiplier = 1;
509     for (i = 0; i < NL; i++){
510       /* This construction entails that days corresponding to
            longer lead times contribute less to stateVal. */
511       stateVal = stateVal + multiplier*stateVars->state[
            policyIndex][NC][(currentDayIndex + baseLT + returnVal -
             i)%Ax];
512       multiplier = multiplier*(capacity + 1);
513     }
514   }
515
516   if(simMode ==0){
517     var = uniformVariate(2, 0, 1);
518     incr = (double) 1/NL;
519     stat->currentState = stateVal;
520
521     for (i = 0; i < NL; i++){
522       if(policyIndex > 0)
523         stateVars->state[policyIndex][NC+7][(currentDayIndex +
              baseLT + i)%Ax] = stateVal;
524       else
525         stateVars->state[policyIndex][NC+7][(currentDayIndex +
              baseLT + i)%Ax] = stateVars->state[policyIndex][NC][(
```

```
            currentDayIndex + baseLT + i)%Ax];
526       multiplier = i;
527       if (var <= (multiplier+1)*incr){
528         returnVal = i;
529         break;
530       }
531     }
532   }
533   else if(simMode > 1 && simMode < 4){  /* simMode == 2 or
        simMode == 3. */
534     for(i = baseLT; i < baseLT + NL - 1; i++){
535       load = stateVars->state[policyIndex][NC][(currentDayIndex +
            i)%Ax];
536       if(simMode == 2)
537         stateVal = load;
538       if(load < capacity){
539         if(stat->Q[policyIndex][i - 1][stateVal][intNewArrival][
              NL] == 1.0){
540           returnVal = i - baseLT;
541           break;
542         }
543       }
544     }
545   }
546   else if(simMode == 4){
547     returnVal = (int) stat->Q[policyIndex][demandType][stateVal][
          intNewArrival][NL];
548   }
549   else{  /* simMode == 1. */
550     for(i = baseLT; i < baseLT + NL - 1; i++){
551       load = stateVars->state[policyIndex][NC][(currentDayIndex +
            i)%Ax];
552       if(intNewArrival){
553         if((load < userPolicy[demandType][i - baseLT]) && (load <
              capacity)){
554           returnVal = i - baseLT;
555           break;
556         }
557       }
558       else{
559         if(load < capacity){
560           returnVal = i - baseLT;
561           break;
562         }
563       }
564     }
565   }
```

```
566   return returnVal;
567 }
568
569 void demand(stcState* stateVars, stcStat* stat, int intNewArrival
      , int policyIndex){  /* Demand event function. */
570
571   int i, intOffer, intOfferIndex, intLTO, intDecision;
572   double dblReward, divisor, dblScheduleReward;
573
574   if(simMode > 0 && policyIndex < 0)  /* We are testing a fixed
        simple threshold policy, */
575     policyIndex = 0;
576
577   /* Determine offset from most favorable lead time */
578   intLTO = schedule(stateVars, stat, intNewArrival, policyIndex);
579
580   intOffer = currentDay + leadTimes[demandType][0] + intLTO;
581   intOfferIndex = intOffer%Ax;
582
583   if(stateVars->state[policyIndex][NC][intOfferIndex] < capacity)
        {
584     if(intNewArrival)
585       intDecision = makeOffer(intOffer);
586     else
587       intDecision = 1;
588
589     dblReward = getReward(stateVars, policyIndex, intDecision,
          intOfferIndex, intLTO, intNewArrival);
590   }
591   else{
592     intDecision = -1;
593     if(intNewArrival)
594       dblReward = 0;
595     else{
596       if(simMode > 0)
597         printf("Lost␣demand␣on␣rescheduling.\n");
598       dblReward = (double) - (NL+1)*(NC - demandType);
599     }
600   }
601
602   if(simMode > 0 && stateVars->state[policyIndex][NC+1][
        intOfferIndex] >= Ax){
603     if(intNewArrival){
604       repDemands++;
605       repPotentialReward = repPotentialReward + potentialRewards[
            demandType];
606       if(intDecision == 1)
```

```
607           repSuccesses++;
608         else if (intDecision == 0)
609           repRejects++;
610         else
611           repLosses++;
612       }
613       else{
614         if(intDecision == 1)
615           repReschedules++;
616         else
617           repRescheduleLosses++;
618       }
619
620       repReward = repReward + dblReward;
621     }
622
623     if(simMode == 0){
624       if(policyIndex == 2)
625         learnRecord(stateVars, stat, policyIndex, intOffer,
              intNewArrival, intLTO, dblReward);
626       else{
627         if(intNewArrival)
628           dblScheduleReward = dblReward;
629         else
630           dblScheduleReward = (NL+1)*(NC - demandType) + dblReward;
631
632         if(intLTO == 0)
633           learnRecord(stateVars, stat, policyIndex, intOffer,
                intNewArrival, 1, dblScheduleReward);
634         else{
635           divisor = (double) intLTO + 1;
636           for(i = -intLTO; i<=-1;i++)
637             learnRecord(stateVars, stat, policyIndex, intOffer+i,
                    intNewArrival, 0, dblScheduleReward/divisor);
638           learnRecord(stateVars, stat, policyIndex, intOffer,
                intNewArrival, 1, dblScheduleReward/divisor);
639         }
640       }
641     }
642
643     if(intDecision > 0)  /* intDecision is 1 if customer accepts or
          a rescheduling succeeds. */
644       updateState(stateVars, policyIndex, intOfferIndex);
645   }
646
647   void end_day(stcState* stateVars, stcStat* stat, int rep){  /*
        End of day event function. */
```

```
648
649     int i, policyStart, policyEnd, policyIndex, intToReschedule,
            intMaxReschedule;
650
651     intMaxReschedule = 0;
652
653     if(uniformVariate(4, 0, 1) < invDistPr)
654       intMaxReschedule = (int) round(penaltyThreshold*
              uniformVariate(5, 0, 1));
655
656     if(simMode == 0){
657       policyStart = 0;
658       policyEnd = NP - 1;
659     }
660     else if(simMode == 1){
661       policyStart = 0;
662       policyEnd = 0;
663     }
664     else{
665       policyStart = simMode - 2;
666       policyEnd = simMode - 2;
667     }
668
669     for (policyIndex = policyStart; policyIndex <= policyEnd;
            policyIndex++){
670
671       intToReschedule = 0;
672
673       /* Determine the extent of the capacity disruption, if any.
              */
674       if(intMaxReschedule > 0){
675         if(stateVars->state[policyIndex][NC][(currentDayIndex+1)%Ax
                ] > intMaxReschedule)
676           intToReschedule = intMaxReschedule;
677         else
678           intToReschedule = stateVars->state[policyIndex][NC][(
                  currentDayIndex+1)%Ax];
679       }
680
681       /* End-of-day learning for policies 0 & 1.  CHECK: LAST
              CONDITION MAY BE OMITTED. */
682       if(simMode == 0 && policyIndex < 2 && stateVars->state[
              policyIndex][NC+1][(currentDayIndex+1)%Ax] >= Ax &&
              stateVars->state[policyIndex][NC+5][(currentDayIndex+1)%Ax
              ] > -1)
683         rqlearn(stateVars, stat, 1, (currentDayIndex+1)%Ax, 1,
                policyIndex);  /* The penultimate 0 means the arrival
```

