DISSERTATION

APPROXIMATE DYNAMIC PROGRAMMING APPLICATION TO INVENTORY MANAGEMENT

Submitted by

Tatpong Katanyukul

Department of Mechanical Engineering

In partial fulfillment of the requirements

For the Degree of Doctor of Philosophy

Colorado State University

Fort Collins, Colorado

Spring, 2010

COLORADO STATE UNIVERSITY

April 6, 2010

WE HEREBY RECOMMEND THAT THE DISSERTATION PREPARED UNDER OUR SUPERVISION BY TATPONG KATANYUKUL ENTITLED APPROXIMATE DYNAMIC PROGRAMMING APPLICATION TO INVENTORY MANAGEMENT BE ACCEPTED AS FULFILLING IN PART REQUIREMENTS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY.

Committee on Graduate work

Allan T. Kirkpatrick

Christian Puttlitz

Edwin K. P. Chong

Advisor: William S. Duff

Department Head: Allan T. Kirkpatrick

ABSTRACT OF DISSERTATION

APPROXIMATE DYNAMIC PROGRAMMING APPLICATION TO INVENTORY MANAGEMENT

This study has developed a new method and investigated the performance of current Approximate Dynamic Programming (ADP) approaches in the context of common inventory circumstances that have not been adequately studied in the literature. The new method uses a technique similar to eligibility trace[113] to improve performance of the residual gradient method[7]. The ADP approach uses approximation techniques, including learning and simulation schemes, to provide the flexible and adaptive control needed for practical inventory management. However, though ADP has received extensive attention in inventory management research lately, there are still many issues left uninvestigated. Some of the issues include (1) an application of ADP with a scaleable, linear operating capable, and universal approximation function, i.e., Radial Basis Function (RBF); (2) performance of bootstrapping and convergence-guaranteed learning schemes, i.e., Eligibility Trace and Residual Gradient, respectively; (3) an effect of latent state variables, introduced by recently found GARCH(1,1), to a model-free property of learning-based ADPs; and (4) a performance comparison between two main ADP families, learning-based and simulation-based ADPs. The purpose of this study is to determine appropriate ADP components and corresponding settings for practical inventory problems by examining these issues.

A series of simulation-based experiments are employed to study each of the ADP issues. Due to its simplicity in implementation and popularity as a benchmark in ADP research, the Look-Ahead method is used as a benchmark in this study. Conclusions are drawn mainly based on the significance test with aggregate costs as performance measurement. The performance of each ADP method was tested to be comparable to Look-Ahead for inventory problems with low variance demand and shown to have significantly better performance than performance of Look-Ahead, at 0.05 significance level, for an inventory problem with high variance demand. The analysis of experimental results shows that (1) RBF, with evenly distributed centers and half midpoint effect scales, is an effective approximate cost-to-go method; (2) Sarsa, a widely used algorithm based on one-step temporal difference learning

(TD0), is the most efficient learning scheme compared to its eligibility trace enhancement, Sarsa(λ), or to the Residual Gradient method; (3) the new method, Direct Credit Back, works significantly better than the benchmark Look-Ahead, but it does not show significant improvement over Residual Gradient in either zero or one-period leadtime problem; (4) a model-free property of learning-based ADPs is affirmed under the presence of GARCH(1,1) latent state variables; and (5) performance of a simulation-based ADP, i.e., Rollout and Hindsight Optimization, is superior to performance of a learning-based ADP. In addition, links between ADP setting, i.e., Sarsa(λ)'s Eligibility Trace factor and Rollout's number of simulations and horizon, and conservative behavior, i.e., maintaining higher inventory level, have been found.

Our conclusions show agreement with theoretical and early speculations on ADP applicability, RBF and TD0 effectiveness, learning-based ADP's model-free property, and that there is an advantage of simulation-based ADP. On the other hand, our findings contradict any significance of GARCH(1,1) awareness, identified by Zhang [130], at least when a learning-based ADP is used. The work presented here has profound implications for future studies of adaptive control for practical inventory management and may one day help solve the problem associated with stochastic supply chain management.

Tatpong Katanyukul
Department of Mechanical Engineering
Colorado State University
Fort Collins, Colorado 80523
Spring, 2010

ACKNOWLEDGEMENTS

First of all, I am pleased to thank my father Weerawuth, my mother Petch and my brother Nitipat Katanyukul for major financial, morale and spiritual support before and throughout this academic pursuit. I am grateful to my advisor, Dr. William Duff, for his guidance, patience and Mettā (Buddhism loving kindness); to Dr. Edwin Chong for his counseling, encouragement and positive attitude toward this learning process, research, academic career and life; to Dr. Charles Anderson for his suggestion, comments and passion on machine learning that carries on to inspire parts of this research.

I would also appreciate Dr. Allan Kirkpatrick and Dr. Christian Puttlitz for serving as my committee members; Karen Mueller for copying editing this dissertation; NPSpecies project manager and my boss Alison Loar for providing me a student friendly and international student permitted job that helps support my living as well as broaden my perspective on biodiversity, conservation, nature, history, recreation, public work and national park roles in nurturing society; Adam Berrada and his father for proof-reading the first draft of my proposal; Ivan Rivas for encouraging and lending me Bolker's *Writing Your Dissertation in Fifteen Minutes a Day* that, though I spent more than 15 minutes a day, helps persevere me on writing this dissertation; Direk Khajonrat for assisting me on Matlab and Latex as well as sharing his academic pursuit experience; Manupat and Ornrat Lohitnavy for helping me settle down in Fort Collins when I first came; Sirirat Niyom for listening, understanding and comforting my frustration and anxiety later of this pursuit; and who I did not mention their names here including other professors, extended family members, friends and friendly people around for inspiring, motivating, encouraging, supporting, comforting and helping me in my study or other aspects of life that complement this learning process of mine.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

"The most important dimension of ADP is 'learning how to learn', and as a result the process of getting approximate dynamic programming to work can be a rewarding educational experience." - Warren B. Powell [97].

1.1 Research Framework

Inventory management is a major function of many businesses, especially in wholesaling, retailing and manufacturing. Proper management of inventory can help corporations reduce costs and stay competitive. Hence, there have been many inventory management studies since Harris (1913), who is credited with making the first real inventory study[40]. The motivation of our study originated from recurring problems of inefficient inventory control of an agrochemical-product distributor in Thailand. Inefficient inventory control causes the business to be short of cash from time-to-time and may result in unnecessary expenditures.

A multi-period inventory management problem can be modeled as a Markov Decision Process. It can be solved by problem specific analyses or dynamic programming methods. The structure of these approaches are often too problem specific[73, 118]. In addition, they frequently require hard-to-obtain information, like a transition probability in case of exact dynamic programming[1].

Analytical approaches to inventory problems have been studied extensively. These approaches can provide optimal solutions when its assumptions are justified. However, due to a variety of inventory structures, inventory problems appear in various forms and their forms often change over time. Since an analytical approach is usually highly problem specific and requires high levels of analytical skills and much effort[64, 73]. Therefore, these are usually not suitable approaches for practical inventory management, especially for small businesses which have limited resources.

Exact dynamic programming is based on an analytical analysis and it is designed to obtain an optimal answer, called a control policy. It is based on searching all state-action space as well as the calculation of expected values. An expectation calculation requires knowing a transition probability and calculation of all possible states, making exact dynamic programming inefficient for problems with a large state-action space. This is referred to as the curse of dimensionality[13][2].

In order to obtain good control within a Markov Decision Process, the future consequence of current control has to be taken into the account. Cost-to-go is a simple term often referred to as this future consequence. The exact cost-to-go solution is extremely difficult, if not impossible, to obtain in practice for any stochastic problem. Exact dynamic programming uses expectation calculations for this cost-to-go solution. Expectation calculations often have high computational requirements

[1] Werbos [126] uses term exact dynamic programming to distinguish it from approximate dynamic programming.
[2] An example to illustrate the curse of dimensionality is provided in §2.4.

and require hard-to-obtain transition probabilities. Obtaining an exact solution usually requires a rigorous analysis[73] and other hard-to-obtain information. Often, rigid assumptions are made in order to develop a solution. Van Roy et al. [118] also raises the issue of inflexibility. That is, exact solutions tend to be too problem specific and cannot be adapted well to a change in the environment. And, they are likely to perform poorly when the underlying assumptions are violated. Many articles, such as Silver [109], Lee and Billington [77], and Bertsimas and Thiele [16] to name a few, address the need for an efficient flexible inventory solution that is simple to implement in practice.

Recently the use of Approximate Dynamic Programming (ADP) has received growing attention for many decision and automatic control applications. ADP solution approaches tend to be more flexible and adaptable than analytical or exact dynamic programming approaches. This property makes ADP suitable for practical decision applications, including inventory management. ADP approaches use various approximation techniques, depending on each ADP method, to overcome difficulties, such as the high computation/memory and transition probability requirements of exact dynamic programming.

The adaptability of ADP is attributed in part to a learning-based ADP structure. With observed information, a learning-based ADP method uses a learning scheme to correct a relation between the state-action and its consequence. In addition to learning-based ADP, there is simulation-based ADP. A simulation-based ADP method is a good alternative for inventory problems, since a simulation model of a particular application is relatively easy to develop. With the model, a simulation-based ADP method uses simulation to assist in inventory decisions. The types of ADPs to use, how they can be used, and other major ADP associate issues are investigated in the current study.

A mechanism of learning-based ADPs is generally achieved with two main components: a learning strategy and a function approximation. A function approximation is a method to memorize relations that have been learned. There are broad ranges of possible implementation choices for a learning strategy and an approximation function. An inappropriate choice could lead to divergence and poor performance, as discussed by Bertsekas and Tsitsiklis [15] and Falas and Stafylopatis [41]. Sutton and Barto [114] suggest a function belonging to a linear family for a cost-to-go approximation. As discussed by Barreto and Anderson [9], the Radial Basis Function (RBF) is in the linear family and is one of the most widely-used approximation functions. However, the RBF has not been studied with ADP for inventory problems. While the RBF approach is well-developed for supervised learning applications, such as regression and classification, applying ADP with RBF to inventory problems is much less so. A supervised learning application has all data available, allowing for a data design approach, such as Chen et al. [26]'s Orthogonal Least Square (OLS) with a single scale. In an ADP

context, data is obtained incrementally. The RBF has to be designed either from initial data or from another RBF design scheme. For inventory problems, reasonable ranges of a system state and action can be estimated. This domain information can then be used in an RBF design. We propose an intuitive RBF design, show its advantage over an OLS design, and develop a systematic approach to determine associated parameters.

The other component of a learning-based ADP method is a learning strategy. A learning strategy is a method to correct learned relations using observed information. Due to its effectiveness and its link to mammal learning processes, one-step temporal-difference learning, TD(0), is one of the most widely studied learning strategies. Eligibility Trace, TD(λ), is a bootstrapping technique used to speed up a learning process in TD(0). It has been shown to be an effective method in many studies, including Tesauro [116] and Gelly and Silver [43]. However, TD(λ) has never been studied for inventory problems before. Experiments in the current research use Sarsa as an implementation of TD(0) and Sarsa(λ) as an implementation of TD(λ). The results show no performance improvement of Sarsa(λ) over Sarsa. However, unexpectedly, a link between a degree of bootstrapping and conservative behavior was found.

Residual Gradient is a learning strategy designed to be used with function approximation. Its convergence is guaranteed, but performance is slow comparing to Sarsa. [See 7, for details]. In order to improve the Residual Gradient approach, we took the idea of Eligibility Trace and developed the Direct Credit Back (DCB) method. Our experiments indicate that DCB's average costs were lower than those when using the Residual Gradient method, but the significance tests could not confirm the difference at 0.05 significance level.

Recently, Zhang [130] has found evidence of temporal demand heteroscedasticity, GARCH(1,1), in inventory data and showed a significant cost penalty was incurred when the GARCH(1,1) model was not accounted for. The GARCH(1,1) model introduces two extra state variables, which are unobservable without a correct model of the problem. These latent state variables posted relate to a model-free property of a learning-based ADP method. Without a complete model, these latent state variables will be unintentionally left out. It should be noted that this is unlike a case of a Partially Observable Markov Decision Process (POMDP)[See 66, 79, for a short introduction to POMDP], because we are unaware of these latent state variables and they are not taken into account. Our experiments showed robustness of a learning-based ADP method against the missing information and provided evidence that the model-free property of a learning-based ADP method is viable.

When a model of a problem is available, a simulation-based ADP method is an alternative. With a model, a simulation-based ADP method uses simulation to generate possible consequences

from candidate actions. The action is then chosen based on information obtained from simulated consequences. Simulation-based ADP, Rollout and Hindsight Optimization, are investigated here. They are shown to perform better than learning-based ADP with Sarsa. Similar to Sarsa(λ), a link between Rollout parameters and their conservative behavior consequence is also found.

The findings here provide guidance for a practical approach to designing an ADP method for an inventory problem and an insight into relations of ADP components, performance, and control behavior. In addition, the results reaffirm the model-free property of a learning-based ADP method even in the presence of latent state variables introduced by the GARCH(1,1) model. These findings are expected to improve the efficiency of inventory management and convey the merit of ADP research into practice.

1.2 Research Statement

Although ADP has been used for inventory management, many of its aspects have not been investigated. Our study addresses many of the unanswered or inadequately answered questions: How should RBF be set up for ADP, can Eligibility Trace improve TD(0) performance for inventory problems, how does TD(0) perform without the latent state variables introduced by the GARCH(1,1) model and how do simulation-based ADP methods do compared to learning-based ADP methods. RBF has strong potential for ADP applications beyond single-echelon inventory problems. A systematic approach for setting up RBF will yield benefits for problems studied here as well as larger and more complicated problems.

Eligibility Trace is seen as a technique that can speed up the learning process and improve ADP performance. However, the use of Eligibility Trace in inventory management has not yet been studied. The investigation of its application here will promote understanding of how Eligibility Trace affects decisions and provides an assessment of whether it is worth an extra effort to implement it.

The study here of TD(0) performance in the absence of latent state variables will provide evidence supporting or contradicting a model-free attribute of a learning-based ADP method under GARCH(1,1) latent state variables. That is, the results here may support or contradict Zhang [130]'s concern about the presence of the GARCH(1,1) model in inventory data.

Lastly, an examination of simulation-based ADP methods may focus renewed attention on a less studied family of ADP methods.

Findings in this research are expected to provide an improved capability for finding practical solutions for inventory control as well as establish new insights into ADP behavior in general.

1.3 Literature Review

ADP has been recently introduced into inventory management research by Van Roy et al. [118], Godfrey and Powell [48], Pontrandolfo et al. [93], Giannoccaro and Pontrandolfo [47], Shervais et al. [108], Kim et al. [70], Choi et al. [30], Topaloglu and Kunnumkal [117], Iida and Zipkin [62], Chaharsooghi et al. [25], Kim et al. [71], Kwon et al. [75], and Jiang and Sheng [64].

Of all these authors, only Choi et al. [30] investigates the application of simulation-based ADP. Simulation was used to provide reduced state space, reduced action space, and approximate transition probabilities for a dynamic program, which in turn was solved with either value iteration or Rollout. Rollout uses simulation to provide approximate state-action costs. The simulation requires a control method to provide decisions in simulation. Such a control method is called a base policy. For their base policy, Choi et al. used an (s,S) policy whose parameters were obtained from a heuristic search over pre-defined sets. The pre-defined sets of parameters used in Choi et al. are problem specific and it is unclear how Choi et al. obtained them. Rollout is also investigated in our study. We use a simple formula based on the well-known Economic Order Quantity (EOQ) equation to determine parameters for the base policy. In addition to Rollout, Hindsight Optimization (HO), introduced by Chong et al. [31], has never been investigated for inventory problems. It is another simulation based ADP approach that does not require a base policy. Therefore we investigate HO for its own virtues as well as to provide a useful measure of how simulation-based ADP performs without the choice of a base policy.

Authors studying learning-based ADP methods investigated several learning schemes. Van Roy et al. [118] used one-step temporal difference learning (TD0). Chaharsooghi et al. [25] used Q-learning, an off-policy variation of TD(0). Kim et al. [70] used an action-value method whose learning scheme was based on a weighted average value of a current approximation and a new observation. Their approach is similar to TD(0), but it only approximates a current state-action value without a value-to-go. Kwon et al. [75] and Jiang and Sheng [64] used the case-based myopic reinforcement learning (CMRL) method developed by Kwon et al. CMRL is based on a combination of an action-value method and a case-based reasoning technique. Case-based reasoning is state aggregation with an ability to create a new aggregation when an observed state value may vary over a preset range of any existing aggregation group. Kim et al. [71] proposed and used an asynchronous action-reward learning method. For a fast changing inventory system they assumed that information of action-consequence relations, regardless of state, was sufficient for decision making. Their asynchronous action-reward learning scheme is developed based on characteristics of inventory problems that allows simultaneously multiple action updates. Multiple action updates help accelerate the learning

process to enable it to catch up with changes in the system. Instead of only updating an action-reward value for an action taken, approximate values of actions not taken were updated as well. Given an observation of an exogenous variable, such as demand, consequences of actions not taken can be calculated and the multiple updates achieved with these computed consequences. Shervais et al. [108] used the dual heuristic programming method (DHP), introduced by Werbos [124]. DHP is a learning ADP scheme that updates a control policy directly using derivatives of the cost function. It should be noted that inclusion of a set up cost, formulated as a mathematical step function, renders this method inapplicable to the problems addressed in our study, because a step function is not differentiable[3].

Giannoccaro and Pontrandolfo [47] used the SMART algorithm, developed by Das et al. [36]. The SMART algorithm is similar to Q-learning, developed by Watkins [123]. In Q-learning, every time step is assumed to be equal. Giannoccaro and Pontrandolfo studied an inventory problem whose time response is a function of a current state, a next state and a current action. To handle varied time response, SMART uses a time correction term and its associate procedures to approximate an average state-action value. Our work investigates TD(0) implementation Sarsa and Eligibility Trace implementation Sarsa(λ). In addition, to improve Residual Gradient performance a Residual Gradient method, developed and guaranteed to converge by Baird [7], and a Direct Credit Back method, developed in our current research, are included in our study. A learning scheme used in Van Roy et al. [118, §6.2] is equivalent to Sarsa. The Sarsa(λ) and Residual Gradient approaches have never been studied for inventory problems before. The development of the Direct Credit Back method is original with our analysis.

For the issue of function approximation, Jiang and Sheng [64], Kim et al. [71], Kwon et al. [75], Kim et al. [70] used a Look-Up table to implement a cost-to-go approximation. A Look-Up table is a simple index table whose entry, such as an approximate cost, can be acccessed by an index, such as a state-action pair. Giannoccaro and Pontrandolfo [47] and Chaharsooghi et al. [25] used an Aggregation. An Aggregation is an ehhanced version of a Look-Up table. It is a Look-Up table with a group of indices, instead of a single index. Any indexing value falling within the same index group will be linked to the same entry. For the same problem, an Aggregation will need a smaller size table than a Look-Up table. Van Roy et al. [118] experimented with a linear combination of features and the Multilayer Percentron Neural Network (MLP). Shervais et al. [108] also used MLP for an approximate cost-to-go. Among these approximation choices, a Look-Up table is simplest to

[3] There is a method to approximate a step function with a sigmoid function. A sigmoid function is differentiable. However, our pre-experiments showed that even though an approximate step function was differentiable, simple approximation of a step funciton with a sigmoid function has lead to highly inefficient computation.

implement, but it suffers from a scalability issue. An Aggregation is a good alternative, but a size of its aggregation step needs to be carefully designed. A linear combination of features provides the efficiency of a linear computation, but it requires a customized selection of features specific to each problem. MLP is a very powerful approximation function, but its highly nonlinear nature makes it difficult to fine tune with ADP. A Radial Basis Function (RBF) is a universal approximation function. It can be operated in a linear mode, which results in a more stable ADP approach. RBF is a linear combination of locally active functions. Therefore it can be viewed either as a smooth interpolation of an Aggregation or as a linear combination of features, which are the Radial Bases. Our study investigates an application of ADP with RBF to inventory problems to provide information of this unexplored alternative for a cost-to-go approximation.

Among previous authors applying ADP to inventory problems, no one has investigated the performance differences between simulation-based and learning-based ADP, the performance of bootstrapping for TD(0), applicability to inventory problems of ADP with RBF, nor the effect of GARCH(1,1) latent state variables in learning-based ADP. The intent of our study is to provide insights into these unexplored issues in order to foster ADP application to practical inventory management.

1.4 Research Evaluation

From the point-of-view of the inventory research community, Simchi-Levi et al. [111] pointed to an evaluation of inventory solutions as a fundamental research question and identified empirical comparisons, worst-case analysis and average-case analysis as three commonly used methods. However, Simchi-Levi et al. [111] commented that analysis of a worst-case or average-case performance may be technically very difficult, especially for complicated systems. Expressed as a similar view from the ADP research community, Powell [97] also referred to such an evaluation as one of the major issues in ADP research. A common stratagy is to compare ADP to benchmarks, such as an optimal solution in a simplified problem, an optimal deterministic solution and a simple-to-implement Look-Ahead method, sometimes referred to as a rolling horizon policy.

Previous authors applying ADP to inventory problems also use empirical comparisons to evaluate their performance. Those authors are Van Roy et al. [118], Godfrey and Powell [48], Shervais et al. [108], Kim et al. [70], Topaloglu and Kunnumkal [117], Choi et al. [30], Iida and Zipkin [62] and Lu et al. [80]. Their evaluations vary depending on objectives and criteria of problems and on research questions posed in each individual work. Benchmarks used are different among different studies. So are the performance measurements. Total cost, total profit, and their other variations are among the most commonly used performanace measurements.

Van Roy et al. [118] investigated the potential of two ADP methods: an approximate policy iteration method and a TD(0) method. They studied them using two different problems: (1) a system having one warehouse and one retailer and (2) a system having one warehouse and ten retailers with a significant transportation delay. The ADP methods were used to determine a base-stock parameter for a base-stock policy. (See Nahmias and Smith [86] for a base stock policy.) Van Roy et al. [118] used an average cost as a performance indicator. These results were compared with a base-stock policy whose parameters were determined by exhaustive search. Van Roy et al. [118] used a lengthy simulation to allow enough time for ADP performance to converge. It should also be noted that latter studies have put more effort into stabilizing ADP control. Shervais et al. [108] used a more stable control to start up the system. Kim et al. [70] used a combination of a deterministic method and ADP. The ADP method was used to control only the uncertainty parts of the system via a mechanism of safety factors. Choi et al. [30] and Iida and Zipkin [62] used simulation-based ADP methods to provide more stable control.

Kim et al. [70] investigated the combination of ADP and a deterministic approach. They use ADP to control only the uncertainty part of the problem and use the deterministic approach to handle the more predictable parts of the problem. This was done to stabilize the system while allowing the solution to still be adaptive enough to handle uncertainty and changes. A Temporal Difference learning method and a softmax method were used to determine parameters that handled uncertainty, a safety leadtime and safety stocks. Then a safety leadtime and safety stocks were put into a deterministic forecasting formula to determine a replenishment order. Kim et al. [70] investigated both centralized and distributed control structures for two-echelon inventory problems. Their objective was to control service levels to a predefined target. The target service level is the percentage of customer demand that has to be satisfied during the time interval between order placement and inventory replenishment. Their simulation results were presented with service levels versus iterations and service levels versus different non-stationary conditions. Looking at service levels versus iterations shows how much a service level deviates from the target as time progresses. Looking at service levels versus different non-stationary conditions allows for a comparison of their different approaches, such as centralized and distributed controls. Since they intended to investigate the multi-echelon strategies between centralized and distributed controls, they compared decentralized and centralized results with one another. The results showed that the centralized control is more stable than the distributed control, as the centralized control can deliver more consistent service levels throughout different scenarios.

Godfrey and Powell [48] investigated a single-period inventory problem, often called a newsvendor problem. Unlike a multi-stage problem, a decision in each time period of a newsvendor problem

will have no consequence in latter periods. They proposed a concave piecewise linear approximation method, referred to as CAVE, and used it to approximate a relation between profit and a replenishment order. Since the problem is single period, this relation can be used to determine a replenishment order directly. Godfrey and Powell [48] used a total profit as a performance measurement. Their objective was mainly to demonstrate how their proposed CAVE method could be used to approximate the concave relation without any assumption or prior knowledge of a distribution of demand. They used an inventory control based on a Guassian model as a benchmark to show how robust CAVE was compared to a model-based method. As expected, their simulation results showed that a Gaussian based method performed better when demands were generated from Gaussian and Poisson distributions with large means. The CAVE approach worked better than a Gaussian based method when demand was generated from a uniform distribution.

Shervais et al. [108] studied an application of Dual Heuristic Programming (DHP) by Werbos [124] on a mixed inventory and transportation problem in a two-echelon structure under both stationary and non-stationary customer demands. They used a more stable control, a linear programming (LP) method or a genetic algorithm (GA), to initialize the system and later switched to DHP, which is more adaptive, to improve the initial performance. The objective of their study was to investigate that particular combination control strategy. That is, the use of a stable control to stabilize operations during initial runs and then using an adaptive control to improve later performance. They then compared results obtained from the combination control to each stable control alone. The stable control used was a fixed control policy obtained initially from either LP or GA. They conducted simulations with stationary, smooth increase and step increase demands to evaluate their approach. They claimed the validity of pink noise, also known as a 1/f distribution, to model demand used in their study. A total cost was used as a performance measurement. Their results showed that DHP improves performance of each stable control significantly. The combination of GA initialized control and DHP delivered the best performance among all test scenarios.

Topaloglu and Kunnumkal [117] studied approaches to solve multi-echelon problems with multiple suppliers. They proposed two approaches: an approach using linear programming to solve a linear approximation of the problem and an approach using Lagrangian relaxation, discussed by Hawkins [54], to relax the constraints that link decisions to suppliers. Topaloglu and Kunnumkal [117] evaluated both approaches with simulation of different scenarios. The total expected profit was used as a performance measurement and the eight-period Look-Ahead method was used as a benchmark. Their results showed that the Lagrangian relaxation-based method outperformed the linear programming-based method and both of their methods outperformed the eight-period Look-Ahead method.

Choi et al. [30] proposed a method, called DP in a heuristically restricted state space, to obtain a dynamic program with reduced state space of multi-echelon inventory problems. To improve efficiency of dynamic programming and to provide required information, they used simulation of various potential scenarios for generating approximating information, such as reduced state space, reduced action space, and approximate transition probabilities. Total profit was their performance measure. Their approach is evaluated with simulation-based experiments and a heuristic search is used as a benchmark. A similar heuristic search was also used as a base policy in a simulation that generates approximate information. Choi et al. [30] claimed their proposed method achieved about 4.5 % performance improvement over the heuristic control alone.

Iida and Zipkin [62] used the Martingale Model of Forecast Evolution (MMFE)[53, 57] to explicitly incorporate the demand forecast into an inventory model. Without a set up cost in their problems, Iida and Zipkin arranged the one-period cost formulation such that the one-period cost was not a function of an initial inventory. Then, with an approximation of a cost function as a piecewise linear function, the problem was solved backward to obtain the optimal base-stock level. It should be noted that the presence of a set up cost in our investigation does not allow for a similar rearrangement of the one-period cost formulation. Iida and Zipkin [62] analyzed performance bounds and used simulation-based experiments to evaluate their proposed method. An estimate expected total cost was used as a performance measurement. Since the purpose of their study was to investigate the effect of a forecast horizon, performance of their methods with different forecast horizons were compared. Their results showed that there was no significant difference in performance among one- to four-period forecast horizons and led to a conclusion that a one-period forecast has the most significant effect.

Similar to Iida and Zipkin [62], Lu et al. [80] investigated an inventory problem with MMFE. Lu et al. [80] used an analysis of a sample path, a concept based on a sequence of events, to develop upper and lower bounds of the optimal base-stock level. Then, they determined the base-stock level from a weighted combination of the two bounds whose weights minimized an upper bound of a relative cost error. Lu et al. [80] used the Iida and Zipkin [62] method as a benchmark. Their simulation results showed that their solution yielded lower values of an upper bound of relative cost errors in most of the cases they examined. However, it should be noted that while Iida and Zipkin [62]'s method is ready to use without significant extra analytical work, the method of Lu et al. [80] requires extra work in the form of determining an expectation of the sample path, to implement it in practice.

As a commonly accepted approach to evaluate an ADP solution for an inventory problem, our study also employs simulation-based experiments. A Look-Ahead method is used as a benchmark.

An aggregate cost is used as the main performance measurement. Other observations are included when needed or to enhance the analysis.

CHAPTER 2

BACKGROUND

This chapter explains a background for this research. The content is organized into three sections: (1) inventory types and classical inventory management, (2) previous inventory studies and our original research motivation, (3) a Markov Decision Process and classical Dynamic Programming methods and (4) Approximate Dynamic Programming and its related issues.

2.1 Inventory

Inventory management is activities of planning and maintaining an appropriate inventory level in a storage, e.g., a warehouse, in order to keep operating costs low without jeopardizing customer service or disrupting other activities, e.g., production (production inventory), maintenance and preventive maintenance (spare part inventory). Due to the amount of capital tied up in inventory, the cost of expediting replenishment or a potentially negative consequence of inventory shortages, an inventory decision is a major concern in management. Silver [110] provided practical examples illustrating benefits of inventory modeling. They are (1) a case of $20-million-a-year savings for IBM by using a new spare part multi-echelon inventory system; (2) a case of $2-million-a-year savings for US Navy by using an approach based on inventory modeling; and (3) a case of $23.9-million-savings and 95% drop in backorders over 3-year period for Pfizer Pharmaceuticals by using inventory modeling.

Inventory plays many important roles in a firm. Lambert et al. [76] identifies these roles as a way to benefit from economy of scale, to balance supply and demand, to gather products from different manufacturers in one place and to buffer uncertainty in supply and demand[1].

Inventory can be categorized from many points of view, for example, its function, how it is modeled, items it held and how it is managed. Lambert et al. [76] classified inventory by a function or a purpose of the inventory into cycle stocks, in-transit stocks, safety stocks, speculative stocks and seasonal stocks. Cycle inventories are items stocked to supply the predicted demand. Generally, they refer to a repeated replenishment cycle. In-transit inventories are items in transit from one location to another. They are considered not available to serve demand. Once they arrive at their destination, they will become another kind of inventory. Safety or buffer inventories are items held in excess of a cycle stock to handle uncertainties in demand or supply. Speculative inventories are items held for special benefit such as taking advantage of economics of scale. Seasonal inventories are items held for either seasonal supply or seasonal demand. Dead inventories are items having no demand for a specified period of time. Usually, these inventories refer to obsolete items.

[1] Currently an inventory role as a buffer easing down uncertainty is disputable. Many works, including a well-known work of Lee et al. [78], showed how inventories, without proper coordination, amplified uncertainty in a supply chain.

Waters [122] classified inventory by the type of items into raw materials, work in process, finished goods, spare parts and consumables. Raw materials are items to be processed before they can be used. Work-in-process are items being processed but not completely finished. Finished goods are items ready to be used. Spare parts are items to replace other similar type items that are defective or scheduled for replacement. Consumables are items such as oil and fuel.

Brown [20] classified inventory by the way it is managed into pull and push systems. In pull systems, no inventory status information is shared with suppliers. Inventory is managed by an inventory owner. Suppliers are unaware of a status of the inventory. The inventory is viewed as it is pulled from suppliers by a replenishment order from an inventory owner. In push systems, some inventory information is shared with suppliers. A shared inventory status lets suppliers better plan to provide enough supply for an inventory. The inventory owner still manages the inventory. In addition, Vendor-Managed Inventory (VMI), rather than just share information, lets suppliers manage its inventory directly, usually under agreed constraints, e.g. maintaining a customer service level within a specific range. From a modeling point of view, VMI can be modeled as multi-echelon inventory as if inventories and suppliers are only facilities of different hierarchies in the same organization.

Quantitative studies classified inventory by their modeling characteristics. (1) On-hand inventories are items held in stock and ready to deliver to customers immediately. A cycle stock, a safety stock, a speculative stock, a seasonal stock or a dead stock is an on-hand inventory. (2) On-order inventories are items in transit. Many Operations Research practitioners combine on-hand and on-order inventories as an inventory status variable, an inventory level, in order to simplify modeling and to avoid multiple orders during the replenishment period. In addition to tangible on-hand and on-order inventories, an abstract inventory can be established to handle certain modeling situations. For example, a backlog order is an abstract inventory used to handle shortages. When there is an inventory shortage, a situation where demand exceeds an inventory level, either a backlog order or lost sales is a common assumption in inventory modeling. We assumes that a customer will wait until the items arrive under a backlog assumption. A backlog assumption allows an inventory level to be negative to represent the unfulfilled demand. We assumes that a customer will go to another company and the excess demand is lost under a lost sale assumption. A lost sale assumption simply discards the unfulfilled demand, but the shortage may be recorded in order to measure a customer service level. Silver [109] mentions a substitution as another assumption for shortages. This assumption allows substitution for shortage items. However, this assumption is rarely seen in the more recent literature, with the exception of Karakul [68].

2.1.1 Economic Order Quantity

Economic Order Quantity (EOQ) is a dominant method in inventory control, as mentioned by Waters [122]. This method uses order quantities to determine replenishment orders. The order quantity is calculated to minimize cost for an inventory problem having a single item with a set up cost, a constant demand rate and a constant holding cost rate.

EOQ has several variations. The method described here is based on Waters [122], where an order quantity is considered as a combination of a cycle stock and a safety stock. For a cycle stock, a total cost C can be formulated as in Equation 2.1,

$$\begin{aligned} C &= \text{total reorder costs} + \text{total holding costs} \\ &= K \cdot D/Q + h \cdot Q/2 \end{aligned} \tag{2.1}$$

where K is a set up cost ($/order), D is a demand rate (units/week), Q is a replenishment size for each order(items) and h is a holding cost ($/unit for a week).

EOQ can be found by a derivative of a cost C with respect to an order size Q. The standard formula for EOQ is shown in Equation 2.2.

$$EOQ = Q = \sqrt{\frac{2 \cdot K \cdot D}{h}} \tag{2.2}$$

The length of a decision period, or a stock cycle, T_q can be simply calculated from $T_q = Q/D$. Since the replenishment requires a leadtime for delivery, it should be ordered when current stock will last until the next replenishment quantity arrives. A reorder level r is a stock level that signals when it is time to place a replenishment order. It is obtained from $r = L \times D$, where L is the leadtime. In general, when a leadtime is shorter than a stock cycle, the calculation $r = L \times D$ is sufficient. However, if a leadtime is longer than a stock cycle, it results in a reorder level that is greater than the highest stock level and consequently the reorder level will not be reached. For a case of $L > T_q$, a replenishment order has to be placed L div T_q cycle(s) earlier with the reorder level $r = (L \cdot D) \bmod Q$. The operators div and mod result, respectively, in a quotient and a remainder of their first argument divided by their second argument.

Originally EOQ was developed for deterministic problems, however a modification has been made to extend it to handle uncertainty by introducing a safety stock. (See Axsäter [6] for error bound of EOQ in stochastic problems.) A safety stock r_{ss} will not change the order quantity, but it will act as an offset for the reorder level, as shown in Equation 2.3.

$$r = (L \cdot D) \bmod Q + r_{ss} \tag{2.3}$$

A safety stock is used to balance a trade-off between holding cost and the possibility of inventory shortage. For a demand rate $\tilde{D}$ having a Normal distribution of mean D and variance σ^2, a safety stock can be obtained as shown in Equation 2.4.

$$r_{ss} = Z \cdot \sigma \cdot \sqrt{L} \tag{2.4}$$

A factor Z is used to control the possibility of shortage, e.g. $Z = 3$ allows about 0.1% chance of shortage within a stock cycle. Given $100 \cdot \alpha$ percentage of allowable shortage within the stock cycle, the value of z can be obtained from $z = N^{-1}(1 - \alpha)$ where $N^{-1}(\cdot)$ is an inverse cumulative distribution of a standard Normal distribution and $\alpha \in (0, 1)$.

2.1.2 (s,S) Policies

An (s,S) policy is a periodic review inventory policy where inventory level is reviewed at specific periods. If the level is at or below a reordering point s, a replenishment order of sufficient size will be placed to attain an inventory level of S.

An (s,S) policy is one of the most widely used inventory policies. It has many variations corresponding to different inventory problem structures. An (s,S) policy has been proved to be the optimum approach by using a concept of K-convexity. (See Simchi-Levi et al. [111] for detail.) Parameters of an (s,S) policy can be determined by dynamic programming.

For example[2], a stochastic stationary inventory problem in with zero leadtime and backlogging system has an objective function as shown in Equation 2.5.

$$\begin{aligned} C_t(x_t) &= \min_{y_t \geq x_t} E\left[K \cdot \delta(y_t - x_t) + c \cdot (y_t - x_t) + h^+ \cdot \max(y_t - D_t, 0) + h^- \cdot \max(D_t - y_t, 0)\right] \\ &\quad + E[\alpha \cdot C_{t+1}(y_t - D_t)] \\ &= \min_{y_t \geq x_t} R(x_t, y_t) + \alpha \cdot E[C_{t+1}(y_t - D_t)] \end{aligned} \tag{2.5}$$

where $C_t(x_t)$ is the expected cost accumulating since period t, x_t is an initial inventory level, y_t is an inventory level immediately after replenishment, K is a set up cost, $\delta(\cdot)$ is a step function defined as $\delta(a) = 1$ if $a > 0$ and $\delta(a) = 0$ if $a \leq 0$, c is a unit cost, D_t is the demand during period t, h^+ is a unit holding cost for a period, h^- is a unit shortage penalty cost for a period, α is a discounted factor, $E[\cdot]$ is the expectation over random demand, $R(x_t, y_t)$ is the expectation of a one-period cost and $C_{T+1}(\cdot) = 0$.

The operator $\min_{a \in A} f(a)$ is a minimization operator returning the minimum value of $f(a)$ by choosing value a from members of set A. Operator $\max(A, B)$ represents a maximum function returning a value of either A or B, whichever is larger. The optimal set of actions $\vec{y}^* = \{y_t^*, y_{t+1}^*, ..., y_T^*\}$,

[2] This presentation is based on Simchi-Levi et al. [111].

can be obtained by solving Equation 2.5. Equation 2.5 is calculated with $T-t+1$ variables. Given that a policy of the form (s,S) is optimal, Equation 2.5 can be simplified to Equation 2.6. Equation 2.6 is solved for only two variables: a reorder point s and an order-upto-level S. Under a stationary problem, an (s,S) policy provides a simpler calculation, especially when a decision horizon is long. The Bellman equation for the (s,S) policy is

$$C_t(x_t) = \min_{(s,S)} R(x_t, x_t + \pi(x_t; s, S)) + \alpha \cdot E[C_{t+1}(x_t + \pi(x_t; s, S) - D_t)] \tag{2.6}$$

where $\pi(x_t; s, S)$ is a policy function, shown in Equation 2.7, returning an order quantity. The (s,S) policy equation is

$$\pi(y_t; s, S) = \begin{cases} 0, & \text{if } y_t > s \\ S - y_t, & \text{otherwise.} \end{cases} \tag{2.7}$$

2.2 Inventory Studies

Hax and Candea [55] classified decisions into three levels: (1) a strategic level where decisions have a long-lasting effect; (2) a tactical level where decisions may be required weekly, monthly or quarterly, e.g. inventory policies; and (3) an operational level where decisions are made on a day-to-day basis. Inventory studies reviewed here focus on the tactical level of decisions to determine proper policies to manage an inventory level.

The use of operations research approaches to inventory management has been practiced for many years. The origin could be dated back to 1913 when Harris [52] developed the Economic Order Quantity (EOQ) approach. The EOQ approach is based on a deterministic model. Scarf [103] and Arrow [4] credit Arrow et al. [5] as pioneers of stochastic inventory research. Arrow et al. [5] studied 3 types of problems: deterministic problems and single-period and multi-period stochastic problems. For a multi-period stochastic problem with a set up cost, Arrow et al. used an analytical method to determine parameters of an (s,S) policy.

Since then, a wide variety of inventory problems have been studied. Inventory problems can be categorized by combinations of different characteristics. This categorization is not into mutually exclusive sets as the choice of certain characteristics may exclude some other types of included characteristics. For example, an infinite horizon problem can be multi-period, but it will not be a single-period problem because a horizon is specified in a single-period problem. The characteristics classified here are based on the survey works of Silver [109] and Silver [110] with some additions. These characteristics are listed as follows:

1. Multiplicity of items

   Inventory problems may be studied for either a case of single type of item or a case of multiple

types of items. The multiple-item problems may have extra characteristics: (i) a constraint on overall space or budget, (ii) an opportunity to save on fixed ordering costs through coordinated control, (iii) an opportunity to offer a substitution to a customer and (iv) dependency on different demands of different types of items.

2. A demand assumption

   Demand is a major source of uncertainty in inventory problems. Various types of demand have been studied: (i) deterministic demand, (ii) demand from a known probability distribution, e.g. Normal and Poisson distributions, (iii) demand from other special known distributions such as an intermittent distribution representing occasional large demands intermixed with small demands and (iv) Bayesian demands (assuming a known distribution, but with unknown parameters).

   Problems with deterministic demand, also known as lot size problems, are widely studied. Most studies in this category are based on an EOQ approach and its variations. Problems with known probability demand are also widely studied. Most studies in this category are based on problem specific analysis which requires very high analytical skill to make estimates. In addition to the demand assumptions mentioned above, recent inventory research uses approaches allowing a distribution-free assumption. Van Roy et al. [118], Godfrey and Powell [48], Kleinau and Thonemann [73], Kim et al. [70], Choi et al. [30], Topaloglu and Kunnumkal [117] and others have used such new approaches, like approximate dynamic programming and genetic programming, to solve inventory problems that require no assumption on a demand distribution.

3. Types of items held

   Types of items held are classified as (i) consumable, (ii) returnable or (iii) repairable. Most inventory studies investigate consumable items. Returnable items are products that can be returned after sale, where re-stocking and associated costs are accounted for in modeling. Repairable items are associated with maintenance items, e.g. spare parts. Sherbrooke [107] studied spare-part inventory problems for repairable items.

4. A decision period

   A decision period, also known as a decision epoch, is an inter-arrival time between two consecutive actions. There are two types of decision periods: (i) single periods and (ii) multiple periods.

A single-period problem is a problem where the effect of an action in one period has no effect on later periods. In this case inventory is ordered once at the beginning of a period, and inventory remaining at the end of the period cannot be used for a following period. The remaining inventory may be sold as scrap at a much lower price, be thrown away with no price or cost, or be disposed with a salvage cost.

A multiple-period problem is a problem where an effect of an action in one period will be carried over to a following period, i.e. remaining inventory from one period may still be used in subsequent periods.

5. Shelf life

   Items may be considered to be (i) perishable or (ii) non-perishable. A perishable or short shelf-life item may have a considerably shortened shelf life by becoming obsolete or having greatly dropped in price or quality. A non-perishable item has a very long shelf life. Note that a problem with short shelf-life items is different from a single-period problem. In a short shelf-life problem, replenishment inventory may be reordered multiple times before items run out of shelf life. In a single period problem, replenishment inventory is ordered once, or not ordered at all, at the beginning of decision period.

6. Continuity of time

   Time can be either (i) continuous or (ii) discrete.

7. Time dependency

   A problem may be assumed to be (i) stationary or (ii) time dependent. The latter has at least one parameter assumed to be a function of time.

8. Nature of supply

   Supply characteristics can vary from (i) a highly reliable nature with unlimited capacity and fixed leadtime, (ii) an uncertain delivery nature (for example, replenishment has a random leadtime with known distribution), (iii) an uncertain capacity nature (for example, supply capacity is random such that there is a chance that only part of a replenishment order can be delivered), (iv) a limited nature (for example, suppliers have a capacity restriction).

9. A supply constraint

   The supplier may have an order constraint e.g., a minimum order size, a fixed batch size.