```
            argument will not be used in rqlearn. */
684
685       /* Reset the loads for the current day (which will become
              furthest visible day). */
686       for (i = 0; i <= NC; i++)
687         stateVars->state[policyIndex][i][currentDayIndex] = 0;
688
689       /* Add the date for the furthest visible day. */
690       stateVars->state[policyIndex][NC+1][currentDayIndex] += Ax;
691
692       /* The furthest visible day starts with no last action. */
693       stateVars->state[policyIndex][NC+5][currentDayIndex] = -1;
694
695       /* Decrement all indicators of days remaining to production.
              */
696       for (i = currentDayIndex; i <= currentDayIndex + (Ax-1); i++)
697         stateVars->state[policyIndex][NC+2][i%Ax] = (stateVars->
              state[policyIndex][NC+2][i%Ax] + Ax - 1)%Ax;
698
699       /* Advance the current day and corresponding index for
              rescheduling. */
700       currentDay++;
701       currentDayIndex = currentDay%Ax;
702
703       /* Reschedule as needed from the new current day. */
704       if(intToReschedule)
705         reschedule(stateVars, stat, intToReschedule, policyIndex);
706
707       /* Reset the current day and corresponding index for further
              loops. */
708       currentDay--;
709       currentDayIndex = currentDay%Ax;
710     }
711
712     /* Advance the current day and corresponding index definitively
          . */
713     currentDay++;
714     currentDayIndex = currentDay%Ax;
715
716     /* Print a statement of progress. */
717     if(currentDay%10000 == 0)
718       printf("simMode␣%d,␣rep␣%d,␣step␣%d\n", simMode, rep,
              currentDay/10000);
719
720     /* Set the time of the next end-of-day event. */
721     timeNextEvent[3] = simTime + 1.0;
722   }
```

```
723
724 void reschedule(stcState* stateVars, stcStat* stat, int
      intToReschedule, int policyIndex){
725
726   int i, j, nextArrivalType, reschedulingType, *currentLoads, *
        classNeeds;
727
728   currentLoads = calloc(NC, sizeof(int));
729   if (currentLoads == NULL) exit(1);
730
731   classNeeds = calloc(NC, sizeof(int));
732   if (classNeeds == NULL) exit(1);
733
734   nextArrivalType = demandType;  /* Record this in order to be
        able to reset it when rescheduling is finished. */
735   j = 0;  /* j will track the cumulative number of demands
        rescheduled as we proceed through the classes. */
736   reschedulingType = 0;
737
738   for (i = 0; i < NC; i++)
739     currentLoads[i] = stateVars->state[policyIndex][i][
          currentDayIndex];
740
741   for (i = NC-1; i >= 0; i--){
742     if(intToReschedule - j > currentLoads[i]){
743       classNeeds[i] = currentLoads[i];
744       j = j + currentLoads[i];
745     }
746     else{
747       classNeeds[i] = intToReschedule - j;
748       break;
749     }
750   }
751
752   for (i = 0; i < NC; i++){
753     if(i > 0)
754         reschedulingType = i;
755     do{
756       demandType = reschedulingType;
757       if(simMode != 1)
758         demand(stateVars, stat, 0, policyIndex);
759       else
760         demand(stateVars, stat, 0, simMode - 1);
761       classNeeds[i]--;
762     } while (classNeeds[i] > 0);
763   }
764   demandType = nextArrivalType;
```

```
765     free(currentLoads);
766     free(classNeeds);
767   }
768
769   void decodeState(int state, int* stateLevels, int policyIndex){
770
771     int i, modulus;
772     double divisor;
773     divisor = 1;
774     modulus = capacity + 1;
775
776     /* See conclude() for construction of the state input. */
777     if(policyIndex == 0){
778       stateLevels[0] = ((int) floor((double)state / (double)(NC*NL
            *(capacity + 1))));     /* Schedule mode */
779       stateLevels[1] = ((int) floor((double)state / (double)(
            capacity + 1)))%(NC*NL);     /* Class index */
780       stateLevels[2] = state%(capacity + 1);
                                    /* Simple state */
781     }
782     else if(policyIndex == 1){
783       stateLevels[0] = ((int) floor((double)state / (double)(NC*NL*
            NS)));     /* Schedule mode */
784       stateLevels[1] = ((int) floor((double)state / (double)(NS)))
            %(NC*NL);  /* Class index */
785       stateLevels[2] = state%NS;                          /*
            Generalized state */
786     }
787     else{
788       stateLevels[0] = ((int) floor((double)state / (double)(NC*NS)
            ));     /* Schedule mode */
789       stateLevels[1] = ((int) floor((double)state / (double)(NS)))
            %(NC);  /* Class index */
790       stateLevels[2] = state%NS;      /* Generalized state */
791     }
792
793     /* For generalized states, determine the specific load levels
          for each target
794     day within the class; specifically, store the load level for
          target day
795     NL - 1 - i in stateLevels[3 + NL -1 - i] (larger index for
          longer lead time). */
796
797     if(policyIndex > 0){
798       for(i = 0; i < NL; i++){
799         stateLevels[3 + (NL-1) - i] = ((int) floor((double)
              stateLevels[2] / divisor))%modulus;
```

```
800         divisor = divisor*modulus;
801       }
802     }
803     else{
804       for(i = 0; i < NL; i++)
805         stateLevels[3 + i] = 0;
806     }
807     return;
808 }
809
810 void conclude(stcStat* stat, int rep){  /* Get policy or report
        results. */
811
812     int i, policyIndex, genState, numGenStates,
          currentClassDayIndex, currentScheduleMode, intTestDuration,
          *currentDayLoads, *stateLevels;
813     double action, qCurrent, dblAverageReward, dblRewardVariance,
          ciw, expectedDemandValue;
814     time_t timeCurrent;
815
816     stateLevels = calloc(3+NL, sizeof(int));
817       if (stateLevels == NULL) exit(1);
818
819     if(simMode == 0){
820       for(policyIndex = 0; policyIndex < NP; policyIndex++){
821
822         fprintf(filePolicy, "\n POLICY %d\n\n", policyIndex);
823
824         /* If policyIndex > 0 and NL > 2, i.e., there are more than
              two possible lead times for each class and it's not a
              single threshold policy, then stateLevels we will
              indicate the loads on each possible target day.  In
              order to print the policy, we need to know when we
              increment the load on any except the last two target
              days.  The currentDayLoads array will contain the
              relevant load information. */
825
826         if(policyIndex > 0 && NL > 2){
827           currentDayLoads = calloc(NL-2, sizeof(int));
828           if (currentDayLoads == NULL) exit(1);
829         }
830         else
831           currentDayLoads = calloc(1, sizeof(int));
832
833         if(policyIndex == 0)
834           numGenStates = NC*NL * (capacity + 1) * NM;
835         else if(policyIndex == 1)
```

```
836           numGenStates = NC*NL * NS * NM;
837         else
838           numGenStates = NC * NS * NM;
839
840         currentScheduleMode = 0;
841         currentClassDayIndex = 0;
842
843         fprintf(filePolicy, "Current schedule mode = %d\n",
              currentScheduleMode);
844         fprintf(filePolicy, "Days to production: %d\n",
              currentClassDayIndex + 1);
845
846         if(policyIndex > 0 && NL > 2){
847           for(i = 0; i < NL-2; i++){
848             currentDayLoads[i] = 0;
849             fprintf(filePolicy, "Target day %d load = %d\n", i+1,
                    currentDayLoads[i]);
850           }
851         }
852
853         for(i = 0; i <= capacity; i++)
854           fprintf(filePolicy, "%d\t", i);
855         fprintf(filePolicy, "\n");
856
857         for(genState = 0; genState < numGenStates; genState++){
858           decodeState(genState, stateLevels, policyIndex);
859
860           if(stateLevels[0] != currentScheduleMode){  /* Note a
                change of schedule mode. */
861             currentScheduleMode = stateLevels[0];
862             fprintf(filePolicy, "\n\n\n Current schedule mode = %d"
                    , currentScheduleMode);
863           }
864           if(stateLevels[1] != currentClassDayIndex){  /* Note a
                change of class day. */
865             currentClassDayIndex = stateLevels[1];
866             if(policyIndex < 2)
867               fprintf(filePolicy, "\n Days to production: %d\n",
                      currentClassDayIndex + 1);
868             else
869               fprintf(filePolicy, "\n Demand Class: %d\n",
                      currentClassDayIndex + 1);
870           }
871           if(policyIndex > 0 && NL > 2){  /* Note any changes in
                earlier target day loads. */
872             for(i = 0; i < NL-2; i++){
873               if(currentDayLoads[i] != stateLevels[3+i]){
```

```
874                 currentDayLoads[i] = stateLevels[3+i];
875                 fprintf(filePolicy, "Target day %d load = %d\n", i
                        +1, currentDayLoads[i]);
876               }
877             }
878           }
879
880           if(policyIndex < 2){
881             if(stat->V[policyIndex][stateLevels[1]][stateLevels
                    [2]][stateLevels[0]][1] > minV){
882               if(stat->Q[policyIndex][stateLevels[1]][stateLevels
                      [2]][stateLevels[0]][0] > stat->Q[policyIndex][
                      stateLevels[1]][stateLevels[2]][stateLevels
                      [0]][1])
883                 action = 0.0;
884               else
885                 action = 1.0;
886             }
887             else if(policyIndex == 0){
888               if(stateLevels[2] > 0)
889                 action = stat->Q[policyIndex][stateLevels[1]][
                        stateLevels[2]-1][stateLevels[0]][NL] + 0.2/(
                        stateLevels[1] + 1);
890               else
891                 action = 1.0;
892             } /* policyIndex > 0 && visits <= minV */
893             else
894             /* Q-factor results for policy 1 are inconclusive at
                    this point, so substitute the simple threshold (
                    policy type 0) action at the load of the
                    currentClassDayIndex. The load of
                    currentClassDayIndex is given by stateLevels[3 +
                    stateLevels[1]%NL]. */
895               action = stat->Q[0][stateLevels[1]][stateLevels[3 +
                      stateLevels[1]%NL]][stateLevels[0]][NL] + 0.2;
896
897             stat->Q[policyIndex][stateLevels[1]][stateLevels[2]][
                    stateLevels[0]][NL] = floor(action);
898           }
899           else{
900             if(stat->V[policyIndex][stateLevels[1]][stateLevels
                    [2]][stateLevels[0]][1] > minV){
901               qCurrent = SMALL;
902               for(i = 0; i < NL; i++){
903                 if(stat->Q[policyIndex][stateLevels[1]][stateLevels
                        [2]][stateLevels[0]][i] > qCurrent){
904                   action = (double) i;
```

```
905                     stat->Q[policyIndex][stateLevels[1]][stateLevels
                            [2]][stateLevels[0]][NL] = action;
906                     qCurrent = stat->Q[policyIndex][stateLevels[1]][
                            stateLevels[2]][stateLevels[0]][i];
907                   }
908                 }
909               }
910               else{
911               /* Q-factor results for policy 2 are inconclusive at
                     this point, so substitute the first day that has an
                     acceptance action for this class in the simple
                     threshold (policy type 0). Note that stateLevels[1]
                      gives the class in the context of policy type 2, so
                      use the looping index i for the corresponding
                     classDayIndex in policy 0. */
912                 for(i = leadTimes[stateLevels[1]][0] - 1; i <=
                        leadTimes[stateLevels[1]][NL - 1] - 1; i++){
913                   if(stat->Q[0][i][stateLevels[3 + i%NL]][stateLevels
                          [0]][NL] == 1.0){
914                     action = (double) (i%NL) + 0.2;
915                     break;
916                   }
917                 }
918                 stat->Q[policyIndex][stateLevels[1]][stateLevels[2]][
                        stateLevels[0]][NL] = floor(action);
919               }
920             }
921
922             if(action == floor(action))
923               fprintf(filePolicy,"%1.0f\t", action);
924             else
925               fprintf(filePolicy,"%1.0f*\t", action);
926
927             /* Besides the work done above to distinguish policy
                   regions by load on earlier target days, we still need
                   to put the printed policy for the last two target days
                    in matrix form. Start a new line every time policy
                   values for (capacity + 1) generalized states have been
                    printed. */
928
929             if((stateLevels[2]+1)%(capacity+1) == 0)
930               fprintf(filePolicy, "\n");
931           }
932         }
933       fclose(filePolicy);
934       free(currentDayLoads);
935
```

```
936     timeCurrent = time (NULL);
937     intLearnDuration = timeCurrent - timeStart;
938     timeStart = timeCurrent;
939     }
940     else{
941       repData[rep] = repReward/repPotentialReward;
942       totalDemands = totalDemands + repDemands;
943       totalSuccesses = totalSuccesses + repSuccesses;
944       totalRejects = totalRejects + repRejects;
945       totalLosses = totalLosses + repLosses;
946       totalRescheduled = totalRescheduled + repReschedules;
947       totalRescheduleLosses = totalRescheduleLosses +
            repRescheduleLosses;
948       printf("T: %d, S: %d, R: %f\n", repDemands, repSuccesses,
            repReward);
949       printf("Avg. reward: %f\n", repData[rep]);
950       if(rep == totalReps - 1){
951
952         expectedDemandValue = 0;
953         dblAverageReward = 0;
954         dblRewardVariance = 0;
955
956         for(i = 0; i < NC; i++){
957           expectedDemandValue = expectedDemandValue + ((double)
                demandDist[i])*potentialRewards[i];
958         }
959
960         for(i = 0; i < totalReps; i++){
961           dblAverageReward = dblAverageReward + repData[i];
962         }
963         dblAverageReward = dblAverageReward/totalReps;
964
965         for(i = 0; i < totalReps; i++){
966           dblRewardVariance = dblRewardVariance + pow(repData[i]-
                dblAverageReward,2);
967         }
968         dblRewardVariance = dblRewardVariance/((double) rep);
969         ciw = 2.009574018*sqrt(dblRewardVariance/totalReps);
970
971         timeCurrent = time (NULL);
972         intTestDuration = timeCurrent - timeStart;
973         timeStart = timeCurrent;
974
975         fileResults = fopen("results.txt", "a");
976
977         fprintf(fileResults, "%1.3f\t", expectedDemandValue);
978         fprintf(fileResults, "%d\t", NC);
```

```
979         fprintf(fileResults, "%d\t", NL);
980         fprintf(fileResults, "%3.0f\t", 1/interdemandMean);
981         fprintf(fileResults, "%d\t", capacity);
982         fprintf(fileResults, "%d\t", penaltyThreshold);
983         fprintf(fileResults, "%2.3f\t", penalty);
984         fprintf(fileResults, "%2.3f\t", invDistPr);
985         fprintf(fileResults, "%d\t", simMode - 1);
986         fprintf(fileResults, "%d\t", learnMultiple);
987         fprintf(fileResults, "%d\t", intLearnDuration);
988         fprintf(fileResults, "%d\t", intTestDuration);
989         fprintf(fileResults, "%d\t", totalDemands);
990         fprintf(fileResults, "%f%%\t", ((double) totalSuccesses
               *100) / (double) totalDemands);
991         fprintf(fileResults, "%f%%\t", ((double) totalRejects*100)
               / (double) totalDemands);
992         fprintf(fileResults, "%f%%\t", ((double) totalLosses*100) /
                (double) totalDemands);
993         fprintf(fileResults, "%f%%\t", ((double) totalRescheduled
               *100) / (double) totalDemands);
994         fprintf(fileResults, "%f%%\t", ((double)
               totalRescheduleLosses*100) / (double) totalDemands);
995         fprintf(fileResults, "%f\t", dblRewardVariance);
996         fprintf(fileResults, "%2.6f␣%2.6f␣%2.6f\n",
               dblAverageReward - ciw, dblAverageReward,
               dblAverageReward + ciw);
997
998         fclose(fileResults);
999
1000        for(i = 0; i < totalReps; i++)
1001          repData[i] = 0;
1002      }
1003    }
1004    free(stateLevels);
1005  }
1006
1007  int nextDemandType(void){
1008
1009    int i, returnVal;
1010    double var;
1011    float cumProb;
1012    var = uniformVariate(3, 0, 1);
1013    cumProb = demandDist[0];
1014
1015    for(i=0;i<=NC-1;i++){
1016      if (var <= cumProb || i == NC-1){
1017        returnVal = i;
1018        break;
```