10. A cost structure

    Ordering cost can have both (i) a fixed ordering cost per transaction, also known as a setup

cost, and (ii) a variable cost. Variable costs can be a fixed rate (e.g. cost per unit) or various fixed rates that are a function of order quantity (e.g. a large quantity discount). In addition to these structures, some inventory studies consider salvage cost, which is a cost to remove unwanted items. You [129] studied large quantity discount problems where the cost structure is a piecewise linear function.

11. Shortage handling

    When a shortage happens, the excess demands are taken either as (i) lost sales or as (ii) backlog orders, where it is assumed that the customer is willing to wait until the items are available.

12. Shortage control mechanism

    The chance of inventory shortage can be controlled by a mechanism of a penalty cost or a customer service level cost consequence.

13. Multiplicity of stocking points

    Stocking points of inventory can be structured as (i) a single stocking point, known as a single echelon, (ii) multiple stocking points, known as multi-echelon, or (iii) horizontal multiplicity, which is organized in hierarchy as multi-echelon but allowing transshipments among stocking points at the same echelon level.

14. A decision horizon

    Performance of inventory solutions can be considered over (i) a finite horizon or (ii) an infinite horizon. Performance of finite horizon problems will be considered over a specific number of periods. For example, the inventory may be reviewed every week and inventory cost of a whole year will be used to evaluate the inventory policy. In this case, the decision epoch is one week and the horizon is about 52 epochs. Performance of infinite horizon problems will be considered over infinite number of periods.

15. A discount factor in decision horizon

    A discount factor is a weighting factor of values over time. It is used as a mechanism to adjust values that occur in different time periods. A discount factor can be (i) one (no discounting effect), (ii) constant, between 0 and 1, or (iii) a function of time.

16. Inventory reviewing

    An inventory level is reviewed for replenishment needs. An inventory can be reviewed (i) periodically or (ii) continuously. A periodic-review system checks inventory at a particular time point e.g., the beginning of each epoch. If replenishment is issued, it will be issued at that

particular time. A continuous-review system continuously checks inventory and replenishment can be issued at any time.

17. Observable information

    From a modeling point of view, the information can be assumed to be (i) obtained perfectly (e.g. a perfect demand information), (ii) acquired partially (such as sales information which indicates censored demand information) and (iii) shared among partners.

18. Information acquisition

    Inventory information can be acquired (i) continuously or (ii) at a discrete point in time.

19. Marketing interaction

    Instead of treating demand as a given, there is ongoing research attempting to maximize profit by synchronizing inventory control with marketing campaigns. A variation can be (i) no interaction, considering marketing and inventory control independent, (ii) an attempt to regulate demand directly with marketing tools such as price or promotion or (iii) an attempt to affect demand information in advance by using an incentive for early ordering.

    Pricing strategy has been studied with inventory management as an approach to regulate demand to some degree. Petruzzi and Dada [92] reviewed pricing strategies for single-period inventory problems. Karakul [68] extended the Petruzzi and Dada results for problems with two item types. Federgruen and Heching [42] studied pricing strategies for multi-period problems. Chen and Simchi-Levi [27] studied an (s,S) policy and proposed their (s, S, A, p) policy for multi-period problems. These works formulated demand as a monotonic function of price with white noise error. Chen and Simchi-Levi [28] extended Chen and Simchi-Levi [27] to infinite horizon problems. They used a supremum limit[3] when the time period approached infinity to set the objective function. In addition to using price to influence demand, inventory on display can be exploited as a marketing tool. Gerchak and Wang [46] studied inventory problems where demand depended on a level of inventory on display. Instead of trying to influence demand by a pricing strategy, Özer and Wei [88] studied opportunities to use an incentive to get advanced demand information for better inventory planning. In addition to combining marketing and inventory management, Simchi-Levi et al. [111] integrated decision theory into inventory management and developed a risk averse inventory model to include risk aversion as an additional factor.

[3] Personal comment from the author is that Veinott and Wagner [119] had justified the use of infimum limit on a cost function. Therefore, it implicitly justified supremum limit on a profit function.

However, integration of marketing or a decision theory into inventory management results in complicated models. Pricing strategies here need to be applied carefully. Each type of demand has its own level of stimuli tolerance. For example, cheaper luxury consumer products may be able to attract more demand when there is a large potential for this. On the other extreme, cheaper fertilizer may not stimulate any more demands, when farmers have already fertilized their crops. Furthermore, in terms of marketing itself, pricing techniques should be applied cautiously. It may end up that downstream retailers overstock the promoted items, which may deceivingly show a monotonic relationship in the beginning, but cause an unpredictable demand pattern later. This marketing disturbance can cause amplification of demand variance upstream, known as a bullwhip effect, as systematically pointed out by Lee et al. [78].

An inventory problem is defined by a combination of the above characteristics. Some combinations are widely studied and have been uniquely named. A stationary problem with deterministic demand is also known as a lot size problem, or an economic lot size problem. The most widely known solution approach for a lot size problem is EOQ.

A non-stationary problem with deterministic demand is also known as a dynamic lot size problem, as indicated in works by Wagner and Whitin [121], Paterson and Silver [91], Buzacott [22], Brown [19] and Naddor [84].

A single-period problem is often referred to as a newsvendor, a newsboy, or a Christmas-tree problem. Khouja [69] provided a comprehensive survey for literature of this type. A problem with significant shelf-life items is also known as a problem with perishable items. Nahmias [85] studied inventory problems with perishable items. Zipkin [132], Porteus [94] and Simchi-Levi et al. [111] provided more recent reviews of these inventory problems.

One advantage of information sharing is to extend inventory management to larger and more complicated inventory systems (for example, a vendor-managed inventory (VMI) and supply chain management (SCM)). Both VMI and SCM are used to reduce the Bullwhip effect. The Bullwhip effect, made well known by Lee et al. [78], is an effect where demand variation of an inventory unit causes higher demand variation of an upstream partner of that inventory unit. For example, a retailer observing a surge from variation of an end customer demand wrongly perceives an uptrend demand and raises inventory with speculation of an upturn. Consequently, the wholesaler observing a surge in a retailer's order wrongly perceives a business upturn and raises inventory. Variation in end customer demand amplifies variation in the retailer's order; variation in a retailer order amplifies variation in a wholesaler order and so on.

The popularity of VMI and SCM is credited to Lee et al. [78] who investigated the Bullwhip effect in 1997, but earlier studies of inventory on a macro level by Caplin [24] in 1985 showed that order variance exceeded sales variance, contradicting the intuition that the retail inventory acts as a buffer between a producer and consumers.

The idea of SCM is relatively recent. The inventory aspect of SCM has been studied in the context of a multiple-stocking-point problem, also known as a multi-echelon problem, in Clark and Scarf [33] in 1960. Earlier, the multi-echelon inventory model is studied in the context of an intra-organization management. Later, the concept of information sharing makes corporate barriers more transparent and allows the model to be used for multiple inventories in a supply chain. Although information sharing can help reduce supply chain cost, Zhao and Xie [131] showed that it may increase costs for retailers, especially when there is high prediction error on the value of information sharing between retailers and a supplier. Hence it is important to consider every partner's costs as well as the entire chain cost.

In addition to the above characteristics, it is worth mentioning another management approach related to inventory management. Lean management, such as JIT, as discussed in Hirano and Furuya [60], accentuates the role of information and transportation in managing inventory flow such that a cost of holding inventory can be reduced dramatically or eliminated completely.

The original motivation of our study originated from recurring problems of inefficient inventory control of a small agrochemical-product distributor in Thailand. Inefficient inventory control caused the business to be short of cash and sometimes resulted in unnecessary loans. Characteristics of this inventory problem are described as follows.

- The inventory behavior usually is one-hierarchy, or single echelon.
- The distributor is not ready for a concept of extensive information sharing central to vendor-managed inventory and supply chain management models.
- Unlike on-shelf sales, the distributor takes purchase orders from downstream retailers. So, they generally can perceive all orders regardless of inventory level status, or at least they know when demand has exceeded their inventory. This situation provides more information than a censored data assumption studied by Godfrey and Powell [48]. Making a purchase order in this manner also decouples dependency between demand and inventory level. Therefore, demand can be considered to be independent from the effects of inventory levels, unlike a case studied by Gerchak and Wang [46].

- Costs, prices or marketing campaigns may be implemented and changed in a variety of ways. Although price can fluctuate and can be used as a marketing tool, it is indirectly regulated by fierce competition. Demand for agrochemical products can also tolerate some degree of price change without effect. For example, farmers do not want to over fertilize plants, or overstock fertilizers just because cheaper fertilizers are available. Hence, the price-demand relation in this case may not be simply estimated with a monotonic function, as in studies of Federgruen and Heching [42] and Chen and Simchi-Levi [27]. Furthermore, in terms of the marketing itself, pricing techniques should be cautiously applied, because it may only let downstream retailers overstock promotional items and may not be able to significantly stimulate real demand. Therefore, our study treats price as an exogenous factor.

- Since a supplier's capacity is usually much larger than the business volume of the distributor, supply can be assumed to be unlimited. In addition, because of the reliability of the delivery channel, a leadtime can be assumed to be a known constant.

2.3 Markov decision problems

A Markov decision problem is an optimization problem involving a Markov decision process or a Markov decision chain. A Markov decision process is a system having a sequence of interrelated decisions. It is a system such that the decision for one period does not only yield an immediate return for that period but also affects outcomes of later periods. A Markov process is a continuous-time system. A Markov chain is a discrete-time system. The immediate return may be referred to as a reward in a maximization problem or as a period cost in a minimization problem. Our study uses these terms: period cost, reward, one-period value, one-period cost and immediate return, interchangeably.

2.3.1 Dynamic Programming

White III [128] defined Dynamic Programming as a problem solving approach that can be effectively applied to mathematically describable problems having a sequence of interrelated decisions. There are many approaches to solving Markov decision problems. Dynamic programming is more efficient than direct search or linear programming. Bellman [13] is credited as a pioneer work on dynamic programming. White III [128] provided key references to dynamic programming including Bertsekas [14], Denardo [39], Heyman and Sobel [58], Hillier and Lieberman [59] and Ross [100].

Dynamic Programming uses an objective function to facilitate the search for the optimal policy. A policy is a rule to map a state to an action. An optimal policy is a policy that gives the optimal

value of the objective function over a specific horizon, regardless of an initial state. A general objective function can be written as Equation 2.8,

$$\begin{aligned} C_t(S_t) &= \sum_{i=t}^{T} \alpha^{i-t} \cdot R_i(S_i) \\ &= R_t(S_t) + \alpha \cdot C_{t+1}(S_{t+1}) \end{aligned} \tag{2.8}$$

where $C_t(S_t)$ is the total cost over all decision periods from t to T, S_t is a state at period t, $R_t(S_t)$ is a cost for period t for the given S_t, α is a discount factor and S_{t+1} is a next state which is a function of current state S_t.

A period cost of Equation 2.8 is $R_t(S_t)$. The effect of a decision on later periods may be referred to as a future value, a value-to-go, a future cost, or a cost-to-go. The cost-to-go of Equation 2.8 is $C_{t+1}(S_{t+1})$. Equation 2.8 is referred to as a Bellman equation or a Hamilton-Jacobi-Bellman equation. A stochastic version of this equation can be written as $C_t(S_t) = E\left[R_t(S_t) + \alpha \cdot C_{t+1}(S_{t+1})|S_t\right]$, where $E[\cdot|\cdot]$ is a conditional expectation on S_t (with respect to all states). In order to emphasize the role of the policy, the objective function can be written as Equation 2.9,

$$\begin{aligned} C_t^{\pi}(S_t) &= E[R_t^{\pi}(S_t)|S_t] + \alpha E\left[C_{t+1}^{\pi}(S_{t+1})|S_t\right] \\ &= r_t^{\pi}(S_t) + \alpha \sum_{s' \in S} p^{\pi}(s'|S_t) C^{\pi}(s') \end{aligned} \tag{2.9}$$

where $C_t^{\pi}(S_t)$ is a cost function when starting from period t with state S_t and using policy π, $r_t^{\pi}(S_t)$ is the expected cost of state S_t at period t given policy π and $p^{\pi}(s'|S_t)$ is the transition probability of state S_t to s' given policy π.

There are many dynamic programming algorithms. The algorithms described in the following are based on Powell [96], Sutton and Barto [114] and Puterman [98].

Finite Horizon Problems

In a finite horizon problem a number of decision periods is known and finite. The optimization equation can be written as Equation 2.10. The terminating value $C_T(S_T)$ is assumed to be obtainable (either $C_T(S_T)$ is known or it can be easily calculated). The equation is

$$\begin{aligned} C_t(S_t) &= \min_{a_t \in A} E[R_t(S_t, a_t) + \alpha C_{t+1}(S_{t+1})|S_t] \\ &= \min_{a_t \in A} \left\{ r_t(S_t, a_t) + \alpha \sum_{S_{t+1} \in S} p(S_{t+1}|S_t, a_t) \cdot C_{t+1}(S_{t+1}) \right\} \end{aligned} \tag{2.10}$$

where A is a set of possible actions, S is a set of possible states, $r_t(S_t, a_t)$ is an expected cost at period t for the given state S_t and the given action a_t and $p(S_{t+1}|S_t, a_t)$ is a transition probability

that a state S_t will become S_{t+1} when an action a_t is taken. It should be noted that equations presented here are for a maximization problem while the objective function shown in Equation 2.5 is for a minimization problem.

Table 2.1 illustrates the backward dynamic programming algorithm. The calculation starts from the last period value and is computed backward to period 0. Powell [96] suggested setting a model horizon T to be substantially larger than the actual decision horizon in order to improve the quality of the obtained optimal action a^*.

Table 2.1: Backward Dynamic Programming Algorithm

• Step 1: Initialization	Initialize $C_T(S_T)$. Set $t := T - 1$.
• Step 2: Evaluation	Calculate Equation 2.10 for all $S_t \in S$.
• Step 3: Loop and Termination	Check if $t > 0$, set $t := t - 1$ and go to step 1, otherwise stop.
• Result	Result is a sequence of optimal actions $a^* = \{a_0^*, a_1^*, ..., a_{T-1}^*\}$.

Equation 2.10 is a backward recursion. It can be formulated as a forward recursion as shown in Equation 2.11.

$$C_{k+1}(S_{k+1}) = \min_{a_{k+1}} \{R_{k+1} + \alpha \sum_{S_k} p(S_k|S_{k+1}, a_{k+1}) \cdot C_k(S_k)\} \tag{2.11}$$

where k is a number of period(s) before the terminating period.

Infinite Horizon Problems

In an Infinite Horizon Problem, a number of total decision periods is unknown or infinite. When a sequence $\{C_k\}$, of Equation 2.11, converges as $k \to \infty$, the subscript k can be dropped as shown in Equation 2.12,

$$C(s) = \min_{a \in A_s} \left\{ r(s,a) + \alpha \sum_{s' \in S} p(s'|s,a) C(s') \right\} \tag{2.12}$$

where s is a state, $s \in S$; s' is a state following state s; S is a set of states and A_s is a set of possible actions for the given state s.

Puterman [98] identified three classical algorithms to solve infinite horizon problems. They are (1) Value Iteration, (2) Policy Iteration and (3) Linear Programming for a Markov decision process.

Value Iteration: For each state s, the Value Iteration algorithm draws an arbitrary value from a set of feasible values of $C(s)$. Then it recalculates a value of $C(s)$ and repeats the calculation until a value of $C(s)$ converges. So $C(s) \approx \min_a\{r + \alpha \sum_{s'} p(s|s', a) \cdot C(s')\}$. An action a at equilibrium is the optimal solution. Table 2.2 shows a procedure of a Value Iteration algorithm. Since the optimal actions a is obtained with a tolerance of ϵ, this algorithm may be referred to as a ϵ-optimal stationary policy.

Table 2.2: Value Iteration Algorithm

• Step 1:	Initialization specify a tolerance $\epsilon > 0$ Initialize $c^{(0)}(s) \in C(s)$ for each $s \in S$. Set $n := 0$.
• Step 2:	For each $s \in S$ updating: $c^{(n+1)}(s) = \min_{a \in A_s} \left\{ r(s, a) + \alpha \sum_{s' \in S} p(s'|s, a) c^{(n)}(s') \right\}$ and $a^{(n+1)}(s) = \arg\min_{a \in A_s} \left\{ r(s, a) + \alpha \sum_{s' \in S} p(s'|s, a) c^{(n)}(s') \right\}$
• Step 3:	Loop and Termination Check if $\|c^{(n+1)} - c^{(n)}\| \geq \epsilon \frac{1-\alpha}{2\alpha}$, set $n := n + 1$ and go to step 2, otherwise stop.
• Result	The result with tolerance of ϵ is: the action $a^{(n+1)}$ giving the cost $c^{(n+1)}$.

$\|x\|$ is a euclidean distance, $\|x\| = \sqrt{x_1^2 + x_2^2 + ... + x_D^2}$
where x is a vector and its elements are $x_1, x_2, ..., x_D$.

Puterman [98] showed a justification for the termination condition, $\|c^{(n+1)} - c^{(n)}\| \geq \epsilon \frac{1-\alpha}{2\alpha}$, and a convergence proof for value iteration algorithms. A Gauss-Seidel variation of the value iteration algorithm improves the convergence rate by substituting equations in Step 2 of Table 2.2 with Equation 2.13 and 2.14. Rather than updating costs with calculated costs obtained in the last iteration, the Gauss-Seidel variation updates costs with the most recent calculated costs. The substitute equations are

$$c^{(n+1)}(s) = \min_{a \in A_s} \left\{ r(s, a) + \alpha \cdot \left(\sum_{s' < s} p(s'|s, a) c^{(n+1)}(s') + \sum_{s' \geq s} p(s'|s, a) c^{(n)}(s') \right) \right\} \tag{2.13}$$

and

$$a^{(n+1)}(s) = \arg\min_{a \in A_s} \left\{ r(s, a) + \alpha \cdot \left(\sum_{s' < s} p(s'|s, a) c^{(n+1)}(s') + \sum_{s' \geq s} p(s'|s, a) c^{(n)}(s') \right) \right\}. \tag{2.14}$$

Policy Iteration: The Policy Iteration algorithm uses a two-phase strategy. In phase 1, a Policy Iteration algorithm calculates cost functions for a given policy. This phase is called a policy evaluation. In phase 2, a Policy Iteration algorithm searches for the best policy based on calculated cost functions. This phase is called a policy improvement. The Policy Iteration algorithm calculates value functions and searches for the best policy based on the calculated values. With the new best policy, this algorithm recalculates cost functions, searches for the new best policy and repeats these two phases until a termination criterion is met.

Similar to Equation 2.12, the Bellman equation for the given policy π is written as shown in Equation 2.15,

$$C^{\pi}(s) = r^{\pi}(s) + \alpha \sum_{s' \in S} p^{\pi}(s'|s) C(s') \tag{2.15}$$

where $C^{\pi}(s)$ is a cost of state s given policy π, $r^{\pi}(s)$ is a period cost of state s given policy π, α is a discount factor and $p^{\pi}(s'|s)$ is a probability that the following state is s' given current state s and policy π.

Equation 2.15 can be written in an equilibrium matrix form, as in Equation 2.16. That is,

$$\begin{pmatrix} C^{\pi}(s_1) \\ C^{\pi}(s_2) \\ \vdots \\ C^{\pi}(s_{|S|}) \end{pmatrix} = \begin{pmatrix} r^{\pi}(s_1) \\ r^{\pi}(s_2) \\ \vdots \\ r^{\pi}(s_{|S|}) \end{pmatrix} + \alpha \cdot \begin{pmatrix} p^{\pi}(s_1|s_1) & p^{\pi}(s_2|s_1) & \cdots & p^{\pi}(s_{|S|}|s_1) \\ p^{\pi}(s_1|s_2) & p^{\pi}(s_2|s_2) & \cdots & p^{\pi}(s_{|S|}|s_2) \\ \vdots & \vdots & \ddots & \vdots \\ p^{\pi}(s_1|s_{|S|}) & p^{\pi}(s_2|s_{|S|}) & \cdots & p^{\pi}(s_{|S|}|s_{|S|}) \end{pmatrix} \cdot \begin{pmatrix} C^{\pi}(s_1) \\ C^{\pi}(s_2) \\ \vdots \\ C^{\pi}(s_{|S|}) \end{pmatrix} \tag{2.16}$$

where $|S|$ is a number of states.

Equation 2.16 can be simplified as,

$$\begin{aligned} C^{\pi} &= r^{\pi} + \alpha \cdot P^{\pi} \cdot C^{\pi} \\ &= (I - \alpha P^{\pi})^{-1} r^{\pi} \end{aligned} \tag{2.17}$$

where C^{π}, r^{π} and P^{π} are matrices of a state cost, a period cost, and a transition probability, respectively.

The Policy Iteration algorithm can also evaluate stationary values of state costs based on the matrix calculation in Equation 2.17. However, matrix calculations can be computationally expensive for a large state space. Puterman [98] and Powell [96] discussed this issue and its remedy. Table 2.3 shows the procedure of the Policy Iteration algorithm.

Linear Programming for Markov Decision Process: Linear Programming can be used to solve Markov decision problems. The Bellman equation can be expressed as a primal linear program as shown in Table 2.4. Hence any method developed for linear programming can be used to solve the problem.

Table 2.3: Policy Iteration Algorithm

• Step 1: Initialization	
	Initialize policy $\pi^{(0)} \in \Pi$ Set $n := 0$.
• Step 2: Policy Evaluation	
	obtaining $c^{(n)}$ by solving: $(I - \alpha P^{\pi})c^{(n)} = r^{\pi}$
• Step 3: Policy Improvement	
	Find $\pi^{(n+1)} = \arg\min_{\pi \in \Pi}\{r^{\pi} + \alpha P^{\pi} c^{(n)}\}$ If there is a tie, choose $\pi^{(n+1)}$ that is equal to $\pi^{(n)}$ if possible.
• Step 4: Loop and Termination	
	Check if $\pi^{(n+1)} \neq \pi^{(n)}$, set $n := n + 1$ and go to step 2, otherwise stop.
• Result	
	The result is: the policy $\pi^{(n+1)}$ giving the value $c^{(n)}$.

Π is a set of possible policies.