```
1019      }
1020      else
1021        cumProb = cumProb+demandDist[i+1];
1022    }
1023    return returnVal;
1024  }
1025
1026  int makeOffer(int testDay){
1027
1028    int i, returnVal;
1029    double rejectLevel;
1030    rejectLevel = uniformVariate(0, 0, 1);
1031    returnVal = 0;
1032
1033    for (i = 0; i < NL; i++){
1034      if ((currentDay + leadTimes[demandType][i] == testDay) && (
            rejectLevel > rejectPr[i]))
1035        returnVal = 1;
1036    }
1037    return returnVal;
1038  }
1039
1040  void updateState(stcState* stateVars, int policyIndex, int
       intOfferIndex){
1041
1042    stateVars->state[policyIndex][demandType][intOfferIndex] += 1;
1043    stateVars->state[policyIndex][NC][intOfferIndex] += 1;
1044  }
1045
1046  double getReward(stcState* stateVars, int policyIndex, int
       intDecision, int intOfferIndex, int intLTO, int intNewArrival)
       {
1047
1048    int intCurrentLoad;
1049    double dblReward;
1050    dblReward = 0;
1051
1052    if(intDecision){
1053      if(intNewArrival)
1054        dblReward = potentialRewards[demandType];
1055      else
1056        dblReward = (double) - (intLTO + 1)*(NC - demandType);
1057
1058      intCurrentLoad = stateVars->state[policyIndex][NC][
            intOfferIndex];
1059
1060      if(intCurrentLoad >= penaltyThreshold){
```

```
1061        dblReward = dblReward - penalty*(intCurrentLoad + 1 -
                penaltyThreshold);
1062        totalExcesses++;
1063      }
1064    }
1065
1066    return dblReward;
1067  }
1068
1069  void learnRecord(stcState* stateVars, stcStat* stat, int
        policyIndex, int intOffer, int intNewArrival, int intAction,
        double dblReward){
1070
1071    int intOfferIndex;
1072
1073    intOfferIndex = intOffer%Ax;
1074
1075    if(policyIndex < 2){
1076      if(stateVars->state[policyIndex][NC+1][intOfferIndex] >= Ax
            && stateVars->state[policyIndex][NC+5][intOfferIndex] >
            -1)
1077        rqlearn(stateVars, stat, intNewArrival, intOfferIndex, 0,
              policyIndex);
1078
1079      stateVars->state[policyIndex][NC+3][intOfferIndex] =
            stateVars->state[policyIndex][NC+2][intOfferIndex];
1080      stateVars->state[policyIndex][NC+4][intOfferIndex] =
            stateVars->state[policyIndex][NC+7][intOfferIndex];
1081      stateVars->state[policyIndex][NC+5][intOfferIndex] =
            intAction;
1082      stateVars->state[policyIndex][NC+6][intOfferIndex] =
            intNewArrival;
1083      stateVars->reward[policyIndex][intOfferIndex] = dblReward;
1084    }
1085    else{
1086      stat->currentAction = intAction;
1087      stat->currentReward = dblReward;
1088
1089      if (stateVars->state[policyIndex][NC+1][intOfferIndex] >= Ax
            && stat->prevAct > -1)
1090        rqlearn(stateVars, stat, intNewArrival, intOfferIndex, 0,
              policyIndex); /* intOfferIndex is not used by policy
              type 2 */
1091
1092      stat->prevClass = demandType;
1093      stat->prevArrival = intNewArrival;
1094      stat->prevState = stat->currentState;
```

```
1095        stat->prevAct = stat->currentAction;
1096        stat->prevReward = stat->currentReward;
1097      }
1098    }
1099
1100    void rqlearn(stcState* stateVars, stcStat* stat, int
          intNewArrival, int intRecordIndex, int terminal, int
          policyIndex){  /* Relative Q-Learning */
1101
1102      double qCurrent, qPrevious, learnWeight, rimm;
1103      int NA, action, visits, currentStateA, currentStateB,
            prevStateA, prevStateB, prevAction, prevArrivalType,
            distQstate;
1104
1105      if(policyIndex < 2){
1106        prevStateA = stateVars->state[policyIndex][NC+3][
              intRecordIndex] - 1;
1107        prevStateB = stateVars->state[policyIndex][NC+4][
              intRecordIndex];
1108        prevAction = stateVars->state[policyIndex][NC+5][
              intRecordIndex];
1109        prevArrivalType = stateVars->state[policyIndex][NC+6][
              intRecordIndex];
1110
1111        currentStateA = stateVars->state[policyIndex][NC+2][
              intRecordIndex] - 1;  /* Deduct 1 because currentStateA is
               day index for the Q-factor array. */
1112        currentStateB = stateVars->state[policyIndex][NC+7][
              intRecordIndex];       /* Generalized load. */
1113        rimm = stateVars->reward[policyIndex][intRecordIndex];
1114
1115        NA = 2;
1116      }
1117      else{
1118        prevStateA = stat->prevClass;
1119        prevStateB = stat->prevState;
1120        prevAction = stat->prevAct;
1121        prevArrivalType = stat->prevArrival;
1122
1123        currentStateA = demandType;
1124        currentStateB = stat->currentState;
1125        rimm = stat->prevReward;
1126
1127        NA = NL;
1128      }
1129
1130      if(terminal){
```

```
1131      rimm = stateVars->reward[policyIndex][intRecordIndex];
1132      qCurrent = 0;
1133    }
1134    else{ /* Terminal is always 0 for policy type 2. */
1135      qCurrent = SMALL;
1136      /* Find the best Q-factor in the current state */
1137      for(action = 0; action < NA; action++){
1138        if(stat->Q[policyIndex][currentStateA][currentStateB][
              intNewArrival][action] > qCurrent)
1139          qCurrent = stat->Q[policyIndex][currentStateA][
                currentStateB][intNewArrival][action];
1140      }
1141    }
1142
1143    qPrevious = stat->Q[policyIndex][prevStateA][prevStateB][
          prevArrivalType][prevAction];
1144
1145    stat->V[policyIndex][prevStateA][prevStateB][prevArrivalType][
          prevAction]++;
1146    visits = stat->V[policyIndex][prevStateA][prevStateB][
          prevArrivalType][prevAction];
1147    learnWeight = learnRate/visits;
1148
1149    distQstate = refState[policyIndex];
1150    /* Q[policyIndex][(NC-1)*NL][distQstate][1][1] is the
          distinguished factor */
1151
1152    if(policyIndex < 2)
1153      qPrevious = qPrevious * (1.0 - learnWeight) + (learnWeight*(
            rimm + qCurrent - stat->Q[policyIndex][(NC-1)*NL][
            distQstate][1][1]));
1154    else
1155      qPrevious = qPrevious * (1.0 - learnWeight) + (learnWeight*(
            rimm + qCurrent - stat->Q[policyIndex][NC-1][distQstate
            ][1][1]));
1156
1157    stat->Q[policyIndex][prevStateA][prevStateB][prevArrivalType][
          prevAction] = qPrevious;
1158
1159    return;
1160  }
1161
1162  double uniformVariate(int stream, float lower, float upper){  /*
        Uniform variate generation function. */
1163
1164    /* Return a U(a,b) random variate. */
1165    double variate;
```

```
1166    variate = lower + lcgrand(stream) * (upper - lower);
1167    return variate;
1168  }
1169
1170  double exponVariate(int stream, double mean){  /* Exponential
          random variate generation function. */
1171
1172    /* Return an exponential random variate with mean "mean". */
1173    double variate;
1174    variate = -mean * log(lcgrand(stream));
1175    return variate;
1176  }
1177
1178  /* Generate the next random number. */
1179  double lcgrand(int stream){
1180
1181    double hi, test, lo, seed;
1182
1183    seed  = zrng[stream];
1184    hi=floor(seed/q);
1185    lo=seed-q*hi;
1186    test=a*lo-r*hi;
1187
1188    if (test >0.0)
1189      seed=test;
1190    else
1191      seed=test+m;
1192
1193    zrng[stream] = seed;
1194    return seed/m;
1195  }
1196
1197  double round(double x){
1198    if (x + 0.5 > ceil(x))
1199      return ceil(x);
1200    else
1201      return floor(x);
1202  }
```

APPENDIX C

STATISTICAL RESULTS FOR CHAPTER 4

Description of Variables

All records for the following variables relate to a specific combination of plant and year. In some cases, the variable gives a performance measure from an earlier period or across a period of more than one year. These instances are noted.

Chassis
Number of different chassis configurations in production at the plant.

CumulativeProduction 01 (Launch plants only)
Cumulative production on the launch platform to end of the launch year.

CumulativeProduction 02 (Launch plants only)
Cumulative production on the launch platform to the end of the year after launch.

ExperiencePotential
Indicator of whether a plant has production on a platform that will be used for a launch in the forthcoming year. For instance, if platform P is launched at plant X in 2000, then all plants producing on platform P in 1999 have ExperiencePotential = 1. There may some oversimplification in this assignment, since if platforms P and Q are launched in 2000, plant Y will have ExperiencePotential = 1 if it produces on either platform.

ExperienceReal (Launch plants only)
Indicator of whether, given a launch on platform P at plant X in a given year, plant X had production on P in the preceding year.

FlexAssembly
Assembly Line Flexibility: ratio of number of platforms produced by a plant to number of assembly lines in the plant.

FlexBody
Body Shop Flexibility: ratio of number of platforms produced by a plant to number of body

shops in the plant.

priorHPV
Overall HPV (*i.e.*, across all models).

HPV
Prior year overall HPV (*i.e.*, across all models).

LateModel
Indicator variable: whether a plant manufactures a model that is within the last two years of its life cycle.

RampUp
The length of the ramp-up period following launch. Defined in four possible ways on the two years following launch: time to reach maximum production level, mean production level, median production level, or 90^{th} percentile of maximum production (see p127).

Sales
Total sales for all models produced at the plant in the year preceding the record year.

SalesTrend
Agregate slope coefficient from linear regressions of monthly sales data from the preceding two years for each model produced at the plant. Defined in three possible ways: average of coefficients, maximium coefficient, or minimum coefficient (see p128).

stdDev(Sales)
Standard deviation of total monthly sales for all models produced at the plant in the year preceding the record year.

priorUtilization
Utilization at a plant in the year prior to a particular year.

Utilization
Utilization at a plant in a particular year.

avgUtilization (Launch plants only)
Average utilization across the two year period following launch.

WorkingDays (Launch plants only)
Number of working days.

Y01effect (Launch plants only)
Percentage change in total HPV from the year preceding launch to the year of launch.

Y02effect (Launch plants only)
Percentage change in total HPV from the year preceding launch to the year following launch.

Table C.1: Summary Statistics

	Variable	Mean	Std. Dev	Min	Max
Launch Plants Only	Chassis	2.667	1.782	1	10
	CumulativeProduction 01	125862.000	119612.100	3284	638276
	CumulativeProduction 02	315389.800	182705.300	43514	1009491
	ExperienceReal	0.267	0.446	0	1
	FlexAssembly	1.242	0.500	1.000	3.000
	FlexBody	1.164	0.437	0.500	3.000
	HPV	31.362	8.959	17.300	56.410
	RampUp (90th Percentile)	19.200	9.669	3	43
	RampUp (Max.)	28.900	13.614	4	52
	RampUp (Mean)	13.117	8.967	2	43
	RampUp (Median)	14.883	9.151	2	33
	Sales	165094.500	196691.100	1471	876716
	stdDev(Sales)	5672.389	7845.135	543.089	61969.860
	avgUtilization	0.925	0.219	0.291	1.406
	Utilization	85.648	25.978	25.709	154.470
	WorkingDays	240.883	21.314	190	336
	Y01effect	11.839	15.991	-18.923	57.392
	Y02effect	-3.466	13.203	-28.310	33.816
All Plants	ExperiencePotential	0.127	0.334	0	1
	FlexAssembly	1.206	0.473	0.500	3.000
	FlexBody	1.118	0.387	0.500	3.000
	Chassis	2.761	2.302	1	16
	LateModel	0.130	0.337	0	1
	SalesTrend (Avg.)	-9.976	271.978	-1841.436	2104.311
	SalesTrend (Min.)	-76.999	277.409	-2246.021	2104.311
	SalesTrend (Max.)	70.888	303.109	-542.862	2104.311
	priorHPV	29.212	10.473	15.741	103.075
	priorUtilization	89.989	26.736	15.021	147.848

Table C.2: Variable Correlations - Launch Location Model

	Chassis	Experience Potential	Flex Assembly	Flex Body	prior HPV	Late Model	SalesTrend (avg.)	SalesTrend (max.)	SalesTrend (min.)
ExperiencePotential	0.0894								
FlexAssembly	0.0668	-0.0025							
FlexBody	0.2942***	0.0078	0.5258***						
priorHPV	0.3152***	0.0775	0.0901	0.2796***					
LateModel	-0.1035*	-0.0352	-0.0087	-0.0853	0.1098*				
SalesTrend (avg.)	0.0819	0.0929	-0.0158	0.0655	0.1051*	-0.0298			
SalesTrend (Max.)	0.1057*	0.244***	-0.0089	0.0654	0.0808	-0.0242	0.7749***		
SalesTrend (Min.)	0.058	0.0374	-0.0538	0.066	0.1414**	-0.0140	0.8644***	0.4858***	
priorUtilization	0.2005***	0.0338	-0.0695	-0.0874	-0.3618***	-0.1847	-0.0165	0.0252	-0.0852

Table C.3: Variable Correlations - Productivity Model

	Flex Assembly	Flex Body	Chassis	Cumulative Prod. 01	Cumulative Prod. 02	Experience Potential	Experience Realized
FlexBody	0.6178***						
Chassis	0.0634	0.0532					
Cprod01	0.0146	-0.0925	0.1169				
Cprod02	-0.0882	-0.1235	0.0999	0.8989***			
Experience Potential	-0.0081	-0.1241	0.0349	0.4872***	0.3450***		
Experience Realized	0.0101	-0.1557	-0.0995	0.5430***	0.4384***	0.5409***	
RampUp Max	0.1194	0.0503	0.0755	0.0894	0.1763	-0.0036	-0.2635**
RampUp Mean	0.0031	0.0261	-0.0251	0.0402	0.0729	-0.0374	-0.0206
RampUp Median	-0.0475	-0.0312	-0.0565	0.0935	0.1312	-0.0286	-0.0296
RampUp Percentile	0.0670	0.0108	-0.0876	0.11740	0.13460	-0.0517	-0.0558
Sales	-0.0887	0.0445	0.2546**	-0.0474	0.0735	-0.1896	-0.2406*
StDevSales	-0.0832	-0.0840	-0.1091	0.0114	0.1043	-0.1168	-0.0468
Total HPV Prior	-0.0036	0.0211	0.2466*	-0.0773	-0.2367*	0.0178	-0.1316
Utilization	-0.1144	-0.2112	0.1882	0.3531***	0.4079***	-0.0080	0.0561
avgUtilization	-0.1549	-0.1890	0.1756	0.3267**	0.4581***	-0.0676	0.0522