Table 2.4: Linear Program for Markov Decision Process

	$\min_{c(s)} \sum_{s \in S} \beta_s c(s)$
subject to:	
	$c(s) \leq r(s,a) + \sum_{s' \in S} \alpha p(s'\|s,a) c(s')$ for all s

In Table 2.4, $s \in S$ is a state. $a \in A_s$ is a feasible action for state s. β_s is a weighting factor for each state s. Weights β_s can be set to be equal for all s's when each state cost is equally important.

Puterman [98] identified the strength of linear programming for its comprehensively studied theories, sensitivity analysis and ease of positing constraints. However, both Puterman [98] and Powell [96] commented that linear programming was not an efficient approach for a Markov decision problem with large state or large action space.

2.4 Approximate Dynamic Programming

Approximate Dynamic Programming (ADP), also known as neuro-dynamic programming, reinforcement learning, and heuristic dynamic programming, is an approach to solving Markov decision problems with approximation techniques. It can be considered as a computational modification to overcome limitations of dynamic programming, such as the curse of dimensionality and the requirement of perfect information.

Exact dynamic programming can be used to solve Markov decision problems when the size of state-action space is finite and computationally practical. Higher dimensional state or action spaces cause methods in exact dynamic programming to suffer from the need for a great number of iterations. For example, the Value iteration algorithm has to iterate over all states, search for the optimal action from all actions, calculate an expectation over all possible next states and repeat these procedures until the termination criteria is met. This shortcoming of exact dynamic programming is often referred to as curse of dimensionality, as identified by Bellman [13].

Illustration of the curses of dimensionality. To illustrate curses of dimensionality, first consider an example of a simple one-echelon inventory system. Its state is modeled with a variable for inventory level. Its action is modeled with a variable for a replenishment order. Suppose a possible inventory level can be anything from -30 to 200 and the value of a replenishment order can be any value between 0 and 200. Then the size of state space will be 231 and the size of action space will be 201. It should be noted that it is common in inventory modeling to allow an inventory level to have a negative value. This negative inventory represents an inventory shortage and a promise to deliver the unfulfilled demand in the following period. (See Silver [109] for discussion on modeling an inventory shortage.)

If this problem is solved by a value iteration method (Table 2.2), then 231 objective functions of each state are calculated at each iteration until they converge. To calculate each objective function, all possible choices of action will be searched. Each choice of action will be evaluated with an expectation calculation. Each expectation calculation is a summation over probability products corresponding to each state. Roughly, there will be summations over 231 product terms for 46,431 times in each iteration. Suppose it takes 1,000 iterations to converge. The value iteration will be required to calculate an expectation 46,431,000 times. Each time 230 summations and 231 multiplications are required.

The above example shows the case of a simple one-echelon system. Consider a larger system, a two-echelon system of a single warehouse with two retailers that only receive their replenishment from the single warehouse. Each facility state is modeled with a variable for its inventory level. Each facility action is modeled with a variable for its replenishment order. A state of a whole system has three dimensions; each dimension corresponds to a single facility's inventory level. An action of a whole system has three dimensions; each corresponds to a single facility's replenishment order. Suppose a possible value of a retailer's inventory level is between -30 and 200; a value of a replenishment order can be between 0 and 200; a warehouse inventory level is between -100 and 500; a warehouse replenishment order is between 0 and 500. Hence the size of a system state space will

be $601 \times 231 \times 231 = 32{,}069{,}961$ and the size of a system action space will be $501 \times 201 \times 201 = 20{,}240{,}901$. Suppose it also takes the value iteration algorithm 1,000 iterations to converge. This problem requires calculation of expectation for about 646 trillion times and each time about 32 million summations and 32 million multiplications are required. If this value iteration is computed by the fastest computer[4] IBM Roadrunner at its 1.105 petaflop/s, it would take roughly about 37.42 million seconds, or 433 days to finish the calculation. Thus, brute force application of exact dynamic programming is not a practical approach for an inventory problem.

Table 2.5 further shows how the value iteration calculation required by single-echelon and two-echelon systems increases.

Table 2.5: Illustration of Curses of Dimensionality

System	# of facilities	size		computation required	
		state space	action space	summation	multiplication
single-echelon	1	231	201	1.0679E+10	1.0726E+10
two-echelon	3	32,069,961	20,240,901	2.0672E+22	2.0672E+22

The size of state or action space increases exponentially for each dimension added so that the calculation requirement increases dramatically for each extra dimension in either state or action space. This example illustrates how quickly the required computations grow given additional dimensions.

In addition to the curse of dimensionality, dynamic programming also requires an assumption of perfect information about the model, including transition probabilities $p(s'|s,a)$ (Equation 2.12) or P^{π} (Equation 2.17). In many problems, transition probability information might not be easily obtained. For example, in a stochastic inventory control problem, the transition probability is a function of demand, which is a random variable. In order to obtain the transition probability, an assumption of demand distribution has to be made and corresponding transition probability has to be analyzed.

Unlike exact dynamic programming, ADP uses approximations to overcome computational difficulties. There are two main approaches for these approximations investigated in our study. The first approach is learning-based ADP (Section 2.4.1). A learning-based method ADP uses a learning technique along with function approximation to provide approximate state-action costs. The second approach is simulation-based ADP (Section 2.4.4). A simulation-based ADP uses simulation to provide approximate state-action costs.

[4] Los Alamos lab claimed the title based on the 33rd edition of the list of the worlds TOP500 supercomputers released on June 23, 2009. source: http://www.lanl.gov/roadrunner

It should be noted that a mainstream study of learning-based ADP uses a learning scheme to approximate state-action cost and a decision is made based on the state-action cost. However, there is active research on policy gradient, e.g. Baxter and Bartlett [11], Baxter et al. [12] and Sutton et al. [115], attempting to shortcut this mainstream procedure by using a learning scheme to update policy parameters directly without approximation of state-action cost.

Look-Ahead: Look-Ahead, also known as Rolling-horizon, is a simple method to control a Markov decision process based on future projection. It is often used as a benchmark for other methods in inventory management research. Look-Ahead uses a problem model to project events ahead and chooses actions to minimize the projected cost of a specific number of period(s) ahead. Then the action taken is the first action in a sequence obtained from a Look-Ahead search (shown in Equation 2.18) or $a_t = a^{(1)}$,

$$\vec{a} = \arg\min_{\vec{a}} \sum_{i=1}^{H} \alpha^{i-1} \bar{c}(\bar{s}_{t+i-1}, a^{(i)}) \tag{2.18}$$

where $\vec{a} = [a^{(1)}\ a^{(2)}\ a^{(3)}\ \ldots\ a^{(H)}]'$, $\bar{c}(s,a)$ is a projected period cost, $\bar{s}_{t+1}$, ..., $\bar{s}_{t+H-1}$ are projected states at time $t+1$, ..., $t+H-1$ respectively and $\bar{s}_t$ is an initial state ($\bar{s}_t = s$).

2.4.1 Learning-based ADP

Learning-based ADP, often called Reinforcement Learning (RL), uses an approximation function to assist with calculating the objective function or its variations. An approximation function estimates the objective function or its variations[5] from data either in a batch or incremental mode. In a batch learning scheme the approximation function uses a batch of data to approximate the target value. In an incremental learning scheme, the approximation function uses ongoing data from continuous action to improve the estimate.

A learning-based ADP has three general characteristics (1) a balance between exploitation and exploration, (2) a choice of how an algorithm is set up and (3) a choice of function approximation. These characteristics are not mutually exclusive. A good balance between exploration and exploitation can improve the quality of function approximation. How an algorithm is set up can improve computational efficiency and can reduce accuracy required from approximation without compromising quality. For example, policy approximation requires a less accurate approximation function than value approximation, as discussed by Anderson [2].

[5] Instead of approximating an objective function value, Baxter and Bartlett [11] approximated policy parameters directly from estimated gradient of an objective function.

Balance between exploitation and exploration

A learning-based ADP optimizes an objective function based on approximated information, as it simultaneously improves the approximation from chosen actions. However, the chosen action is determined by approximate information. Unless the action space is well explored, approximate state-action costs might not accurately represent the real values of state-action costs. With poor approximation, an action decision might not be close to an optimal action. The relation between a decision, which is based on obtained information, and information observation, which in turn is influenced by a previous decision, raises the issue of balancing exploitation and exploration. Exploitation is utilization of approximate values for a decision purpose. Exploration is a mechanism to improve the quality of approximate values. For exploitation, the decision is to choose an action expected to give optimal value. For exploration, the decision is to choose an action rarely tried[6]. Both exploitation and exploration are important components in learning-based ADPs. There are many ways to add exploration into ADP. A greedy policy, an ϵ-greedy policy, and a softmax policy are all general policies often seen in the ADP literature.

A greedy policy is a policy that always selects an action that seems to get the optimal return. This policy does not have exploration, because it always chooses an action estimated to give the optimal return without exploring other choices. A greedy policy can be formulated as Equation 2.19,

$$\pi(s,a) = \begin{cases} 1, & \text{if } a = \arg\min_a \ Q(s,a) \\ 0, & \text{if } a \neq \arg\min_a \ Q(s,a) \end{cases} \tag{2.19}$$

where $\pi(s,a)$ is a probability of selecting action a for the given state s and Q is an approximate state-action cost.

An ϵ-greedy policy is a policy that selects an action $\hat{a} = \arg\min_a \ Q(s,a)$ with probability $1-\epsilon$ and any action a, including $\hat{a}$, with probability ϵ. It can be written as Equation 2.20,

$$\pi(s,a) = \begin{cases} 1 - \epsilon + \frac{\epsilon}{|A(s)|}, & \text{if } a = \hat{a} \\ \frac{\epsilon}{|A(s)|}, & \text{if } a \neq \hat{a} \end{cases} \tag{2.20}$$

where $|A(s)|$ is a number of possible actions for the given state s.

A softmax policy is a policy that selects an action with a probability proportional to its corresponding approximate value. A softmin policy is an equivalent policy for a minimization problem. In a softmin policy, if an action is estimated to result in a lower cost, it has a higher chance of being selected. Equation 2.21 shows a softmin policy,

$$\pi(s,a) = \frac{\exp\left(-Q(s,a)/\tau\right)}{\sum_{b \in A(s)} \exp\left(-Q(s,b)/\tau\right)} \tag{2.21}$$

[6] This has meaning in a loose sense. In implementation, it may only have a probability such that every action has a chance to be chosen.

where τ is a parameter to control a degree of exploration and a higher value of τ gives a greater degree of exploration.

For a problem with numerical actions, e.g., replenishment orders, noise can be used to add a degree of exploration. The action taken can be obtained from $\epsilon + \arg\min_a Q(s, a)$, where ϵ is random noise such that the final action is still feasible. A distribution of noise ϵ controls the degree of exploration.

How the algorithm is set up

Learning-based ADPs are set up in different ways. Since ADP is studied by people from different fields, ADP algorithms are classified by different perspectives. Sutton and Barto [114] classify an algorithm setup by how a cost function is calculated[7]. Dynamic Programming uses an expectation to calculate a cost function. An expectation requires a transition probability to evaluate all possible following states and their corresponding costs or cost-to-go, e.g., $C_{t+1}(S_{t+1})$ in Equation 2.8. Monte Carlo is a method that uses a sampling approach to predict a target value of state cost. An average of sampled values is used as an estimation of a target value. The Monte Carlo approach calculates a target value by using only sampled data and it does not require a transition probability. However, in a Markov decision problem, a total cost is determined over multiple periods. Here the Monte Carlo approach has to sample a total cost in order to be able to evaluate a state cost. This means that the Monte Carlo approach will need to wait until some total costs are known before it can average those values and use the average as an approximation of the state cost.

The most widely used Monte Carlo method is one that is called a First-visit method. A First-visit Monte Carlo method for estimating state cost is shown in Table 2.6.

The return $R_{s^{(1)}}$ is a combination of all period costs after state s occurs the first time. So $R_{s^{(1)}} = \sum_{t=t_s}^{T} r_t$ where r_t is a cost at period t and t_s is the first time state s occurs. A trajectory, also known as an episode, is a series of events that is terminated, where total cost is known. A trajectory can be obtained by any observation tool in a real-world or computer simulation. It provides information on states visited, actions taken, and corresponding period costs in sequence. Period costs recorded in a trajectory can be used to calculate total cost.

Monte Carlo methods can be used with a control policy to optimize Markov decision problems. Monte Carlo control uses a two-phase strategy to combine value estimation and policy improvement. Table 2.7 shows a combination of Monte Carlo value estimation with an ϵ-greedy policy. In Table 2.7, $R_{(s,a)^{(1)}}$ is a return following the first occurrence of s and a.

[7] Sutton and Barto [114] used a term "backup", when they referred to how a cost function is calculated.

Table 2.6: First-visit Monte Carlo method for estimating cost function

- Step 1: Initialization
  - Initialize $C(s)$ as an arbitrary value
  - Initialize return record $\hat{R}(s)$ as an empty list for all $s \in S$
- Step 2: Repetition
  - Obtain a series of states s and actions a in an observed trajectory
  - For each state s appearing in the trajectory:
    - $R(s) \leftarrow R_{s^{(1)}}$
    - Append $R(s)$ to $\hat{R}(s)$
    - $C(s) \leftarrow average(\hat{R}(s))$
  - Repeat step 2
- Result
  - The result is:
    - the estimated value $C(s)$

$R_{s^{(1)}}$ is a return following the first occurrence of s.
(This table is based on Sutton and Barto [114]: First visit MC method for estimating C^{π}.)

Table 2.7: ϵ-greedy on-policy Monte Carlo control

- Step 1: Initialization
  - Initialize record $\hat{R}(s,a)$ as an empty list for all $s \in S$ and $a \in A(s)$
- Step 2: Repetition
  - Given policy π, obtain a series of states s and actions a
  - Evaluate policy cost: for each pair s and a appearing in the episode:
    - $R(s,a) \leftarrow R_{(s,a)^{(1)}}$
    - Append $R(s,a)$ to $\hat{R}(s,a)$
    - $Q(s,a) \leftarrow average(\hat{R}(s,a))$
  - Improve policy: for each s in the episode:
    - $a^* = \arg\min_a Q(s,a)$
    - For all $a \in A(s)$:

$$\pi(s,a) \leftarrow \begin{cases} 1-\epsilon+\frac{\epsilon}{|A(s)|}, & \text{if } a = a^* \\ \frac{\epsilon}{|A(s)|}, & \text{if } a \neq a^* \end{cases}$$

  - Repeat step 2
- Result
  - The result is:
    - the optimal policy π

(This table is based on Sutton and Barto [114]: an ϵ-greedy on-policy Monte Carlo control.)

Table 2.8: Sarsa algorithm

• Step 1: Initialization	
	Initialize $Q(s, a)$ for all states and actions
• Step 2:	
	Initialize state s
	Choose action $a \leftarrow \pi(s\|Q)$
	Repeat until s is the terminal state
	Take action a, observe period cost r and next state s'
	Choose action $a' \leftarrow \pi(s'\|Q)$
	Update action-state cost Q by:
	$Q(s, a) \leftarrow Q(s, a) + \rho \cdot (r + \alpha \cdot Q(s', a') - Q(s, a))$
	Move to next step: $s \leftarrow s'$ and $a \leftarrow a'$
• Result	
	The result is:
	the approximate state-action cost Q and
	a series of actions a, decided by policy π

$\pi(s|Q)$ is a policy function taking state s and returning action a for the given state-action cost Q.
ρ is an improvement step size, also known as a learning rate.
α is a discount factor.
(This table is based on Sutton and Barto [114]: Sarsa: An on-policy TD control algorithm.)

A Monte Carlo approach does not require a complete model, but, when applying it to Markov decision problems, it has to wait until the end of a trajectory when the total cost is available. A One-step Temporal-Difference method, abbreviated as TD(0) or TD0, is a method that, in each decision period, uses a period cost recently observed and a current estimate to improve the estimation. So, TD(0) does not have to wait until a trajectory is over to improve its approximation. TD(0) improves the quality of its approximation in every time step. The Sarsa algorithm, as shown in Table 2.8, is a well-known implementation of TD(0).

Sarsa is an on-policy algorithm because it uses an actual chosen action to update a state-action cost. In an on-policy algorithm the action for updating a state-action cost is always the same as the actual action that will be taken. The algorithm, shown in Table 2.9, is an off-policy algorithm because it uses a greedy action for a state-action cost update. The greedy action for updating a state-action value may or may not be the same action that will be taken.

A policy determining an actual action is called a behavior policy. A policy determining a speculated action for state-action cost update is called an estimation policy. In Table 2.9, the behavior policy of Q-learning is π, which can be any policy, but its estimation policy is a greedy policy. It should be noted that Baird [7] showed that off-policy algorithms may be unstable when used with function approximation.

Table 2.9: Q-learning algorithm

• Step 1: Initialization
Initialize $Q(s, a)$ for all states and actions
• Step 2:
Initialize state s
Repeat (for each step of episode) Until s is the terminal state
Take action $a \leftarrow \pi(s\|Q)$ calculated with current values of Q
Observe period cost r and next state s'
Update state-action cost by:
$Q(s, a) \leftarrow Q(s, a) + \rho \cdot (r + \alpha \cdot \min_{a'} Q(s', a') - Q(s, a))$
Move to next step: $s \leftarrow s'$
• Result
The result is:
the estimated state-action cost Q and
series of estimate optimal actions a decided by policy π

(This table is based on Sutton and Barto [114]: Q-learning: An off-policy TD control algorithm.)

The multiple-step Temporal-Difference method, abbreviated as TD(λ), is similar to the one-step temporal-difference method, but instead of improving an approximation one step at a time, it improves the approximation based also on multiple prior steps.

Figure 2.1 illustrates how information about a one-period reward of period t, as a result of state s_t, can be used to improve an approximation of the value of state s_t, as in TD(0).

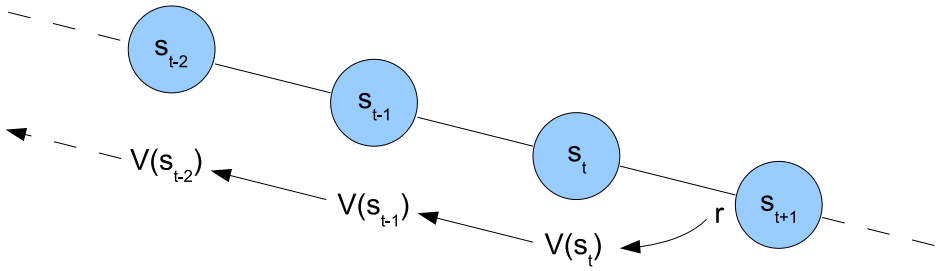

Figure 2.1: Reward and its back tracing (based on Sutton and Barto [114, backward view])

The new value of state s_t can subsequently be used to improve approximations of values of prior states $s_{t-1}, s_{t-2}, s_{t-3}$ and so on. So the current result can be used to improve the cost approximation of multiple states that sequentially lead to the new result. Since state s_t is closer to the result r_t, it should be more heavily weighted than any other prior states. Hence the closer the state to the result, the more weighting it should receive for the result. An Eligibility Trace of each state is defined such a way that its value will be greatest if the information is a direct consequence of that state, and

then the value of an Eligibility Trace will decrease by each earlier time step.

A state Eligibility Trace[8] can be formulated as $e_t(s) = \alpha \cdot \lambda \cdot e_{t-1}(s)$ when $s \neq s_t$ and $e_t(s) = 1$ when $s = s_t$. Singh and Sutton [112] proposed a state-action Eligibility Trace for control application. This state-action Eligibility Trace is similar to a typical state Eligibility Trace, but it has an additional rule for actions. Equation 2.22 shows this type of state-action Eligibility Trace,

$$e_t(s,a) = \begin{cases} 1, & \text{if } s = s_t \text{ and } a = a_t \\ 0, & \text{if } s = s_t \text{ and } a \neq a_t \\ \alpha \cdot \lambda \cdot e_{t-1}(s,a), & \text{if } s \neq s_t \end{cases} \tag{2.22}$$

where $e_t(s,a)$ is an Eligibility Trace at period t of state s and action a, s_t is a state at period t, a_t is an action taken in period t, α is a discount factor and λ is an eligibility factor, also known as a trace-decay parameter.

An eligibility factor λ controls the degree of back tracing weights. A higher λ weights prior states more heavily than a lower λ. When $\lambda = 0$, an Eligibility Trace credits only one step prior and is equivalent to TD(0). The Eligibility Trace approach is an extension of the temporal difference learning technique. A one-step temporal difference learning approach is denoted TD(0) and an Eligibility Trace is denoted TD(λ), where λ is the eligibility factor. An Eligibility Trace version of Sarsa, called Sarsa(λ), is shown in Table 2.10.

It should be noted that $Q(s,\hat{a})$ is an approximate state-action cost where state s is the current state but action $\hat{a}$ is not the current action. It does not get updated because it may take a different action and should not be given any weights from the consequence of state s and action a.

Unlike classification by the method of calculating a cost function suggested by Sutton and Barto [114], Werbos [126] viewed classification by how the algorithm is set up using a starting point or an objective function. A state cost function, $C^\pi(s)$, is used as an objective function in methods such as Dynamic Programming. An alternative can be a state-action cost function, $Q^\pi(s,a)$, that is used in methods such as Sarsa and Q-learning. In addition to state cost and state-action cost, a derivative of cost-to-go can be used to improve policy in a Markov decision problem. An algorithm based on a derivative of cost-to-go is referred to as Dual Heuristic Programming (DHP), as introduced by Werbos [125]. Venayagamoorthy et al. [120] provided a comparison of DHP with other methods using a Turbogenerator application. Shervais et al. [108] applied DHP to a combined inventory-transportation problem. Baxter and Bartlett [11] proposed a similar approach, but they emphasized its role as a policy approximation compared to a cost approximation used by most researchers. Anderson [2] discussed an advantage of policy approximation over cost approximation. In addition to state cost, state-action cost, and a derivative, a hybrid design can be used. A hybrid design

[8] Eligibility Trace, presented here is a replacing Eligibility Trace[114].