Utilization Prior	-0.0437	-0.1363	0.3587***	0.2729**	0.3507***	0.0877	0.0723
Working Days	-0.0259	-0.0028	0.2123	0.2601**	0.1915	0.0927	0.0443
Y01 effect	-0.0109	-0.1082	-0.0431	-0.2228*	-0.1094	-0.2366*	-0.3579***
Y02 effect	0.2078	0.0904	0.0424	-0.0341	-0.0333	-0.1250	-0.1721

	RampUp Max	RampUp Mean	RampUp Median	RampUp Percentile	Sales	stdDev (Sales)
FlexBody						
Chassis						
Cprod01						
Cprod02						
Experience Potential						
Experience Realized						
RampUp Max						
RampUp Mean	0.4627***					
RampUp Median	0.4783***	0.8836***				
RampUp Percentile	0.6741***	0.7312***	0.8033***			
Sales	0.1375	-0.0133	0.0088	-0.0363		
StDevSales	0.1811	-0.0121	-0.0183	0.0637	0.1576	
Total HPV Prior	-0.0323	0.2086	0.1255	0.0304	-0.2328*	-0.2913**
Utilization	0.1043	0.0113	0.0284	-0.0096	0.3301**	0.1040
avgUtilization	0.1215	0.1753	0.1946	0.0933	0.3471***	0.1873
Utilization Prior	0.1597	0.0677	0.0673	0.0486	0.4394***	0.0950
Working Days	0.1233	0.1522	0.1698	0.1556	-0.2031	-0.0308
Y01 effect	0.1769	0.1363	0.2130	0.1954	0.0864	-0.0180
Y02 effect	0.1031	-0.1810	-0.2036	-0.0695	0.1765	0.0470

	Total HPV Prior	Utilization	avg Utilization	prior Utilization	Working Days	Y01 effect
FlexBody						
Chassis						
Cprod01						
Cprod02						
Experience Potential						
Experience Realized						
RampUp Max						
RampUp Mean						
RampUp Median						
RampUp Percentile						
Sales						
StDevSales						
Total HPV Prior						
Utilization	-0.1815					
avgUtilization	-0.1545	0.8526***				
Utilization Prior	-0.1418	0.6704***	0.6876***			
Working Days	0.1305	0.2282*	0.3432***	0.2980**		
Y01 effect	-0.2156*	-0.2056	-0.1530	-0.1275	-0.0492	
Y02 effect	-0.4099***	0.1370	-0.0504	0.1272	-0.1797	0.4681***

Table C.4: OLS Regression for Launch Impact

Dependent variable: log(HPV).

Parameter	Base Model Coefficient	Launch Model Coefficient
Intercept	0.557***	0.469***
DCX	0.019	0.009
Ford	0.046***	0.051***
Nissan	-0.003	-0.045
Toyota	0.019	0.031
2000	-0.003	0.006
2001	0.046**	0.057***
2002	-0.011	0.009
2003	-0.000	0.016
2004	-0.009	-0.036
2005	-0.018	-0.023
log(Chassis)	0.013	0.011
log(priorHPV)	0.822***	0.844
priorUtilization	-0.000	-0.000***
FlexBody	0.003	-0.013
Launch		0.151***
R-sq.	0.817	R-sq. 0.862

Table C.5: Logistic Regression for Launch Location

Dependent variable: launch occurrence
at each plant (indicator).

Parameter	Odds Ratio	Coefficient
Intercept	NA	0.958
DCX	1.959*	0.682*
Ford	0.903	-0.080
Nissan	2.898	1.059
Toyota	0.596	-0.518
2000	1.271	0.218
2001	1.284	0.215
2002	0.449	-0.610
2003	1.187	0.085
2004	3.736**	1.395**
2005	2.549	0.909
log(priorHPV)	0.280**	-1.375**
LateModel	1.943	0.740
SalesTrend	1.000	0.000
priorUtilization	1.000	0.000
Experience	7.564***	2.010***
FlexBody	2.790**	0.976**
	Wald $chi^2(15)$	69.010
	P > chi^2	0.000

Table C.6: OLS Regression, Y01 effect

Dependent variable: percentage HPV change, launch year from prior year.

Parameter	Base Model Coefficient	Extended Model Coefficient
Intercept	51.539	71.385**
Auto Alliance	-10.091	-12.882
DCX	-5.740	-8.607
Ford	7.685	6.455
Nissan	-93.322	-101.640*
Toyota	7.104	3.506
2000	-5.359	-8.336
2001	1.428	0.591
2002	8.776	4.957
2003	3.554	2.658
2004	-3.934	-1.118
2005	-12.939	-6.021
log(Chassis)	0.702	0.875
CumulativeProd.	0.000*	0.000
RampUp	0.276	0.283
log(Sales)	-2.250	-2.178
stdDev(Sales)	1.487	1.544*
Utilization	-0.115	-0.166*
WorkingDays	-0.040	-0.053
Experience		-15.055**
FlexBody		-10.163**
	R-sq. 0.370	R-sq. 0.487

Table C.7: OLS Regression, Y01 effect with interaction term

Dependent variable: percentage HPV change,
launch year from prior year.

Parameter	Base Model Coefficient	Extended Model Coefficient
Intercept	51.539	48.105
Auto Alliance	-10.091	-13.917
DCX	-5.740	-10.238*
Ford	7.685	4.377
Nissan	-93.322	-121.562**
Toyota	7.104	1.923
2000	-5.359	-11.289
2001	1.428	-2.189
2002	8.776	1.696
2003	3.554	-0.024
2004	-3.934	-2.571
2005	-12.939	-5.974
log(Chassis)	0.702	0.933
CumulativeProd.	0.000	0.000
RampUp	0.276	0.896*
log(Sales)	-2.250	-1.699
stdDevSales	1.487	1.864**
Utilization	-0.115	-0.163*
Working Days	-0.040	-0.058
NoExperience*Ramp-up		-0.821
NoExperience		26.375**
FlexBody		-12.018**
R-sq.	0.400	R-sq. 0.515

Table C.8: OLS Regression, Y02 effect, all plants

Dependent variable: percentage HPV change from year before launch to second year after launch.

Parameter	Base Model Coefficient	Extended Model Coefficient
Intercept	-1.185	2.635
Auto Alliance	9.708	10.574
DCX	1.567	1.741
Ford	13.315***	13.631***
Nissan	-52.217	-55.899
Toyota	11.844	12.496
2000	-2.353	-5.412
2001	-0.731	-2.861
2002	0.322	-2.746
2003	-2.375	-5.696
2004	1.658	0.043
2005	-8.733	-5.372
log(Chassis)	3.113	3.019
log(Sales)	-0.197	-0.537
stdDev(Sales)	0.001	0.001
avgUtilization	-11.618	-10.685
Experience		-6.310
FlexBody		1.678
R-sq.	0.294	R-sq. 0.320

Table C.9: OLS Regression, Y02 effect, improving plants only

Dependent variable: percentage HPV change from year before launch to second year after launch.

Parameter	Base Model Coefficient	Extended Model Coefficient
Intercept	-22.125	-15.397
DCX	7.339**	6.023
Ford	9.352**	8.366*
Nissan	-43.153	-61.810
2000	-2.793	-3.416
2001	-6.143	-5.209
2002	3.404	4.185
2003	-0.642	0.572
2004	3.584	5.149
2005	-8.219*	-8.644*
log(Chassis)	3.405	3.257
log(Sales)	1.035	0.715
stdDev(Sales)	0.001	0.001
avgUtilization	-12.933*	-13.527*
Experience		-0.951
FlexBody		-2.723
R-sq.	0.551	R-sq. 0.563

Table C.10: Heckman Correction

		Coefficient
Productivity Model Dependent variable Y01effect	Intercept	49.166***
	DCX	-11.161**
	Ford	5.427
	2000	-2.653
	2001	9.208
	2002	13.321
	2003	8.351
	2004	-3.735
	2005	-1.841
	log(Chassis)	-0.433
	CumulativeProd.	-0.595
	Utilization	-0.127
	Experience	-15.037**
	FlexBody	-11.255**

		Coefficient
Launch Location Model Dependent variable Launch occurrence	Intercept	1.161
	Ford	-0.046
	DCX	0.441*
	2000	0.128
	2001	0.118
	2002	-0.331
	2003	0.051
	2004	0.708*
	2005	0.261
	log(priorHPV)	-0.980**
	LateModel	0.415
	SalesTrend	0.000
	priorUtilization	-0.001
	Experience	1.121***
	FlexBody	0.635***

/athrho	-0.571
/lnsigma	2.631***
rho	-0.516
sigma	13.892
lambda	-7.171

LR test of independent equations (rho = 0)
$Chi^2(1) = 0.870$, Prob > $chi^2 = 0.350$

BIBLIOGRAPHY

R. Adkins and D. Paxson. Optimality in asset renewals. In *Proceedings of the 10th Annual International Conference on Real Options*, 2006.

Adjo A. Amekudzi and Sue McNeil. *Infrastructure reporting and asset management: best practices and opportunities*. ASCE Publications, 2008.

Richard Bellman. *Dynamic programming*. Princeton University Press, Princeton, 1957.

J. W. M. Bertrand. The effect of workload dependent due-dates on job shop performance. *Management Science*, 29(7):799–816, 1983.

Dimitri P. Bertsekas. *Dynamic Programming and Optimal Control, Two Volume Set*. Athena Scientific, 2001.

Dmitri P. Bertsekas and John. N. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, 1996.