Table 2.10: Sarsa(λ) algorithm with replacing Eligibility Trace

- Step 1: Initialization
  - Initialize $Q(s, a)$ for all states and actions
  - Initialize $e(s, a) \leftarrow 0$ for all s, a
- Step 2:
  - Initialize state s
  - Choose action $a \leftarrow \pi(s|Q)$
  - Repeat until s is the terminal state
    - Take action a, observe period cost r and next state s'
    - Choose action $a' \leftarrow \pi(s'|Q)$
    - $\delta \leftarrow r + \alpha \cdot Q(s', a') - Q(s, a)$
    - $e(s, a) \leftarrow 1$
    - $Q(s, a) \leftarrow Q(s, a) + \rho \cdot \delta$
    - For all actions $\hat{a} \neq a$
      - $e(s, \hat{a}) \leftarrow 0$
    - For all states $\hat{s} \neq s$ and all actions $\hat{a}$
      - $e(\hat{s}, \hat{a}) \leftarrow \alpha \cdot \lambda \cdot e(\hat{s}, \hat{a})$
      - $Q(\hat{s}, \hat{a}) \leftarrow Q(\hat{s}, \hat{a}) + \rho \cdot \delta \cdot e(\hat{s}, \hat{a})$
    - Move to next step: $s \leftarrow s'$ and $a \leftarrow a'$
- Result
  - The result is:
    - the approximate state-action cost Q and
    - a series of approximate optimal actions a decided by policy π

is a combination of multiple ADP systems with other systems. That is the use of hybrid control, artificial intelligence, and spatial structure, in order to scale solutions up to larger problems. (See Werbos [126] for discussion about hybrid design.)

2.4.2 Function Approximation

Learning-based ADPs use a lookup table as a method to store predicted cost function. In some circumstances, however, state or state-action cost functions may not be represented efficiently by a simple lookup table. For example, the state-action space may be too large or it may be continuous. In such circumstances, an approximation function is often used.

There are many available approximation functions that are either parametric or non-parametric. A parametric function is based on a model with a fixed number of parameters and this number is generally small. This is often referred to as a model-based function. A non-parametric function is based on a model with a variable number of parameters and the number can be large. Geman et al. [44] studied the approximation approach and its associated errors. They classified approximation error into its components of bias error and variance error. A bias error is the expected squared difference between the expected target values and their approximate values with respect to training samples. A variance error is the variance of the approximation function itself with respect to training

samples. A good approximation function requires the error levels to be small. (See Geman et al. [44] for discussion about bias/variance dilemma.)

With a limited number of parameters, a parametric function has limited flexibility. Therefore, a parametric function model has to be carefully selected to accurately represent a target variable. When an approximation function cannot estimate a target value well, it causes a high bias estimation error.

With a changeable number of parameters, a non-parametric function has substantial flexibility. It can be tuned to have a low bias error. It should be noted that it also can be tuned to have a zero bias error, but this approach comes with a risk of an extremely high variance error. A choice of a model in this case is not as critical as the case of a parametric model. However, a non-parametric function is susceptible to a high variance error because there are more free parameters to be determined. Consequently, to minimize bias and variance errors, a non-parametric function generally requires more data and greater computation effort and a mechanism such as cross-validation, regularization or a bayesian approach to prevent overfitting. Overfitting happens when an approximation function is tuned to fit a specific data sample but poorly represents another sample from the same data. (See Bishop [18] for detail of overfitting and its remedies.)

There are many well-studied general non-parametric functions and they can be classified by how the output value relates to adjustable parameters. A linear basis approximation function has an output value of a linear combination of weighted bases. Parameter values of a linear basis approximation function are relatively easy to determine. An artificial neural network is the most widely used general non-linear nonparametric function. Its output is not a linear combination of parameters. A neural network is able to represent an arbitrary complex function with fewer parameters than a linear nonparametric function at the same accuracy, as discussed by Werbos [126]. However, determining parameters of a neural network is more difficult than determining parameters of a linear approximation function.

Among non-parametric functions, a two-layer feedforward neural network (FFNet) and a radial basis function (RBF) are among the most widely used approximation functions. FFNet has been shown to be a universal approximation function by Cybenko [35], Hornik et al. [61] and Barron [10]. RBF has been shown to be a universal approximation function by Powell [95] and Park and Sandberg [89]. A universal approximation function is able to approximate an arbitrary continuous function when there is an adequate number of parameters.

Two-layer Feedforward Neural Network

A Feedforward Neural Network (FFNet)[9], also known as a Multilayer Perceptron Neural Network (MLP), is a general non-parametric approximation function. The FFNet variation presented here is a two-layer version. A two-layer FFNet is a weighted summation of nonlinear elements that are hyperbolic tangent functions of weighted summations of the inputs. (See Hagan et al. [51] and Bishop [18] for general discussion about an Artificial Neural Network.) Equation 2.23 shows output $\vec{y} \in \Re^K$ of FFNet of M hidden units for the given input $\vec{x} \in \Re^D$,

$$y_k = w_{0,k} + \sum_{m=1}^{M} w_{m,k} \cdot h\left(v_{0,m} + \sum_{d=1}^{D} v_{d,m} \cdot x_d\right) \qquad \text{for } k = 1, \ldots, K \tag{2.23}$$

where $v_{d,m}$ and $w_{m,k}$ are function parameters for $d = 0, \ldots, D$, $m = 0, \ldots, M$ and $k = 1, \ldots, K$ and $h(x)$ is a hyperbolic tangent ($h(x) = \tanh(x)$).

To determine FFNet parameters, a data set is required. This data set is usually referred to as training data. Given samples of training data, parameters of FFNet can be determined by a method of error backpropagation, as discussed by Rumelhart et al. [101]. Parameter values of the v's and w's are determined to minimize a mean squared error between training data outputs and FFNet approximate outputs. It should be noted that a method of maximum likelihood is a widely-used alternative to determine these parameters.

Equation 2.24 shows the (half) squared error for output of n^{th} sample. Equation 2.25 shows the total squared error of all N samples.

$$\xi_n = \frac{1}{2} \sum_{k=1}^{K} \left(y_k(\vec{x}_n) - t_{k,n}\right)^2 \tag{2.24}$$

$$\xi = \sum_{n=1}^{N} \xi_n \tag{2.25}$$

where $\vec{x}_n$ is the training data input of the n^{th} sample (each input has D dimension(s) or $x_n = [x_{n,1} \ldots x_{n,D}]$), $t_{k,n}$ is the training data output at k^{th} dimension of the n^{th} sample and $y_k(\vec{x}_n)$ is the FFNet approximate output at k^{th} dimension when taking the input $\vec{x}_n$.

[9] Material for FFNet presented here is mainly based on Anderson [3].

Partial derivative of two-layer networks are shown in Equation 2.26 and 2.27.

$$\begin{aligned}
\frac{\partial \xi_n}{\partial w_{m,k}} &= (y_k(\vec{x}_n) - t_{k,n}) \cdot \frac{\partial y_k(\vec{x}_n)}{\partial w_{m,k}} \\
&= (y_k(\vec{x}_n) - t_{k,n}) \cdot \begin{cases} 1 & \text{for } m = 0 \\ z_m(\vec{x}_n) & \text{for } m = 1, \ldots, M \end{cases} \\
&= (y_k(\vec{x}_n) - t_{k,n}) \cdot z_m(\vec{x}_n) \qquad (2.26) \\
\frac{\partial \xi_n}{\partial v_{j,m}} &= \sum_{k=1}^{K} (y_k(\vec{x}_n) - t_{k,n}) \cdot w_{m,k} \cdot \frac{\partial z_m(\vec{x}_n)}{\partial v_{j,m}} \\
&= \sum_{k=1}^{K} (y_k(\vec{x}_n) - t_{k,n}) \cdot w_{m,k} \cdot h'\left(v_{0,m} + \sum_{d=1}^{D} v_{d,m} \cdot x_{n,d}\right) \cdot \begin{cases} 1 & \text{for } j = 0 \\ x_{n,j} & \text{for } j = 1, \ldots, D \end{cases} \\
&= \sum_{k=1}^{K} (y_k(\vec{x}_n) - t_{k,n}) \cdot w_{m,k} \cdot h'\left(v_{0,m} + \sum_{d=1}^{D} v_{d,m} \cdot x_{n,d}\right) \cdot x_{n,j} \qquad (2.27)
\end{aligned}$$

where $z_m(\vec{x}_n) = h\left(v_{0,m} + \sum_{d=1}^{D} v_{d,m} \cdot x_{n,d}\right)$ for $m = 1, \ldots, M$, $z_0(\vec{x}_n) = 1$, $h(a)$ is a transfer function, $h'(a)$ is a derivative of $h(a)$ and $x_{n,d}$ is the n^{th} sample input at d^{th} dimension, with $x_{n,0} = 1$.

Parameters v's and w's can be determined in a batch mode as shown in Equation 2.28 and 2.29. Alternatively, they can be determined in an increment mode as shown in Equation 2.30 and 2.31, where each batch update uses all data and incremental update uses one sample at a time.

Batch mode:

$$v_{j,m} \leftarrow v_{j,m} - \rho_h \cdot \frac{1}{N} \sum_{n=1}^{N} \sum_{k=1}^{K} \{(y_k(x_n) - t_{k,n}) \cdot w_{m,k}\} h'\left(v_{0,m} + \sum_{d=1}^{D} v_{d,m} \cdot x_{n,d}\right) \cdot x_{n,j} \qquad (2.28)$$

$$w_{m,k} \leftarrow w_{m,k} - \rho_o \cdot \frac{1}{N} \sum_{n=1}^{N} (y_k(x_n) - t_{k,n}) \cdot z_m(x_n) \qquad (2.29)$$

Increment mode:

$$v_{j,m} \leftarrow v_{j,m} - \rho_h \sum_{k=1}^{K} \{(y_k(x_n) - t_{k,n}) \cdot w_{m,k}\} h'\left(v_{0,m} + \sum_{d=1}^{D} v_{d,m} \cdot x_{n,d}\right) \cdot x_{n,j} \qquad (2.30)$$

$$w_{m,k} \leftarrow w_{m,k} - \rho_o \cdot (y_k(x_n) - t_{k,n}) \cdot z_m(x_n) \qquad (2.31)$$

where ρ_h and ρ_o are step sizes.

In these formula, step sizes ρ_h and ρ_o must be selected. Small step size will cause slow convergence. Large step size will cause unstable FFNet output.

Equations 2.28, 2.29, 2.30 and 2.31 are based on the Gradient Descent (GD) method. Batch mode methods that are more efficient than the GD method include Levenberg-Marquardt (LM) as discussed by Hagan and Menhaj [50]; Resilient Backpropagation (RP) as discussed by Riedmiller and Braun [99]; and scale conjugate gradient (SCG) as discussed by Moller [83]. According to speed

and memory comparison and recommendation, as discussed in MathWorks document[10], SCG stands out among batch update methods because it performs efficiently across a wide range of applications and can be used effectively with early stopping, a widely used method to prevent overfitting.

In addition to efficiency, while GD's step size has to be chosen and is critical to the performance of the GD method, SCG has no critical parameter as claimed by its inventor, Moller [83]. An SCG method performs so well that, although it was designed for batch update, Falas and Stafylopatis [41] attempted to apply this method in an increment mode with TD(0). However, the results were unsatisfactory.

Radial Basis Function

A Radial Basis Function Network, or Radial Basis Function (RBF), is another general non-parametric approximation function. The RBF used in our study is the RBF with Gaussian bases. It is a weighted summation of Gaussian functions of scaled distances between inputs and centers. Weights, scales and centers are parameters. Barreto and Anderson [9] mentioned the RBF as a frequent choice for an ADP approximation function. Equation 2.32 shows output $\vec{y} \in \Re^K$ for the given input $\vec{x} \in \Re^D$ when RBF has M hidden units,

$$y_k = w_{0,k} + \sum_{m=1}^{M} w_{m,k} \cdot \phi\left(\|\vec{x}_n - \vec{v}_m\|_{\mathbf{Z}_m}\right) \qquad \text{for } k = 1, \ldots, K \tag{2.32}$$

$$\mathbf{Z}_m = \begin{bmatrix} z_{1,m} & 0 & \cdots & 0 \\ 0 & z_{2,m} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & z_{D,m} \end{bmatrix} \tag{2.33}$$

$$\begin{aligned} \|\vec{x}_n - \vec{v}_m\|_{\mathbf{Z}_m} &= \sqrt{(\vec{x}_n - \vec{v}_m)^{\mathrm{T}} \cdot \mathbf{Z}_m \cdot (\vec{x}_n - \vec{v}_m)} \\ &= \sqrt{\sum_{d=1}^{D} z_{d,m} \cdot (x_{n,d} - v_{d,m})^2} \end{aligned} \tag{2.34}$$

where $\mathbf{Z}_m$, $\vec{v}_m$, $w_{0,k}$, and $w_{m,k}$, for $m = 1, \ldots, M$ and $k = 1, \ldots, K$, are parameters and $\phi(a)$ is a radial basis function, which for our study is Gaussian or $\phi(a) = \exp(-a^2)$. It should be noted that $\mathbf{Z}_m$ is a $D \times D$ matrix. The notation $z_{d,m}$ represents the d^{th} element along diagonal of Z_m matrix.

Compared to FFNet, RBF parameters are easier to determine. Clustering techniques, such as the K-means algorithm, as discussed in MacQueen [81], and Self-organizing map, as discussed in Kohonen [74], can be used to set up centers. Scales and weights can then be determined for the designed centers.

[10] 1984-2006 The MathWorks, Inc.

K-means: A K-means algorithm is a widely used clustering method. Given a pre-specific number of groups, it clusters multidimensional data points into groups such that differences among data points in the same group are minimized. The result from a K-means approach is a group assignment of data points and group centers. The data points of a K-means assignment are RBF inputs $\vec{x}_n$ for $n = 1, ..., N$. The K-means group centers will be used as RBF centers.

Given the number of clusters κ, the K-means procedure starts with κ centers, called centroids. Each centroid corresponds to each cluster and has the same dimension as the data points. These initial centroids can be assigned to any κ sampled data points drawn randomly from the entire data set $\{\vec{x}_1, \ldots, \vec{x}_N\}$. In the K-means approach, each data point $\vec{x}_n$ is assigned to the cluster where the cluster's centroid is closest to the data point. This distance is usually measured as a Euclidean distance. An incremental mode K-means approach will re-estimate values of each centroid after each data point assignment. A batch mode K-means approach will re-estimate values of each centroid only once after all data points are assigned. A Centroid value is an average value of all data points in its cluster. The process alternates between cluster assignment and centroid re-estimation, until the termination criteria is met. The termination criteria is that changes in cluster assignment cease. (See Bishop [18] for discussion about K-means, its issues and related methods.)

Once centers are determined, scales can be set to vary based on the reciprocal of maximum distance between centers as shown in Equation 2.35.

$$z_{d,m} = \frac{1}{\max_{m'=1,\ldots,M} |v_{d,m} - v_{d,m'}|} \tag{2.35}$$

Then, weights, which are linear in the output, can be determined by linear least square methods such as a singular value decomposition technique. (See Demmel [38] for more information about linear least square methods.)

The number of clusters, which later will become the number of hidden units in RBF, has to be specified for a K-means algorithm. This number can be determined by Akaike Information Criteria, as discussed by Akaike [1]. Akaike Information Criteria (AIC)[11] indicate a trade-off between goodness-of-fit and complexity of a model. AIC is defined as $\text{AIC} = -2 \cdot \log L + 2 \cdot M_\theta$ where $\log L$ is the maximum log likelihood of the data and M_θ is the number of model parameters. The variable M_θ in this context refers to the K-means parameters. Since the K-means approach has κ centers and each center has D dimensions, $M_\theta = \kappa \cdot D$. The number of centers, M, determined by AIC for the K-means approach is shown in Equation 2.36 [see 82, for detail]. The first term inside

[11] Akaike is one of many approaches in model selection. Bishop [18] commented about AIC and its variations, e.g. BIC, having excessive tendency toward a simple model. (See Burnham and Anderson [21] for model selection and Orr [87] for model selection for RBF)

minimization is the Residual Sum of Squares (RSS) at an optimal clustering of κ clusters. The RSS is a measurement of how well the K-means approach can cluster data into κ groups. The second term in the minimization represents model complexity,

$$M = \arg\min_{\kappa} \{\mathrm{RSS}^*(\kappa) + 2 \cdot \kappa \cdot D\} \tag{2.36}$$

$$\mathrm{RSS}^*(\kappa) = \min_{\pi \in \Pi(\kappa)} \mathrm{RSS}^{\pi}(\kappa) \tag{2.37}$$

$$\mathrm{RSS}^{\pi}(\kappa) = \sum_{m=1}^{\kappa} \mathrm{RSS}_m^{\pi} \tag{2.38}$$

$$\mathrm{RSS}_m^{\pi} = \sum_{\vec{x} \in \Omega_m^{\pi}} \|\vec{x} - \vec{\mu}_m^{\pi}\|^2 \tag{2.39}$$

where M is the number of the clusters, D is a number of dimensions of the data, $\mathrm{RSS}^{\pi}(\kappa)$ is RSS of the κ clusters grouped by the assignment π, $\Pi(\kappa)$ is a set of possible assignments of data for κ clusters (a set of converged assignments from different initialization), $\vec{x}$ is a data point, RBF inputs, $\vec{\mu}_m^{\pi}$ is a centroid of the m^{th} cluster when it is grouped by the assignment π, Ω_m^{π} is a set of data points assigned to the group m and $|\Omega_m^{\pi}|$ is a number of data points in the group m.

RBF with K-means example: This is a simple example to illustrate how RBF can be designed with K-means. (See Chapter 3 for an RBF design investigated in our study.)

Given the one-dimensional input $\vec{x}'$ = [18, 14, 32, 13, 39, 14, 41, 7, 42, 41, 30] and output $\vec{t}'$ = [-256.9, -262.7, -242.8, -264.4, -235.9, -263.6, -233.2, -270.7, -235.6, -235.5, -245.3], RBF can be designed starting by determining RBF centers. Table 2.11 shows the K-means cluster assignment for different κ's. Each assignment is the best result, having the minimum RSS, out of ten replications[12]. When $\kappa = 1$, every data point is assigned to the same cluster. When $\kappa = 9$ (this data set has only 9 distinct values), each data point is assigned to its own cluster. The value of each centroid is shown in Table 2.12. Table 2.13 shows RSS of each assignment and corresponding AIC. The assignments of 6 and 7 clusters have the minimum AIC. Therefore, the number of clusters will be chosen to be 6 according to AIC.

Choosing $M = 6$, RBF centers are $\vec{v}'$ = [31, 13.67, 39, 18, 41.33, 7] and $\mathbf{Z}_m$ = 0.0417, 0.0361, 0.0313, 0.0429, 0.0291, and 0.0291 for $m = 1, ..., 6$ respectively. Then weights can be calculated and $\vec{w}'$ = [-263.3338, 17.8182, -1.5905, 6.8620, 7.3437, 22.7156, -7.0431]. Figure 2.2 shows the training data and approximation result of obtained RBF.

[12] The results are obtained from a modified version of MathWorks' kmeans (revision 1.4.4.8). It is changed to start with κ random samples of unique data points rather than of all data points.

Table 2.11: Cluster assignment

$x_n =$	18	14	32	13	39	14	41	7	42	41	30
π at $\kappa = 1$	1	1	1	1	1	1	1	1	1	1	1
π at $\kappa = 2$	2	2	1	2	1	2	1	2	1	1	1
π at $\kappa = 3$	2	2	1	2	3	2	3	2	3	3	1
π at $\kappa = 4$	2	2	3	2	4	2	4	1	4	4	3
π at $\kappa = 5$	1	1	2	1	5	1	3	4	3	3	2
π at $\kappa = 6$	4	2	1	2	3	2	5	6	5	5	1
π at $\kappa = 7$	4	7	5	7	6	7	2	3	2	2	1
π at $\kappa = 8$	6	1	8	1	7	1	2	3	4	2	5
π at $\kappa = 9$	4	9	2	3	7	9	5	6	8	5	1

Table 2.12: Cluster centroids

	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9
π at $\kappa = 1$	26.45	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
π at $\kappa = 2$	37.50	13.20	N/A	N/A	N/A	N/A	N/A	N/A	N/A
π at $\kappa = 3$	31	13.20	40.75	N/A	N/A	N/A	N/A	N/A	N/A
π at $\kappa = 4$	7	14.75	31	40.75	N/A	N/A	N/A	N/A	N/A
π at $\kappa = 5$	14.75	31	41.33	7	39	N/A	N/A	N/A	N/A
π at $\kappa = 6$	31	13.67	39	18	41.33	7	N/A	N/A	N/A
π at $\kappa = 7$	30	41.33	7	18	32	39	13.67	N/A	N/A
π at $\kappa = 8$	13.67	41	7	42	30	18	39	32	N/A
π at $\kappa = 9$	30	32	13	18	41	7	39	42	14

Table 2.13: Cluster RSS and AIC

κ	RSS_1^*	RSS_2^*	RSS_3^*	RSS_4^*	RSS_5^*	RSS_6^*	RSS_7^*	RSS_8^*	RSS_9^*	$\text{RSS}^*(\kappa)$	$2\kappa D$	AIC
1	1806.7									1806.7	2	1808.7
2	133.5	62.8								196.3	4	200.3
3	2	62.8	4.7							69.5	6	75.5
4	0	14.7	2	4.7						21.4	8	29.4
5	14.7	2	0.7	0	0					17.4	10	27.4
6	2	0.7	0	0	0.7	0				3.4	12	15.4
7	0	0.7	0	0	0	0	0.7			1.4	14	15.4
8	0.7	0	0	0	0	0	0	0		0.7	16	16.7
9	0	0	0	0	0	0	0	0	0	0	18	18

AIC indicates M = 6 as a choice, because two AIC values tie and M = 6 is less complex.