D. Bertsimas, I.C. Paschalidis, and J.N. Tsitsiklis. Asymptotic buffer overflow probabilities in multiclass multiplexers: an optimal control approach. *IEEE Transactions on Automatic Control*, 43(3):315–335, 1998.

Frederick M. Biedenweg and Robert E. Hutson. Before the roof caves in: A predictive model for physical plant renewal. *APPA Newsletter*, 30(7), 1982.

George W. Blazenko and Andrey D. Pavlov. The economics of maintenance for real estate investments. *Real Estate Economics*, 32(1), 2004.

James H. Bookbinder and Afzal Ibn Noor. Setting job-shop due-dates with service-level constraints. *Journal of the Operational Research Society*, 36(11), 1985.

N.E. Boudette. Chrysler gains edge by giving new flexibility to its factories. *The Wall Street Journal*, April 11, 2006.

Timothy F. Bresnahan and Valerie A. Ramey. Output fluctuations at the plant level. *The Quarterly Journal of Economics*, 109(3), 1994.

John H. Cable and Jocelyn S. Davis. Key performance indicators for federal facilities portfolios. Technical Report 147, National Academies Press, Washington, DC, 2005.

J.E. Carrillo and C. Gaimon. Improving manufacturing performance through process change and knowledge creation. *Management Science*, 46(2), 2000.

H.S. Chang, M.C. Fu, J. Hu, and S.I. Marcus. *Simulation-based Algorithms for Markov Decision Processes*. Springer, 2007.

C-Y. Chen, Z. Zhao, and M. O. Ball. A model for batch advanced available-to-promise. *Production and Operations Management*, Winter, 2002.

T.C.E. Cheng and M.C. Gupta. Survey of scheduling research involving due date determination decisions. *European Journal of Operational Research*, 38(2):156–166, 1989.

J. Chod and N. Rudi. Resource flexibility with responsive pricing. *Operations Research*, 53(3), 2005.

K.B. Clark and T. Fujimoto. *Product Development Performance: Strategy, Organization, and Management in the World Auto Industry*. Harvard Business School Press, 1991.

Richard W. Conway. Priority dispatching and job lateness in a job shop. *Journal of Industrial Engineering*, 16(4):228–237, 1965.

D.R. Cox and W. Smith. *Queues*. Methuen, London :, 1961.

C. Derman. Optimal replacement and maintenance under markovian deterioration with probability bounds on failure. *Management Science*, 9(3), 1963.

A.K. Dixit and R.S. Pindyck. *Investment Under Uncertainty*. Princeton University Press, 1994.

I.M. Dobbs. Replacement investment: Optimal economic life under uncertainty. *Journal of Business Finance & Accounting*, 31(5-6), 2004.

S. Eilon and I. Chowdhury. Due dates in job shop scheduling. *International Journal of Production Research*, 14(2):223–237, 1976.

S. Eilon and R. M. Hodgson. Job shops scheduling with due dates. *International Journal of Production Research*, 6(1):1 – 13, 1967.

D. Epple, L. Argote, and K. Murphy. An empiraical investigation of the micro structure of knowledge acquisition and transfer through learning by doing. GSIA working papers, Carnegie Mellon University, Tepper School of Business, 1995.

C. H. Fine and R. M. Freund. Optimal investment in product-flexible manufacturing capacity. *Management Science*, 36(4), 1990.

C. H. Fine and S. Pappu. Flexible manufacturing technology and product-market competition. Technical report, Massachusetts Institute of Technology, Sloan School of Management, 1990.

M.L. Fisher and C.D. Ittner. The impact of product variety on automobile assembly operations: Empirical evidence and simulation analysis. *Management Science*, 45(6), 1999.

B. Fleischmann, S. Ferber, and P. Henrich. Strategic planning of BMW's global production network. *Interfaces*, 36(3), 2006.

A. Gosavi. *Simulation-Based Optimization: Parametric Optimization Techniques and Reinforcement Learning*. Kluwer Academic, 2003.

M. Goyal and S. Netessine. Strategic technology choice and capacity investment under demand uncertainty. *Management Science*, 53(2), 2007.

M. Goyal, S. Netessine, and T. Randall. Deployment of manufacturing flexibility: An empirical analysis of the north american automotive industry. Technical report, 2006.

J. Michael Harrison. Dynamic Scheduling of a Multiclass Queue: Discount Optimality. *Operations Research*, 23(2):270–282, 1975.

J. Michael Harrison. *Brownian Motion and Stochastic Flow Systems*. John Wiley and Sons, 1985.

J.C. Hartman. An economic replacement model with probabilistic asset utilization. *IIE Transactions*, 33(9), 2000.

N.W. Hatch and D.C. Mowery. Process innovation and learning by doing in semiconductor manufacturing. *Management Science*, 44(11), 1998.

R. Hayes and S. Wheelwright. *Restoring our competitive edge: competing through manufacturing*. John Wily and Sons, 1984.

James J. Heckman. Sample selection bias as a specification error. *Econometrica*, 47(1): 153–161, 1979.

Michael G. Hegedus and Wallace J. Hopp. Due date setting with supply constraints in systems using mrp. *Computers & Industrial Engineering*, 39(3-4):293 – 305, 2001.

Wallace J. Hopp and Melanie Roof Sturgis. A simple, robust leadtime-quoting policy. *Manufacturing & Service Operations Management*, 3(4):321–336, 2001.

W.J. Hopp and S.K. Nair. Markovian deterioration and technological change. *IIE Transactions*, 26(6), 1994.

H. Hotelling. A general mathematical theory of depreciation. *Journal of the American Statistical Association*, 20(151), 1925.

Bongju Jeong, Seung-Bae Sim, Ho-Sang Jeong, and Si-Won Kim. An available-to-promise system for tft lcd manufacturing in supply chain. *Computers & Industrial Engineering*, 43(1-2):191 – 212, 2002.

Robert A. Jones and Joseph M. Ostroy. Flexibility and uncertainty. *Review of Economic Studies*, 51(1):13–32, 1984.

W.C. Jordan and S.C. Graves. Principles on the benefits of manufacturing process flexibility. *Management Science*, 41(4), 1995.

A. W. June. More than just maintenance. *The Chronicle of Higher Education*, October 10, 2003.

H. Kaiser. *Crumbling Academe: Solving the Capital Renewal and Replacement Dilemma*. Association of Governing Boards of Universities and Colleges, Washington, DC, 1984.

J.F. Krafcik. Comparative analysis of performance indicators at world auto assembly plants. Master's thesis, 1988.

Averill M. Law and David M. Kelton. *Simulation Modeling and Analysis*. McGraw-Hill Higher Education, 2000.

Michael Lechner. Program heterogeneity and propensity score matching: An application to the evaluation of active labor market policies. *Review of Economics and Statistics*, 84 (2):205–220, 2002.

B. Levitt and J. G. March. Organizational learning. *Annual Review of Sociology*, 14, 1988.

F. K. Levy. Adaption in the production process. *Management Science*, 11(6), 1965.

Marvin B. Lieberman. Firm-level productivity and management influence: a comparison on u.s. and japanese automobile producers. *Management Science*, 36(10), 1990.

Marvin B. Lieberman and Lieven Demeester. Inventory reduction and productivity growth: Linkages in the japanese automotive industry. *Management Science*, 45(4), 1999.