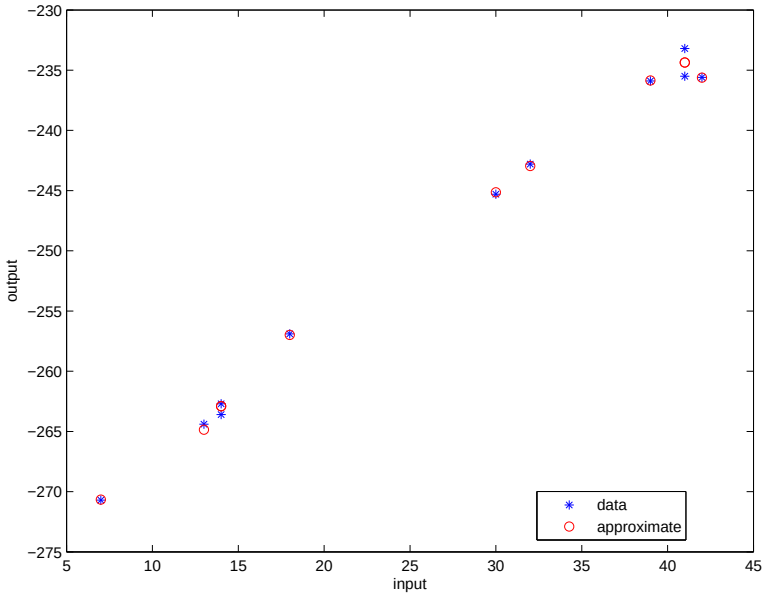

Figure 2.2: One-dimension RBF by using K-means design

Orthogonal Least Squares: In addition to clustering approach, a widely used alternative to determine RBF centers is the Orthogonal Least Squares Learning algorithm (OLS), introduced by Chen et al. [26]. OLS starts from one hidden unit and then keeps adding centers one by one until RBF meets a pre-specific error goal. Each center is chosen from a data point[13] that is projected to reduce the largest error.

Chen et al. [26] recommended that after a few trials, residual variance, $\sigma_\xi^2 \approx \frac{\sum_{n=1}^{N}(y(x_n)-t_n)^2}{N-1}$, can be determined. Then the error goal can be set slightly higher than the ratio of residual variance over target variance, $\frac{\sigma_\xi^2}{\sigma_t^2}$.

In addition to providing a guideline to set error goal, several trials can provide a guideline to set up RBF scales. All scales, $z_{d,m}$'s, can be set to M/Δ^2, where Δ is the maximum distance between any two centers. This makes $\|\vec{x}_n - \vec{v}_m\|_{\mathbf{Z}_m} = b\|\vec{x}_n - \vec{v}_m\|$ where $b = M/\Delta^2$. Increases in $z_{d,m}$ values result in decreased basis spread.[14] (See Chen et al. [26] and Haykin [56] for discussion about OLS.)

RBF with OLS example: To illustrate RBF design by using the OLS method, the one-dimensional input $\vec{x}' =$ [18, 14, 32, 13, 39, 14, 41, 7, 42, 41, 30] is used. Unlike a clustering method, OLS requires training output as well. Suppose a training output data is that $\vec{t}' =$ [-256.9, -262.7, -242.8, -264.4, -235.9, -263.6, -233.2, -270.7, -235.6, -235.5, -245.3].

[13] The explanation of OLS is based on Matlab implementation (newrb.m of revision 1.1.6.2).

[14] Matlab implementation of RBF design (newrb revision 1.1.6.2) takes parameter `sp` as a user specific argument and it defines all scales $z_{d,m}$ to be `b = sqrt(-log(.5))/sp;` where `sp` is called spread.

A trial design with OLS and default parameters is conducted[15]. An error goal is set to 0; a spread is set to unity, $z_m = b = \sqrt{-\log(0.5)} \approx 0.8326$ for all 11 hidden units. Table 2.14 shows centers and weights obtained from the trial design. It should be noted that there are 3 zero weights in this trial design. This makes the effective number of hidden units to 8, instead of the full 11.

With the obtained parameters, the maximum distance between any two centers is 35 and the variance of residual is 0.2542. Therefore, the new spread will be 92.5061 (or $z_m = \frac{11}{35^2} \approx 0.009$, for all m's). Variance of the output (σ_t^2) is 200.1249. Then, a new error goal is set to be slightly over 0.0013. RBF is designed with an error goal of 0.002 and a spread of 92.5061. Table 2.15 shows the final RBF design. Again, there are 5 zero weights, hence only 6 effective hidden units. Figure 2.3 shows the training data and approximation obtained by the OLS design RBF.

Table 2.14: OLS trial design

m	0	1	2	3	4	5	6	7	8	9	10	11
centers v_m	N/A	14	41	7	30	18	32	39	13	42	14	41
spreads z_m	N/A	0.83	0.83	0.83	0.83	0.83	0.83	0.83	0.83	0.83	0.83	0.83
weights w_m	-245.47	0	7.50	-25.23	0	-11.43	2.67	9.09	-13.46	6.10	-10.96	0

Table 2.15: OLS design

m	0	1	2	3	4	5	6	7	8	9	10	11
centers v_m	N/A	7	13	42	41	41	14	14	18	30	39	32
weights w_m	-4.46×10^6	10.92	-38.91	-2.53	0	0	0	0	36.17	0	9.32	-14.96

OLS provides a convenient RBF design, because it determines a number of centers and center values simultaneously. K-means requires a number of centers to be specified. Although, a number of centers can be determined by AIC, multiple trials of different numbers are required to perform AIC.

Incremental RBF: Similar to FFNet, RBF can be trained in either a batch mode, mentioned above, or an incremental mode. For the incremental update, derivatives of error with respect to RBF parameters can be derived in a similar manner to FFNet derivations. The derivative of the Gaussian basis function with respect to its argument is $\phi'(a) = -2 \cdot a \cdot \phi(a)$ and partial derivatives

[15] The set up and result in this example are based on newrb (revision 1.1.6.2), which is MathWorks' implementation of RBF design.

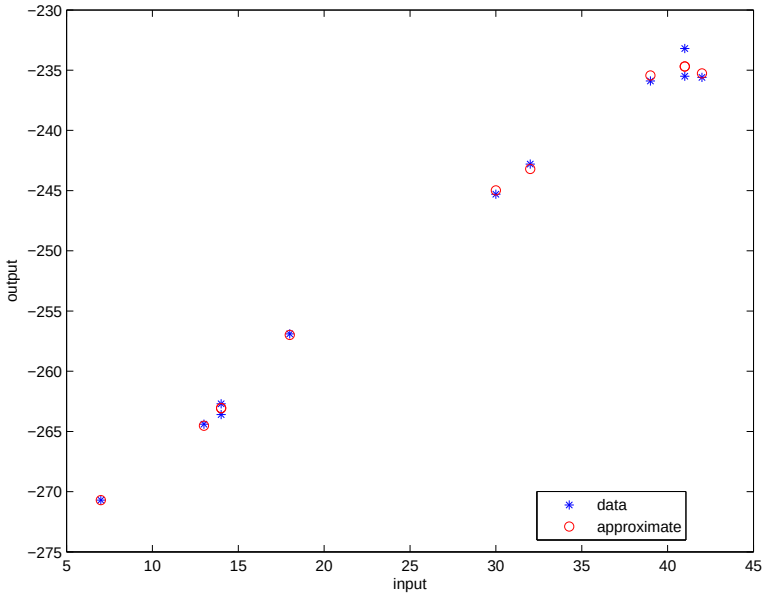

Figure 2.3: One-dimension RBF by using OLS design

of error with respect to w's, z's and v's are shown in Equations 2.40, 2.41, and 2.42 respectively.

$$\begin{aligned}
\frac{\partial \xi_n}{\partial w_{m,k}} &= (y_k(\vec{x}_n) - t_{k,n}) \cdot \frac{\partial y_k(\vec{x}_n)}{\partial w_{m,k}} \\
&= (y_k(\vec{x}_n) - t_{k,n}) \cdot \begin{cases} 1 & \text{for } m = 0 \\ \phi(\|\vec{x}_n - \vec{v}_m\|_{\mathbf{Z}_m}) & \text{for } m = 1, \ldots, M \end{cases} \qquad (2.40) \\
\frac{\partial \xi_n}{\partial z_{d,m}} &= \sum_{k=1}^{K} (y_k(\vec{x}_n) - t_{k,n}) \cdot w_{m,k} \cdot \phi'(\|\vec{x}_n - \vec{v}_m\|_{\mathbf{Z}_m}) \cdot \frac{(x_{n,d} - v_{d,m})^2}{2 \cdot \|\vec{x}_n - \vec{v}_m\|_{\mathbf{Z}_m}} \\
&= -\sum_{k=1}^{K} (y_k(\vec{x}_n) - t_{k,n}) \cdot w_{m,k} \cdot \phi(\|\vec{x}_n - \vec{v}_m\|_{\mathbf{Z}_m}) \cdot (x_{n,d} - v_{d,m})^2 \qquad (2.41) \\
\frac{\partial \xi_n}{\partial v_{d,m}} &= -\sum_{k=1}^{K} (y_k(\vec{x}_n) - t_{k,n}) \cdot w_{m,k} \cdot \phi'(\|\vec{x}_n - \vec{v}_m\|_{\mathbf{Z}_m}) \cdot z_{d,m} \cdot \frac{(x_{n,d} - v_{d,m})}{\|\vec{x}_n - \vec{v}_m\|_{\mathbf{Z}_m}} \\
&= 2 \cdot \sum_{k=1}^{K} (y_k(\vec{x}_n) - t_{k,n}) \cdot w_{m,k} \cdot \phi(\|\vec{x}_n - \vec{v}_m\|_{\mathbf{Z}_m}) \cdot z_{d,m} \cdot (x_{n,d} - v_{d,m}) \qquad (2.42)
\end{aligned}$$

The incremental GD updates of RBF are shown in Equations 2.43, 2.44 and 2.45.

$$\begin{aligned}
z_{d,m} &\leftarrow z_{d,m} + \rho_z \sum_{k=1}^{K} (y_k(\vec{x}_n) - t_{k,n}) \cdot w_{m,k} \cdot \phi_{n,m} \cdot (x_{n,d} - v_{d,m})^2 \qquad (2.43) \\
v_{j,m} &\leftarrow v_{j,m} - \rho_v \cdot 2 \cdot \sum_{k=1}^{K} (y_k(\vec{x}_n) - t_{k,n}) \cdot w_{m,k} \cdot \phi_{n,m} \cdot z_{d,m} \cdot (x_{n,d} - v_{d,m}) \qquad (2.44) \\
w_{0,k} &\leftarrow w_{0,k} - \rho_w (y_k(\vec{x}_n) - t_{k,n}) \\
w_{m,k} &\leftarrow w_{m,k} - \rho_w (y_k(\vec{x}_n) - t_{k,n}) \cdot \phi_{n,m} \qquad \text{for } m = 1, \ldots, M \qquad (2.45)
\end{aligned}$$

where $\phi_{n,m} = \phi(\|\vec{x}_n - \vec{v}_m\|_{\mathbf{Z}_m})$

It should be noted that, after the RBF structure, i.e., centers and scales, are determined, a conventional incremental RBF only updates the weights, which have a linear effect on the output and leaves nonlinear parameters, centers and scales, unchanged. This is done to reduce the extent of nonlinearities, which often relate to poor GD performance. This is discussed by Chen et al. [26]. However, Haykin [56] discussed the benefit of updating RBF centers and scales and used experimental results from Wetterschereck and Dietterich [127] to support his argument. In addition to either no update at all or every step update, Haykin [56] recommended using a less frequent time scale for updating RBF structure, while weights can be updated more frequently.

2.4.3 Updating scheme

An updating scheme is a procedure to improve the quality of approximation according to newly obtained information by tuning parameters, e.g. RBF's weights. The temporal-Difference method (TD), discussed by Sutton [113], is used as an updating scheme in our study. TD is designed for Markov decision processes. A dynamic program of a general Markov decision process, is shown in Equation 2.46,

$$J(s_t) = \min_u E[r(s, a, s') + \alpha \cdot J(s')|s = s_t, a = u] \tag{2.46}$$

where s_t is a current state; u is a current action; s' is a next state (which is a consequence of taking action u at state s_t); J(s) is a state cost function when starting with state s; $r(s, a, s')$ is a cost when the current state is s, the control is a and the next state is s'; α is a discount factor and $E[\cdot|\cdot]$ is a conditional expectation operator.

Rather than expectation, TD uses sampled information with current approximation to calculate a prediction difference and then uses this difference to update approximation parameters. The prediction difference, called the temporal difference, is the difference between an approximate state cost with recently obtained information and an approximate state cost for the same state, but without that information. Equation 2.47 shows formulation of a temporal difference. It is a difference of both approximations when their parameters are determined at time t.

$$\psi_t = r_{t+1} + \alpha \cdot \hat{J}_t(s_{t+1}) - \hat{J}_t(s_t) \tag{2.47}$$

where s_t, a_t and s_{t+1} are a current state, a current action and a next state, respectively; r_{t+1} is a period cost and $\hat{J}_t(s)$ is a current approximate state cost. It should be noted that both s_{t+1} and r_{t+1} are sampled at time $t + 1$.

The first two terms of Equation 2.47, or $r_{t+1} + \alpha \cdot \hat{J}_t(s_{t+1})$, represent an approximate state cost with newly obtained information, a single-period cost r_{t+1} and a next state s_{t+1}. This approximate

state cost is calculated from a combination of a single-period cost and a discounted approximate cost-to-go.

The last term of Equation 2.47, or $\hat{J}_t(s_t)$, represents approximate state cost without new information (r_{t+1}). Without this information, the approximate state cost is approximated directly from a function approximation of current state s_t.

An approximation parameter update can be formulated[16] by TD as shown in Equation 2.48.

$$\theta_m^{(t+1)} \quad = \theta_m^{(t)} + \beta \cdot \psi_t \cdot \sum_{k=0}^{t} (\alpha\lambda)^{t-k} \cdot \nabla_{\theta_m^{(t)}} \hat{J}_t(s_k) \quad ,\forall m \tag{2.48}$$

where $\theta_m^{(t)}$ is the m^{th} approximation parameter at time t, β is an update step size, ψ_t is a temporal difference, as defined in Equation 2.47, α is a discount factor, λ is an eligibility factor and $\nabla_{\theta_m^{(t)}} \hat{J}_t(s_k)$ is a gradient of approximate state cost of s_k with respect to parameter $\theta_m^{(t)}$.

All parameters will be updated to have a new approximate state cost corresponding to newly obtained information.

The term $\psi_t \cdot \sum_{k=0}^{t} (\alpha\lambda)^{t-k} \cdot \nabla_{\theta_m^{(t)}} \hat{J}_t(s_k)$ of Equation 2.48 is an update step of a gradient-descent method, to minimize ψ_t^2. It should be noted that a temporal difference, ψ_t, can be viewed as an error between a sampled single-period cost r_{t+1} and its current approximation, $\hat{r}_{t+1} = \hat{J}_t(s_t) - \alpha \cdot \hat{J}_t(s_{t+1})$. The term $\sum_{k=0}^{t} (\alpha\lambda)^{t-k} \cdot \nabla_{\theta_m^{(t)}} \hat{J}_t(s_k)$ can be viewed as a weighted gradient of the current approximation with respect to the parameter. The weighted gradient is used to account for the fact that a single-period cost r_{t+1} is a consequence of a series of states and so is the temporal difference ψ_t. This weighted gradient is a mechanism to backward credit states in a sequence leading to the current consequence. This backward crediting mechanism is called an Eligibility Trace. The Eligibility Trace gives a recent state higher weight than an earlier state.

Equation 2.48 shows an update of TD with an arbitrary approximation function. Due to the ease of integration of TD, Sutton and Barto [114] recommended using a function belonging to a linear family. (See Sutton and Barto [114] for a discussion on choices of an approximation function for TD, Bertsekas and Tsitsiklis [15] for issues of convergence and Barreto and Anderson [9] for a proposed remedy for nonlinear RBF)

A linear family approximation function having M bases can be written as $\hat{J}_t(s) = \sum_{m=0}^{M} w_m^{(t)} \cdot \phi_m(s)$, where $\phi_m(s)$ is the m^{th} basis given state s and $w_m^{(t)}$ is the m^{th} parameter at time t. Its derivative is $\partial \hat{J}_t(s) / \partial w_m^{(t)} = \phi_m(s)$. Therefore, an update of TD with a linear family approximation function can be simplified to Equation 2.49.

$$w_m^{(t+1)} \quad = w_m^{(t)} + \beta \cdot \psi_t \cdot \sum_{k=0}^{t} (\alpha\lambda)^{t-k} \cdot \phi_m(s_k) \quad ,\forall m \tag{2.49}$$

[16] Updating formula, presented here, is based on TD with function approximation in Bertsekas and Tsitsiklis [15].

Equation 2.49 can be broken down into Equations 2.50 and 2.51.

$$w_m^{(t+1)} = w_m^{(t)} + \beta \cdot \psi_t \cdot \nu_m^{(t)} \tag{2.50}$$

$$\nu_m^{(t)} = \phi_m(s_t) + (\alpha\lambda) \cdot \nu_m^{(t-1)} \tag{2.51}$$

where $\nu_m^{(0)} = 0$ for all m's.

TD with a linear approximation function can be implemented with all eligibilities, ν's, initialized to 0's. At each time step, current eligibility can be computed from Equation 2.51 and then linear parameters can be updated by Equation 2.50. This technique requires memory to store M entries of previous eligibilities and computation for M eligibilities for each update. In practice, most of these entries will have values close to 0. Cichosz [32] discussed Eligibility Trace efficiency and Eligibility Trace implementation to improve efficiency.

Since RBF can be operated effectively in either linear or non-linear mode[17], our study uses RBF as a choice of an approximation function. When RBF is thus used, the basis $\phi_0(s) = 1$ and $\phi_m(s) = \exp(-\|s - v_m\|^2_{\mathbf{Z}_m})$ for $m = 1, \ldots, M$. Here v_m and $\mathbf{Z}_m$ can be obtained by methods mentioned in Section 2.4.2. (See Kaelbling et al. [65], Sutton and Barto [114] and Bertsekas and Tsitsiklis [15] for review of Temporal Difference learning, ADP and related issues.)

Learning rate Powell [97] identified choosing a learning rate, β in Equation 2.50 , as a major decision issue in ADP. A constant step size is the simplest and most widely used learning rate strategy. Previous ADP studies, including Van Roy et al. [118], Godfrey and Powell [48], Pontrandolfo et al. [93], Giannoccaro and Pontrandolfo [47], Shervais et al. [108], Kim et al. [70], Choi et al. [30], Topaloglu and Kunnumkal [117], [62], Chaharsooghi et al. [25], Kim et al. [71], Kwon et al. [75], and Jiang and Sheng [64], used a constant step size. Kim et al. [71] recommended a step size of 0.1 for a nonstationary environment. More advanced step size selection strategies are the subject of ongoing active research. Bertsekas and Tsitsiklis [15], Powell [96] and Powell [97] discussed utilization of a step size that can be a function of time or observation. Powell [96] classified various advanced step size selection strategies into deterministic and stochastic families.

McClain's formula, shown in Equation 2.52, is a deterministic step size strategy.

$$\beta_n = \frac{\beta_{n-1}}{1 + \beta_{n-1} - \bar{\beta}} \tag{2.52}$$

where β_n is a step size at iteration n.

Here the value of step size declines as the learning process converges to a user specific parameter $\bar{\beta}$. Figure 2.4 shows McClain step size with $\beta_0 = 1$ and $\bar{\beta} = 0.1$ comparing to a $1/n$ step size strategy.

[17] RBF is operated in linear mode when ρ_z and ρ_v (in Equations 2.43 and 2.44) are set to 0's.

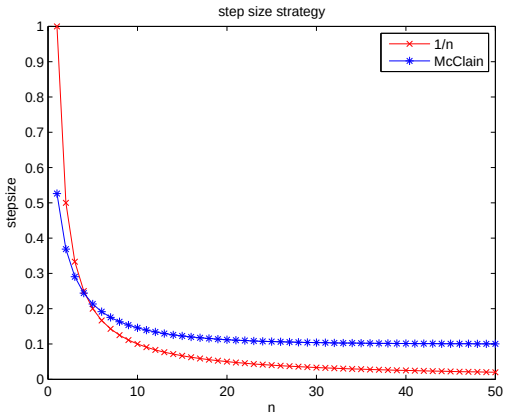

Figure 2.4: McClain step size

The bias-adjusted Kalman filter (BAKF), as discussed by George and Powell [45], is a stochastic step size strategy. It adjusts a step size according to its observation to minimize both bias and variance approximation errors. Table 2.16 shows the BAKF procedure. Powell [96] recommended setting $\bar{\eta} \in (0.05, 0.10)$ for most applications.

Table 2.16: Bias-Adapted Kalman Filter step size rule

Step 0:	Initialization
	a) initialize step sizes, $\beta_0 = \eta_0 = 1$.
	Note: setting $\beta_0 = \eta_0 = 1$ makes $\bar{\psi}_0$, $\bar{b}_0$, $\bar{\nu}_0$, and $\bar{\lambda}_0$ effectless.
	b) set $n = 1$.
Step 1:	Observe a target, ψ_n.
Step 2:	Smooth the baseline estimate.
	$\bar{\psi}_n = (1 - \beta_{n-1})\bar{\psi}_{n-1} + \beta_{n-1}\psi_n$
Step 3:	Approximate parameters.
	a) $\epsilon_n = \bar{\psi}_{n-1} - \psi_n$
	b) $\bar{b}_n = (1 - \eta_{n-1})\bar{b}_{n-1} + \eta_{n-1}\epsilon_n$
	c) $\bar{\nu}_n = (1 - \eta_{n-1})\bar{\nu}_{n-1} + \eta_{n-1}\epsilon_n^2$
	d) $\bar{\sigma}_n^2 = \frac{\bar{\nu}_n - \bar{b}_n^2}{1+\bar{\lambda}_{n-1}}$
Step 4:	Calculate new step sizes.
	a) $\beta_n = \begin{cases} 1/(n+1), & \text{for } n = 1, 2 \\ 1 - \frac{\bar{\sigma}_n^2}{\bar{\nu}_n}, & \text{for } n = 3, 4, 5, ... \end{cases}$
	b) $\eta_n = \frac{\eta_{n-1}}{1+\eta_{n-1}-\bar{\eta}}$
Step 5:	Compute a smoothing coefficient.
	$\bar{\lambda}_n = (1 - \beta_{n-1})^2\bar{\lambda}_{n-1} + \beta_{n-1}^2$
Step 6:	Repeat from step 1 until a termination condition is met.
	$n \leftarrow n + 1$

An ϵ_n is an approximation error at iteration n.
A $\bar{\sigma}_n^2$ is an estimate of a variance, of an approximation error, after iteration n.
A $\bar{b}_n$ is an estimate of a bias, of an approximation error, after iteration n.
A $\bar{\nu}_n$ is an estimate of a variance of a bias after iteration n.
This table is based on Powell [96, Figure 6.8].

Figure 2.5[18] shows a step size obtained from BAKF of different prediction error patterns. When a sequence of errors has alternating signs, BAKF reduces step sizes substantially. Although BAKF seems to be very appealing, Powell [96] commented that BAKF performance deteriorates for systems with high variance. In addition, use of step size functions is still undergoing active research for learning-based ADPs. Our study uses a constant step size scheme for setting ADP learning rate. See Powell [96] and George and Powell [45] for additional discussion of advanced step size strategies.

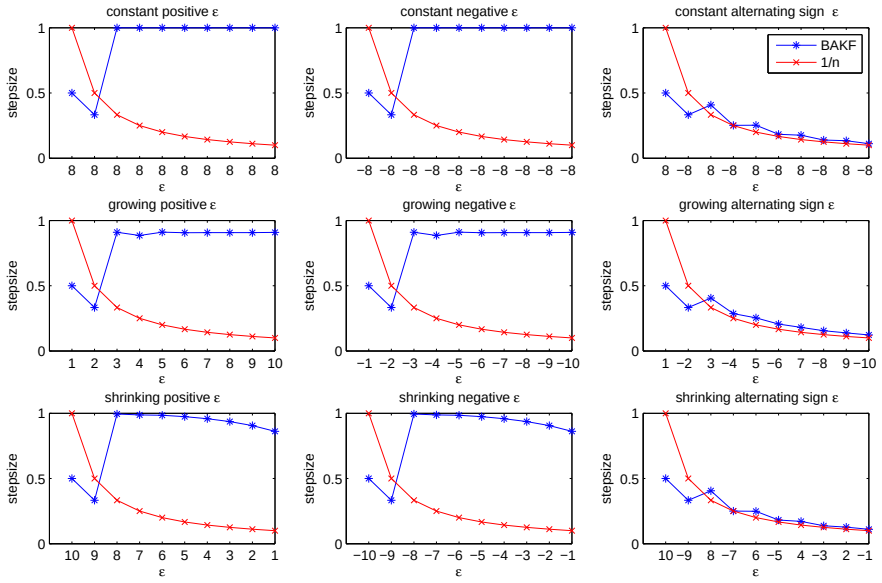

Figure 2.5: BAKF step size

2.4.4 Simulation-based ADP

When a model of the problem is available, simulation can be used to provide information that helps to determine a proper action. Our study investigates two methods of simulation-based ADPs: Roll-out and Hindsight Optimization.

Roll-out

Roll-out, discussed by Bertsekas and Tsitsiklis [15], decides an action based on approximate costs that are obtained with simulation. An action is selected that minimizes an approximate immediate state-action cost and an approximate cost-to-go. A cost-to-go value is obtained by simulating the process with actions determined by a base policy π. Equation 2.53 is used to show how Roll-out

[18] The plot is made with $\alpha_0 = 1$, $\eta_0 = 1$ and $\bar{\eta} = 0.09$ (This setting render β_0, ν_0 and λ_0 effectless).

determines an action

$$a_t = \arg\min_a \sum_{n=1}^{N} \left\{ \hat{c}(s_t, a) + \sum_{i=1}^{T} \alpha^i \hat{c}^{\pi}(\hat{s}_{t+i}) \right\} \tag{2.53}$$

where a_t is the action chosen, N is the number of simulations, T is the simulation horizon, $\hat{c}(s, a)$ is the simulated immediate cost when action a is taken, $\hat{c}^{\pi}(s)$ is the simulated period cost when an action is determined by base policy π and $\hat{s}_{t+i}$ is the simulated state of time at $t + i$.

Roll-out first simulates to obtain an immediate cost $\hat{c}(s_t, a)$ and next state $\hat{s}_{t+1}$. Then the period cost $\hat{c}^{\pi}(\hat{s}_{t+1})$ is obtained by the simulation. The process keeps going until the end of simulation horizon, denoted T. The second term in Equation 2.53 represents an approximate discounted cost-to-go, given policy π. The combination of immediate cost and discounted cost-to-go is an approximated cost of each action in a search space. To take into account process variation, multiple simulated costs are generated and the decision based on their summation.

Hindsight Optimization

Similar to Roll-out, Hindsight Optimization (HO), introduced by Chong et al. [31], uses simulation to estimate an immediate cost and a cost-to-go. An approximate immediate cost is obtained directly with simulation. However, to estimate a cost-to-go a sequence of actions is required. Roll-out uses a base policy to determine this sequence of actions. Unlike Roll-out, HO uses a special simulation that does not require a base policy. HO first simulates a sequence of random variables that are realizations of inventory demand. Then, given the known sequence of random variable values, it chooses a sequence of actions to minimize the cost. Therefore, the cost obtained in HO is the minimal cost achieved for that particular instance. Equation 2.54 summarizes this procedure,

$$a_t = \arg\min_a \sum_{n=1}^{N} \left\{ \hat{c}(s_t, a) + \min_{\vec{u}} \left(\sum_{i=1}^{T} \alpha^i \hat{c}_{\xi_{n,i}}(\hat{s}_{t+i}(\xi_{n,i}), u^{(i)}) \right) \right\} \tag{2.54}$$

where $\xi_{n,i}$ is a simulated uncertainty of the n^{th} simulation at period $t+i$ and $\hat{c}_{\xi_{n,i}}(s, a)$ and $\hat{s}_{t+i}(\xi_{n,i})$ are respectively simulated transition cost and state for the given value of $\xi_{n,i}$, with other variables as mentioned earlier.

It should be noted that HO is different from Rollout with a Look-Ahead method as a base policy. Rollout with a Look-Ahead base policy simulates events with actions that are determined based on an average projection. A cost-to-go approximation in Rollout depends on the deviation of each simulation from an average case. HO simulates events with actions that are determined based on known realized values of random variables. A cost-to-go approximation in HO depends on simulation-generated random variable values. To illustrate the difference, consider an example

of a positively skewed demand distribution whose mean is much greater than its median. Most values of demand generated will be small and close to the median, with few very high values. HO will approximate a low cost because simulated actions are determined with known demand, while Rollout will approximate a very high cost because simulated actions are determined based on an average demand. This is clearly much different than having mostly low values and few very high values.

CHAPTER 3

A RADIAL BASIS FUNCTION AS A COST-TO-GO APPROXIMATOR

Learning-based ADP has recently received increased attention for inventory management research. Its approximate cost-to-go is often implemented with a look-up table or aggregation, such as works of Paternina-Arboleda and Das [90], Giannoccaro and Pontrandolfo [47], Kim et al. [70], Chinthalapati et al. [29], Kwon et al. [75], Kim et al. [71], Chaharsooghi et al. [25] and Jiang and Sheng [64]. Other alternatives include feedforward neural network used by Van Roy et al. [118], Das et al. [36] and Shervais et al. [108] and a linear combination of features used by Van Roy et al. [118]. A look-up table becomes computationally inefficient for a large state-action space. The feedforward neural network is highly nonlinear and requires skill and experience to determine its parameters properly for learning-based ADPs. Bertsekas and Tsitsiklis [15] and Sutton and Barto [114] suggest a linear family approximation function to use with ADP. One of these functions, the Radial Basis Function (RBF), has been proved to be a universal approximation function, being able to approximate an arbitrary function when it is properly set up. See Powell [95] and Park and Sandberg [89]. RBF also can be operated in a linear mode. To operate RBF in linear mode, its structure, i.e., centers and scales, has to be pre-assigned. The literature search revealed no previous studies that investigate the application of ADP with RBF to inventory management systems. This chapter addresses strategies to design RBF centers and scales for inventory management systems and provides simulation-based experiments to evaluate performance.

3.1 Inventory problem with AR1 demand

Figure 3.1 shows a diagram of a single-echelon inventory system. This inventory system includes both in-transit and on-hand inventories. A flow of items starts from a supplier's delivery after receiving a replenishment order. While items are being delivered, they become in-transit inventory. After a given leadtime, items arrive in at a storage point and are classified as on-hand inventory. These items are ready to be delivered to customers once there is a demand.

Problem: An inventory problem can be interpreted as a Markov decision process, where previous demand, on-hand inventory, and in-transit inventory are defined as Markov states. The replenishment order is Markov action. State transition is influenced by the current state and current action A transition cost is a combination of costs for replenishment and handling.

State: On-hand inventory is available to serve the demand. Its level, $\mathring{x}$, is a non-negative integer. In-transit inventory describes items that have been ordered, but have not been delivered. In-transit inventory status, B, is a set of non-negative integers, $\mathrm{B} \subset \{0\} \cup \mathcal{I}^+$, where $\mathcal{I}^+$ is a set of positive integers. A backlog order is a virtual inventory. A backlog order, $\mathring{x}'$, occurs when the demand

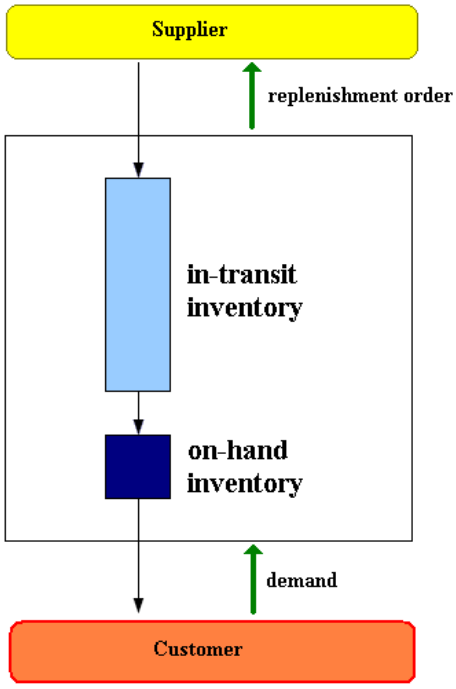

Figure 3.1: Single-echelon inventory problem

exceeds on-hand inventory. Once the items are available, after a replenishment has arrived, the excess demand will be served. A backlog order is a negative inventory and it is represented by a negative integer. It should be noted that an on-hand inventory level, $\mathring{x} \in \{0, \mathcal{I}^+\}$, and a backlog order, $\mathring{x}' \in \mathcal{I}^-$, are mutually exclusive. For convenience, on-site inventory, x, is used to refer to both on-hand inventory and backlog order quantities, such that non-negative x represents on-hand inventory and negative x represents backlog order. Zhang [130] comments that the first order Autoregressive model, or AR(1) is one of the most commonly adopted demand models. This study investigates the case where demand is modeled by AR(1). Inventory x, in-transit inventory B, and demand information are state variables of a Markov decision process for this inventory system.

Action: A replenishment order, denoted u, is a control to regulate this inventory system. The amount of the replenishment order will be part of in-transit inventory, along with previous orders that have not yet arrived. After a leadtime, that part of in-transit inventory will arrive at the destination and the replenishment will be added to the on-site inventory. The replenishment order is a non-negative integer. Hence, an action of this inventory system is a non-negative integer.

State transition: A next-stage inventory x_{t+1} equals a current inventory x_t added to an in-transit inventory that has currently arrived, reducing demand in a current period. Equation 3.1 shows this

transition.

$$x_{t+1} = x_t + B_1^{(t)} - D_t \tag{3.1}$$

where x_{t+1} is the inventory at time $t+1$ (beginning of period $t+1$), x_t is an inventory at time t, $B_1^{(t)}$ is the part of the in-transit inventory that has arrived in period t, and D_t is a cumulative demand during period t.

In-transit inventory at the next stage is its value at the current stage added to a new order and less parts that have arrived. Suppliers' delivery systems are assumed to be dependable and their leadtimes are assumed to be constant at L period(s). For an inventory system with zero leadtime ($L = 0$), let a replenishment order be u_t, and then the next period inventory is $x_{t+1} = x_t + u_t - D_t$. For an inventory system with non-zero constant leadtime ($L > 0$), in-transit inventory at time t can be presented by a vector $\vec{B}_t : \{0, \mathcal{I}^+\}^L$. In-transit inventory entries are shown in Equation 3.2.

$$\begin{aligned} \vec{B}^{(t+1)} &= [\ B_1^{(t+1)},\ B_2^{(t+1)},\ B_3^{(t+1)},\ \ldots,\ B_{L-1}^{(t+1)},\ B_L^{(t+1)}\ ]^T \\ &= [\ B_2^{(t)},\ B_3^{(t)},\ B_4^{(t)},\ \ldots,\ B_L^{(t)},\ \mathring{u}_t\ ]^T \end{aligned} \tag{3.2}$$

For example, when $L = 1$, $\vec{B}^{(t+1)} = [u_t]$. When $L = 2$, $\vec{B}^{(t+1)} = [B_2^{(t)},\ u_t]$. When $L = 3$, $\vec{B}^{(t+1)} = [B_2^{(t)},\ B_3^{(t)},\ u_t]$ and so on.

The demand transition is modeled as AR(1): $D_t - \mu = a_1 \cdot (D_{t-1} - \mu) + e_t$ where μ is a mean of an AR(1) process, a_1 is an AR(1) parameter, e_t is a white noise with zero mean and σ^2 variance, and D_{t-1} is a previous demand. This can be simplified, by $a_0 = \mu - a_1 \cdot \mu$, as shown in Equation 3.3.

$$D_t = a_0 + a_1 \cdot D_{t-1} + e_t \tag{3.3}$$

The state transition of the AR(1) inventory system can be summarized as shown in Equation 3.5 where $\vec{s}$ represents the state.

$$\begin{bmatrix} D_t \\ x_{t+1} \\ B_1^{(t+1)} \\ B_2^{(t+1)} \\ \vdots \\ B_{L-1}^{(t+1)} \\ B_L^{(t+1)} \end{bmatrix} = \begin{bmatrix} a_1 & 0 & 0 & 0 & 0 & \ldots & 0 \\ -a_1 & 1 & 1 & 0 & 0 & \ldots & 0 \\ 0 & 0 & 0 & 1 & 0 & \ldots & 0 \\ 0 & 0 & 0 & 0 & 1 & \ldots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & 0 & \ldots & 1 \\ 0 & 0 & 0 & 0 & 0 & \ldots & 0 \end{bmatrix} \cdot \begin{bmatrix} D_{t-1} \\ x_t \\ B_1^{(t)} \\ B_2^{(t)} \\ \vdots \\ B_{L-1}^{(t)} \\ B_L^{(t)} \end{bmatrix} + \begin{bmatrix} a_0 + e_t \\ -a_0 - e_t \\ 0 \\ 0 \\ \vdots \\ 0 \\ u_t \end{bmatrix} \tag{3.4}$$

$$\text{or} \qquad \vec{s}_{t+1} = \Lambda \cdot \vec{s}_t + \vec{\lambda}_{u_t} \tag{3.5}$$

where $\vec{s}_{t+1}$, $\vec{s}_t$ and $\vec{\lambda}_{u_t}$ are $(2+L)$ column vectors represented in Equation 3.4 as a vector on the left, the first vector on the right and the last vector on the right respectively; Λ is a $(2+L) \times (2+L)$ matrix and represented in Equation 3.4 as a matrix on a right hand side.

For the case of $L = 0$, Equation 3.5 is reduced to:

$$\begin{bmatrix} D_t \\ x_{t+1} \end{bmatrix} = \begin{bmatrix} a_1 & 0 \\ -a_1 & 1 \end{bmatrix} \cdot \begin{bmatrix} D_{t-1} \\ x_t \end{bmatrix} + \begin{bmatrix} a_0 + e_t \\ -a_0 - e_t + \mathring{u}_t \end{bmatrix} \tag{3.6}$$

For the case of $L = 1$, Equation 3.5 is reduced to:

$$\begin{bmatrix} D_t \\ x_{t+1} \\ B_1^{(t+1)} \end{bmatrix} = \begin{bmatrix} a_1 & 0 & 0 \\ -a_1 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} D_{t-1} \\ x_t \\ B_1^{(t)} \end{bmatrix} + \begin{bmatrix} a_0 + e_t \\ -a_0 - e_t \\ \mathring{u}_t \end{bmatrix} \tag{3.7}$$

Transition cost: A transition cost is associated with the system changing from a current state to a next state. A transition cost for this inventory system consists of a replenishment cost and an inventory handling cost. The transition cost for this problem is

$$r_{t+1}(s_t, u_t, s_{t+1}) = K_t \cdot \delta(u_t) + c_t \cdot u_t + h_t \cdot (s_2^{(t+1)})^+ + b_t \cdot (-s_2^{(t+1)})^+ \tag{3.8}$$

where K_t is a set-up cost, c_t is a unit replenishment cost, h_t is a unit holding cost, b_t is a unit backlogging cost, r_{t+1} is a transition cost whose value will be known at the end of period t, u_t is a replenishment order, $s_2^{(t+1)}$ is the second component of $\vec{s}_{t+1}$, $\delta(\cdot)$ is a step function and $(\cdot)^+$ is a positive function, i.e., $(a)^+ = a \cdot \delta(a)$.

The replenishment cost is a combination of a setup cost and a per unit cost. A setup cost does not depend on how many items are ordered. It is a fixed cost that is posted for every transaction. The per unit cost is a variable cost. It is charged for each nit. A setup cost and a unit cost are the first and second terms respectively of Equation 3.8.

An inventory handling cost is a cost for managing the inventory, it consists of holding costs and backlogging costs. A holding cost is per period cost, charged for each item stored in inventory. A backlogging cost is a cost charged per period when demand exceeds inventory. This cost is a penalty when inventory is in shortage. It can be conceived as an expense associated with not losing the customer. For simplicity in the holding and backlogging costs calculation is assumed to fall at the beginning of the period. Holding cost and backlogging cost are the third and forth terms respectively in Equation 3.8.

3.2 Preliminary-Experiments

A preliminary investigation assesses potential RBF center setups with a two-phase strategy and OLS (Section 2.4.2). This approach requires data to design the RBF. Two data sets are investigated. Both are obtained from simulations, but each has different data collection methods. The first data set is collected at every state up to the simulation horizon. This data set will provide the frequency of states visited. The second data set is collected at only initialized states that are pre-specified.

This data set may not provide information about frequency of states, but it provides better coverage of the state space.

Multiple simulations are run. Each simulation has settings: K_t =\$15; c_t = \$10/unit; h_t = \$1/unit; and b_t = \$40/unit. This discounted problem has a discounting coefficient of $\alpha = 0.95$ and a zero leadtime ($L = 0$). The demand is modeled as AR1, where a_0 = -5, $a_1 = 1$, and demand noise is normally distributed with a variance of 10. The simulated process is controlled with a 3-period expected Look-Ahead method that selects an action based on the sum of 3 expected single-period costs ahead. Equation 3.9 is the T-period expected look-ahead policy.

$$a_t = \text{first}\left(\arg\min_{\vec{u}} \sum_{i=1}^{T} E[c(\bar{s}_{t+i-1})|u_{t+i-1}]\right) \tag{3.9}$$

where $\vec{u}$ is a series of actions ($\vec{u} = [u_t\ u_{t+1}\ ...\ u_{t+T-1}]$ and first($\vec{u}$) = u_t), $E[c(\bar{s}_{t+i-1})|u_{t+i-1}]$ is an expected period cost of the state $\bar{s}_{t+i-1}$ for the given action u_{t+i-1}, and $\bar{s}_{t+i-1} = [\bar{D}_{t+i-2}\ \bar{x}_{t+i-1}]'$ ($\bar{D}_{t-1} = D_{t-1}$. $\bar{x}_t = x_t$. $\bar{D}_{t+i-2} = a_1\bar{D}_{t+i-3} + a_0$ and $\bar{x}_{t+i-1} = \bar{x}_{t+i-2} + u_{t+i-2} - \bar{D}_{t+i-2}$ for i = 2, 3, 4, ...).

The first data set: The first dataset, a trajectory dataset, is simulated for 10 replications with 65 periods each. Since all states visited are recorded for this dataset, it provides 650 samples. These samples are randomly separated into a training set of 500 samples and a validating set of 150 samples. The training set is used to design the RBF, while the validating set is used to validate RBF approximation.

The second data set: The second dataset, a mesh dataset, records only initial states and their consequences. To provide two separate datasets for training and validation, each dataset is generated separately. A mesh dataset for training is generated for previous demands of 0, 50, 100, ..., 1000 and inventory levels of -500, -450, -400, ..., 0, 50, 100, ..., 1000. This makes a total of 651 samples. A mesh dataset for validating is generated for previous demands of 0, 60, 120, ..., 900 and inventory levels of -90, 0, 90, 180, 270, ..., 450, for a total of 112 samples.

One RBF is trained with the trajectory dataset and another RBF is trained with the mesh dataset. Both RBFs are then validated with both the trajectory and mesh validating datasets.

Preliminary-Experiment results: Table 3.1 shows the validation mean squared error (MSE) and mean relative absolute error (MRAE) of both RBFs on both validation data sets. The MSE is obtained from $\frac{1}{N}\sum_{n=1}^{N}(\hat{C}(s_n) - C_n)^2$ and MRAE is obtained from $\frac{1}{N}\sum_{n=1}^{N}|(\hat{C}(s_n) - C_n)/C_n|$

Table 3.1: Pre-stage Experimental results

	the first validating set		the second validating set	
	MSE	MRAE	MSE	MRAE
The first training set	9.6822×10^5	8.98%	7.3363×10^{13}	343.17%
The second training set	1.8759×10^6	13.46%	7.5658×10^4	16.94%

where C_n is the n^{th} data point obtained from simulation, $\hat{C}(s_n)$ is a RBF output corresponding to the n^{th} data point; and N is the number of data points.

Figure 3.2 shows RBF centers designed with the first training data set.