J. K. Luellen, W. R. Shadish, and M. H. Clark. Propensity scores: An introduction and experimental test. *Evaluation Review*, 29, 2005.

J.P. MacDuffie, K. Sethuraman, and M.L. Fisher. Product variety and manufacturing performance: evidence from the international automotive assembly plant study. *Management Science*, 42(3), 1996.

D. C. Mauer and S. H. Ott. Investment under uncertainty: The case of replacement investment decisions. *Journal of Financial and Quantitative Analysis*, 30(4), 1995.

E. Mayne. Chrysler sets ambitious new efficiency target. *WardsAuto.com*, (6/2), 2006.

J.J. McCall. Maintenance policies for stochastically failing equipment: A survey. *Management Science*, 11(5), 1965.

R. McDonald. The role of real options in capital budgeting: Theory and practice. *Journal of Applied Corporate Finance*, 18(2), 2006.

R. McDonald and D. Siegel. The value of waiting to invest. *Quarterly Journal of Economics*, 101(4), 1986.

R. McLaughlin and R.A. Taggart. The opportunity cost of using excess capacity. *Financial Management*, 21(2), 1992.

Jan A. van Mieghem. Dynamic scheduling with convex delay costs: The generalized $c\mu$ rule. *The Annals of Applied Probability*, 5(3):809–833, 1995.

S. Moses, H. Grant, L. Gruenwald, and S. Pulat. Real-time due-date promising by build-to-order environments. *International Journal of Production Research*, 42(20):4353–4375, 2004.

M. Muffatto and M. Roveda. Product architecture and platforms: a conceptual framework. *International Journal of Technology Management*, 24(1), 2002.

NASA. *NASA Real Property Facility Capital Plan*. National Aeronautics and Space Administration, April 2008. URL `http://www.hq.nasa.gov/office/codej/codejx/Assets/Docs/5-9-08FacilityCapitalPlanApril2008.pdf`.

R.P. Parker and A. Wirth. Manufacturing flexibility: Measures and relationships. *European Journal of Operational Research*, 118, 1999.

Richard Pibernik. Advanced available-to-promise: Classification, selected methods and requirements for operations and inventory management. *International Journal of Production Economics*, 93-94:239 – 252, 2005. Proceedings of the Twelfth International Symposium on Inventories.

G. Pisano. *The Development Factory*. Harvard Business School Press, 1995.

Warren Powell. *Approximate Dynamic Programming: Solving the Curses of Dimensionality*. Wiley-Interscience, 2007.

Herbert Robbins and Sutton Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, 22(3):400–407, 1951.

D. Rodney. Annual report of the auditor general of alberta. Technical report, Edmonton, Alberta, 2007.

Roland George, Purvin & Gertz. *Natural Gas Infrastructure Investments and Capital Renewal*. The Conference Board of Canada, Ottawa, ON, 2004.

L.H. Röller and M.M. Tombak. Strategic choice of flexible production technology and welfare implications. *Journal of Industrial Economics*, 38(4), 1990.

P. Rosenbaum and D.B. Rubin. The central role of the propensity score in observational studies for causal effects. *Biometrika*, 70, 1983.

Sean C. Rush. *Managing the Facilities Portfolio: A Practical Approach to Institutional Facility Renewal and Deferred Maintenance*. National Association of College and University Business Officers, Washington, DC, 1991.

J. Rust. Optimal replacement of gmc bus engines: An empirical model of harold zurcher. *Econometrica*, 55(5), 1987.

Abraham Seidmann and Milton L. Smith. Due Date Assignment for Production Systems. *Management Science*, 27(5):571–581, 1981.

Brett C. Smith, Sean McAlinden, and Bernard Swiecki. World class vehicle launch timing. Technical report, 1998. URL `http://www.autofieldguide.com/articles/029805.html`.

Mark L. Spearman and Rachel Q. Zhang. Optimal lead time policies. *Management Science*, 45(2):290–295, 1999.

J.D. Stoll. Krygier vows better productivity in 2004. *WardsAuto.com*, (6/11), 2004.

Alexander L. Stolyar and Kavita Ramanan. Largest weighted delay first scheduling: Large deviations and optimality. *The Annals of Applied Probability*, 11(1):1–48, 2001.

R. S. Sutton and A. G. Barto. *Reinforcement learning : an introduction*. A Bradford Book. The MIT Press, 1998.

S.G. Taylor and G.J. Plenert. Finite capacity promising. *Production and Inventory Management Journal*, 40(3):50–56, 1999.

Dong-Wan Tcha and Stanley R. Pliska. Optimal Control of Single-Server Queuing Networks and Multi-Class $M/G/1$ Queues with Feedback. *Operations Research*, 25(2): 248–258, 1977.

G. Terborgh. *Dynamic Equipment Policy*. McGraw-Hill, 1949.

C. Terwiesch and R. E. Bohn. Learning and process improvement during production ramp-up. *International Journal of Production Economics*, 70, 2001.

C. Terwiesch and Y. Xu. The copy-exactly ramp-up strategy: Trading-off learning with process change. *IEEE Transactions on Engineering Management*, 51(1), 2004.

C. Terwiesch, R. E. Bohn, and K.C. Chea. International product transfer and production ramp-up: A case study from the data storage industry. *R & D Management*, 31(4), 2001.

J. Van Biesebroeck. The cost of flexibility. *Assembly Automation*, 27(1), 2007.

J. A. Van Mieghem. Investment strategies for flexible resources. *Management Science*, 44 (8), 1998.

Jan A. Van Mieghem. Due-Date Scheduling: Asymptotic Optimality of Generalized Longest Queue and Generalized Largest Delay Rules. *Operations Research*, 51(1):113–122, 2003.

D. J. Vanier. Why industry needs asset management tools. *Journal of Computing in Civil Engineering*, 15(1), 2000.

T. F. Wallace and J.R. Dougherty. APICS dictionary : The official dictionary of production and inventory management terminology and phrases, 1987.

H. Wang. A survey of maintenance policies of deteriorating systems. *European Journal of Operational Research*, 139(3), 2002.

C. J. C. H. Watkins. *Learning from Delayed Rewards*. PhD thesis, University of Cambidge, England, 1989.

James K. Weeks. A simulation study of predictable due-dates. *Management Science*, 25 (4):363–373, 1979.

Lawrence M. Wein. Due-date setting and priority sequencing in a multiclass $m/g/1$ queue. *Management Science*, 37(7):834–850, 1991.

G. H. Weiss. On the theory of replacement of machinery with a random failure time. *Naval Research Logistics Quarterly*, 3(4), 1956.

E . L. Welker. Relationship between equipment reliability preventive maintenance policy and operating costs. Technical Report 7, ARINC Research Corporation, Washington, DC, 1959.

S. Westfall. *Recapitalization & Capital Renewal - What's the Number?* Tradeline, Inc., January 2001. URL `http://www.tradelineinc.com/reports/E81F7036-BECE-11D4-95B9005004022792`.

D. Winter. Critical differences. *WardsAuto.com*, (6/2), 2006.

J.P. Womack, D.T. Jones, and D. Roos. *The machine that changed the world*. Scribner, 1990.

T. P. Wright. Factors affecting the cost of airplanes. *Journal of Aeronautical Science*, 3, 1936.

F. Yilmaz. Conditional investment policy under uncertainty and irreversibility. *European Journal of Operational Research*, 132(3), 2001.