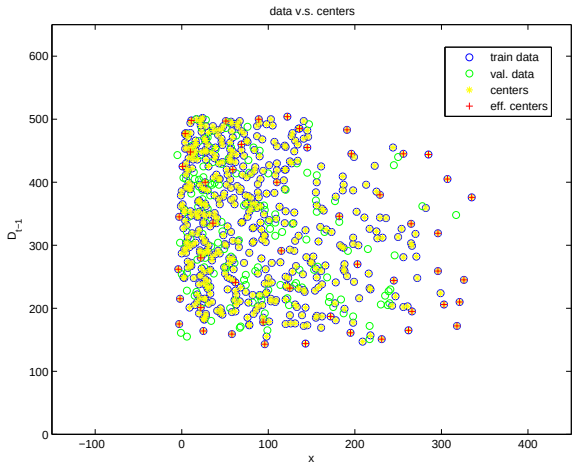

Figure 3.2: The first set data points and RBF centers

Blue bubbles represent coordinates composed of inventory x and previous demand D_{t-1}, of the training data. Green bubbles represent coordinates of the validating data. Yellow asterisks are coordinates of the RBF centers designed by OLS. Although the design takes most of the data points as RBF centers, only a few centers have non-zero weights, as shown by red plus-signs in Figure 3.2.

Figure 3.3 shows a 3D surface plot of RBF output when it is trained with the first training data set. The top left and right plots show training data points in 3D, projected on each plane. These two plots show the range of the state space that this data covers. The bottom left and right plots show 3D and contour plots of RBF output.

Similarly, Figure 3.4 shows RBF centers trained with the second training data set. Figure 3.5 shows 3D surface plots and contour plots of the second training data set and RBF output when it is trained with the second training data set.

Discussions and conclusions: RBF trained with the second data set shows more consistent performance with different sets of validating data. Since this RBF will be used in the real learning

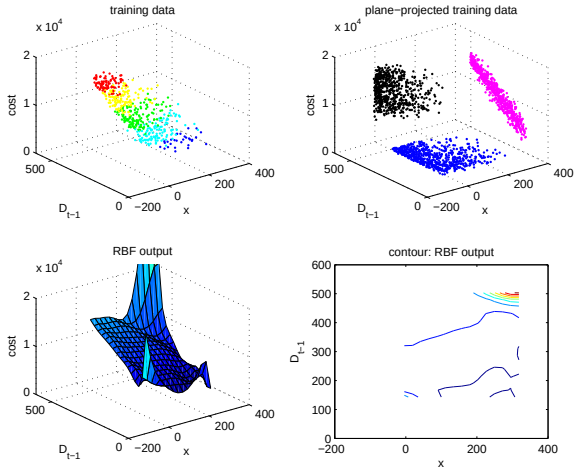

Figure 3.3: The first set data points and RBF output

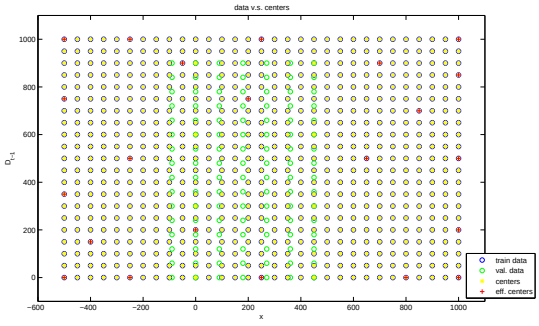

Figure 3.4: The second set data points and RBF centers

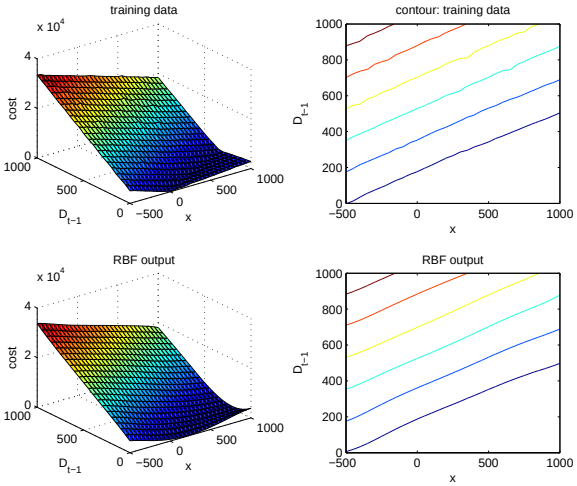

Figure 3.5: The second set data points and RBF output surface

environment, this property is desirable. Therefore, RBF set up with evenly distributed centers is recommended.

In addition, allocating RBF centers evenly throughout the state, or state-action, space is a method for incorporating domain knowledge into ADP. Evenly distributed structure of RBF centers is practical to set up; most adaptive inventory management schemes use either a look-up table or an aggregation technique to implement cost-to-go approximation. Assigning RBF centers can be located in the middle of each aggregate group. This affirms another advantage of RBF as a practical choice for cost-to-go approximation over feedforward neural networks, because its structure is easier to determine, especially in inventory management applications where range of state space is easily specified.

3.3 RBF Scales set up

As discussed in Section 3.2, RBF centers are set up to be evenly distributed. RBF scales correspond to centers, therefore, RBF scales can be set up to have the same values in each dimension. This study proposes a setup strategy for RBF scales based on a relative basis effect at a midpoint.

A midpoint is a virtual point surround by centers and the midpoint location is the centroid of the surrounding centers. The top right plot of Figure 3.6 shows midpoints marked as circles among RBF centers.

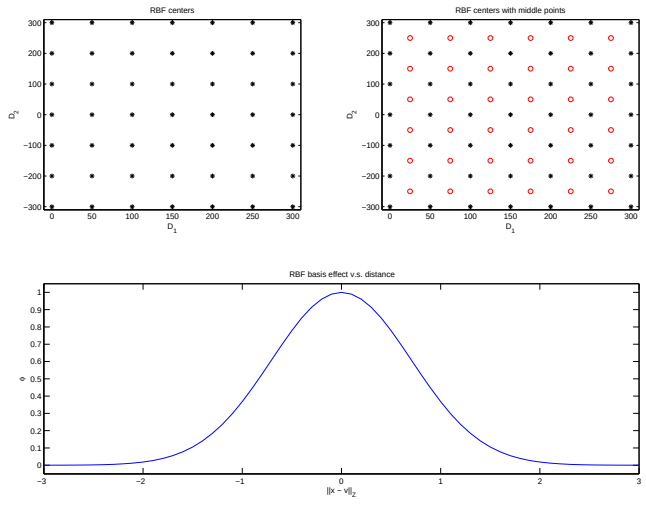

Figure 3.6: RBF centers and middle points

Each Gaussian basis has a peak value 1 at its center and declining as the distance from the center increases at a rate $\exp(-d_h^2)$, where d_h is the scaled distance from its center. The lower plot

of Figure 3.6 shows the Gaussian basis value versus the distance from its center. Midpoint strategy assigns scale values so the effect of each RBF center will drop to a specific fraction, e.g. half, of its peak value at the midpoint. For convenience, this strategy will be referred to as a midpoint strategy and the fraction will be prefixed to indicate the fraction of peak effect at the midpoint. Equation 3.10 shows a midpoint strategy formula to obtain scale along d^{th} dimension.

$$z_d = -\frac{4}{D \cdot \Delta_d^2} \cdot \log \phi \tag{3.10}$$

where z_d is a scale along d^{th} dimension; D is the number of dimensions; Δ_d is the gap between two adjacent centers along d^{th} dimension; and ϕ is the fraction of peak basis value at the midpoint, e.g., $\phi = 0.5$ for the 1/2-midpoint strategy.

For example, 1/10-, 1/2- and 9/10-midpoint strategies assign scales such that the effects of each RBF center will drop to 1/10, 1/2 and 9/10 of its peak value, respectively. It should be noted that RBF has $D \times M$ scales where D is a number of dimensions and M is a number of centers, but this midpoint strategy, used with evenly distributed centers, assigns all scales of the same dimension to the same value, i.e., $z_{d,1} = z_{d,2} = z_{d,3} = ... = z_{d,M}$. Therefore, for simplicity, subscript m is dropped.

3.4 Experiments

Pre-stage experiments investigate the structure of RBF centers. Using RBF with an inventory controlled ADP, however, raises questions, such as whether it will work, how to design its scales and how densely positioned RBF centers should be. To answer these questions, a series of simulation-based experiments are conducted.

Each experiment is repeated 50 times. Each repetition has a 60-period horizon. The problem is set up with K_t =\$80; c_t = \$100/unit; h_t = \$0.05/unit; and b_t = \$180/unit. This is a discounted problem with $\alpha = 0.95$ and zero leadtime, $L = 0$. The demand is modeled with AR1, where $a_0 = 2$, $a_1 = 0.8$, and demand noise is normally distributed with variance of 2. Each experiment is initialized at $D_0 = 50$ and $x_1 = 10$.

Experiments in this chapter investigate RBF as cost-to-go approximation. To focus on approximate cost-to-go and the learning aspect, H1 TD(0), a one-period Look-Ahead method with TD(0) approximate cost-to-go is used. Equation 3.11 shows the formulation of H1 TD(0)'s action. One-period expected Look-Ahead, H1, is used as a benchmark to clearly show the effect of the cost-to-go approximation. H1 selects an action based on the expected cost only, i.e., $a_t = \arg\min_a E[c(s_t)|a]$.

$$a_t = \arg\min_a E[c(s_t)|a] + \alpha \hat{Q}(\bar{s}_{t+1}, a) \tag{3.11}$$

where a_t is a chosen action; $E[c(s_t)|a]$ is an expected period cost of state s_t when action a is taken; $\hat{Q}$ is a state-action value, implemented by RBF and updated by TD(0); $\bar{s}_{t+1}$ is a projected state at time $t+1$ and $\bar{s}_{t+1} = [\bar{x}_{t+1}; \bar{D}_t]$ when demand forecast $\bar{D}_t = a_1 D_{t-1} + a_0$ and projected next period inventory $\bar{x}_{t+1} = x_t + a - \bar{D}_t$.

Given normally distributed demand noise, the expected single period cost $E[c(s_t)|a]$ can be obtained as shown in Equation 3.12.

$$\begin{aligned} E[c(s_t)|a] &= E[c([D_{t-1}, x_t]')|a] \\ &= K_t \cdot \delta(a) + c_t \cdot a + \bar{x}_{t+1} \cdot \{(h_t + b_t) \cdot \mathcal{N}(\bar{x}_{t+1}) - b_t\} + \sigma^2 \cdot (h_t + b_t) \cdot n(\bar{x}_{t+1}) \end{aligned} \quad (3.12)$$

where σ, $\mathcal{N}$, and n are the standard deviation, the cumulative Normal distribution function and the probability Normal distribution function of demand noise respectively. The other variables were previously defined.

Learning rate experiments: These experiments are designed to find a proper learning rate of TD(0) with RBF for comparison to H1. The RBF centers are set to each combination of {-30, -20, -10, 0, ..., 60} × {0, 10, 20, ..., 60} corresponding to x and D_{t-1} respectively. All scales are set to 0.0139. Learning rates, β, are 0.5, 0.6, 0.7, 0.8, 0.9, 0.95, 0.96, 0.97 and 0.99.

RBF scale experiments: These experiments are designed to contrast different midpoint strategies (Section 3.3), that is, the 0.1-, 0.2-, 0.3-, ..., and 0.9-midpoint strategies. Each treatment uses a learning rate of 0.96 and RBF centers at each combination of {-30, -20, -10, 0, ..., 60} × {0, 10, 20, ..., 60}.

RBF centers are positioned with gaps of 10 for both dimensions, i.e., $\Delta_1 = \Delta_2 = 10$, and the scales for both dimensions are the same, i.e., $z_1 = z_2 = -4/(2 \cdot 10^2) \cdot \log \phi$. Therefore, the scales are 0.0461, 0.0322, 0.0241, 0.0183, 0.0139, 0.0102, 0.0071, 0.0045, and 0.0021 for 0.1-, 0.2-, 0.3-, ..., 0.9-midpoint strategies, respectively.

RBF center spacing experiments: Given evenly distributed RBF centers, their density is determined by the gap sizes, or distances between two adjacent centers. These experiments explore the proper choice of gap size. Gap sizes for both dimensions are set to be the same. Gap sizes of 5, 10, and 15 are investigated. For a gap size of 5, RBF centers are set up to each combination {0,5,10,...,60} × {-30,-25,-20,...,60}. The scales are set to 0.0555. For gap size of 10, RBF centers are set up to each combination {0,10,20...,60} × {-30,-20,-10...,60}. The scales are set to 0.0139. For gap size of 15, RBF centers are set up to each combination {0,15,30,...,60} × {-30,-15,0,...,60}. The scales are set to 0.0062. Learning rates of 0.90, 0.95, 0.96, 0.97, and 0.99 are run for each treatment.

3.5 Experimental results

Learning rate experiments: Table 3.2 shows results of the significance tests comparing H1 TD(0) with H1.

Table 3.2: Significance tests: H1 and H1 TD(0) with different learning rates

treatment	sample	BCa interval		test	reject H_0		p value		Rank sum		Normal
β	mean	LCI	UCI	stat.	H_{a^+}	H_{a^-}	H_{a^+}	H_{a^-}	H_*	p val.	
Period 1-12; H1 (sample mean = 27,945.51; CI = [27,201.88 ; 28,710.86]); Normal test is passed											
0.50	32,097.55	31,182.95	32,978.16	-6.85	0	1	1.00	0.00	1	0.00	0
0.60	32,802.80	31,886.64	33,736.48	-7.95	0	1	1.00	0.00	1	0.00	0
0.70	32,491.47	31,587.17	33,372.66	-7.60	0	1	1.00	0.00	1	0.00	0
0.80	33,632.55	32,757.14	34,568.96	-9.38	0	1	1.00	0.00	1	0.00	0
0.90	33,169.37	32,268.52	34,065.45	-8.67	0	1	1.00	0.00	1	0.00	0
0.95	33,059.94	32,019.30	34,255.84	-7.38	0	1	1.00	0.00	1	0.00	0
0.96	33,390.51	32,363.24	34,437.34	-8.31	0	1	1.00	0.00	1	0.00	0
0.97	33,506.77	32,449.63	34,591.54	-8.30	0	1	1.00	0.00	1	0.00	0
0.99	33,385.38	32,302.24	34,578.88	-7.82	0	1	1.00	0.00	1	0.00	0
Period 13-60; H1 (sample mean = 59,236.30; CI = [57,413.74 ; 61,195.24]); Normal test is passed											
0.50	58,476.56	56,541.89	60,426.64	0.54	0	0	0.30	0.70	0	0.71	0
0.60	57,896.94	55,915.77	59,937.51	0.94	0	0	0.17	0.83	0	0.37	0
0.70	57,264.21	55,343.54	59,115.27	1.44	0	0	0.08	0.92	0	0.23	0
0.80	56,692.49	54,806.74	58,593.87	1.84	1	0	0.03	0.97	1	0.10	1
0.90	57,229.28	55,458.53	59,026.24	1.50	0	0	0.07	0.93	0	0.19	0
0.95	56,557.33	54,916.12	58,282.78	2.05	1	0	0.02	0.98	1	0.05	0
0.96	56,023.71	54,261.22	57,688.62	2.44	1	0	0.01	0.99	1	0.03	0
0.97	56,216.56	54,353.15	57,984.76	2.23	1	0	0.01	0.99	1	0.05	0
0.99	56,409.31	54,558.76	58,360.78	2.05	1	0	0.02	0.98	1	0.06	0
Period 1-60; H1 (sample mean = 87,181.81; CI = [85,118.41 ; 89,599.41]); Normal test is passed											
0.50	90,574.11	88,350.81	92,997.43	-2.04	0	1	0.98	0.02	1	0.03	0
0.60	90,699.75	88,490.03	93,242.83	-2.11	0	1	0.98	0.02	1	0.03	0
0.70	89,755.68	87,506.21	91,800.18	-1.62	0	0	0.95	0.05	1	0.07	0
0.80	90,325.04	88,256.07	92,457.66	-1.96	0	1	0.97	0.03	1	0.03	1
0.90	90,398.66	88,429.16	92,683.24	-2.01	0	1	0.98	0.02	1	0.04	0
0.95	89,617.27	87,459.97	91,735.29	-1.52	0	0	0.93	0.07	1	0.09	0
0.96	89,414.22	87,398.37	91,526.66	-1.42	0	0	0.92	0.08	1	0.10	0
0.97	89,723.33	87,621.15	91,954.30	-1.59	0	0	0.94	0.06	1	0.06	0
0.99	89,794.69	87,605.54	92,055.05	-1.59	0	0	0.94	0.06	0	0.10	0

The results are shown separately for Period 1-12, Period 13-60 and Period 1-60. Learning rates are shown in the column labeled treatment. Average aggregate cost is displayed. The Bias Corrected and accelerated percentile method (BCa) is used to obtain a confidence interval. The lower and upper levels of the 95% confidence interval are shown in columns LCI and UCI, under the BCa Interval. T-test test statistics are shown in the column labeled test stat. One-sided test results are shown in columns H_{a^+} and H_{a^-}, under the label reject H_0. An entry of 1 means H_0 can be rejected in favor of an alternative hypothesis. An entry of 0 means H_0 can not be rejected. The rejection is at a significance level of 0.05. A 1 in the null hypothesis H_0 column indicates that two means of aggregate cost show no significant differences. A 1 in the alternative hypothesis H_{a^+} column indicates that the mean of the control, e.g., H1, is significantly higher than the mean of the treatment. A 1 in the alternative hypothesis H_{a^-} column indicates that the mean of the control is significantly lower than

the mean of the treatment. The two columns under label p value show the p values of the test given the alternative hypothesis H_{a^+} and H_{a^-}, as indicated. Two-sided Wilcoxon Rank Sum test results with corresponding p values are shown in columns H_* and p val located under label Rank sum. The Wilcoxon Rank Sum test is done at a 0.10 significance level. Similarly, an entry of 1 in column H_* indicates the median of the control is significantly different than a median of the treatment. An entry of 0 indicates that the medians are not significantly different. Normality of the data is tested with both Chi Square good-of-fit and Lilliefors tests. The normality test result is shown as 1 if either the Chi Square good-of-fit or the Lilliefors test can reject the normality hypothesis (Data fails the normal test.), otherwise it is shown as 0. Each normality test is done at a 0.05 significance level.

Figure 3.7 displays the average inventory level and single-period cost of H1 compared to H1 TD(0) with $\beta = 0.96$. Measurement of H1 is labeled *No C2G* and drawn with a solid line without a marker. Measurement of H1 TD(0) is labeled *TD0* and is drawn with a solid line and asterisk markers.

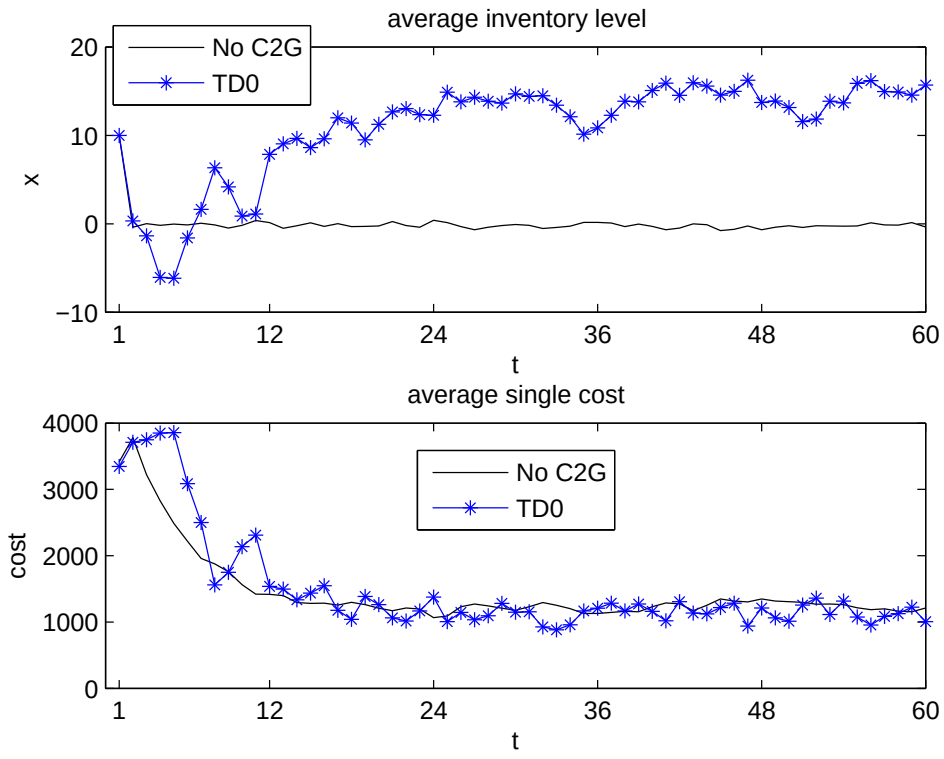

Figure 3.7: Average inventory level and single cost of H1 (No C2G) and H1 TD(0)

RBF scale experiments: Similar to Table 3.2, Table 3.3 shows significance tests comparing H1 to H1 TD(0) with different RBF scales. The scale column shows scales of each treatment. Each

scale corresponds to each midpoint strategy, i.e., scale 0.0461 corresponds to the $\frac{1}{10}$-midpoint, 0.0322 corresponds to the $\frac{2}{10}$-midpoint, 0.0241 corresponds to the $\frac{3}{10}$-midpoint, 0.0183 corresponds to the $\frac{4}{10}$-midpoint, 0.0139 corresponds to the $\frac{5}{10}$-midpoint, 0.0102 corresponds to the $\frac{6}{10}$-midpoint, 0.0071 corresponds to the $\frac{7}{10}$-midpoint, 0.0045 corresponds to the $\frac{8}{10}$-midpoint, and 0.0021 corresponds to the $\frac{9}{10}$-midpoint. The rest of the table is organized as Table 3.2.

Table 3.3: Significance tests: H1 and H1 TD(0) with different scales

scale	sample mean	BCa interval		test stat.	reject H_0		p value		Rank sum		Normal
		LCI	UCI		H_{a+}	H_{a-}	H_{a+}	H_{a-}	H_*	p val.	
Period 1-12; H1 (sample mean = 27,945.51; CI = [27,221.40 ; 28,706.98]); Normal test is passed											
0.0461	30,813.74	30,185.55	31,395.00	-5.80	0	1	1.00	0.00	1	0.00	0
0.0322	30,324.69	29,642.13	31,044.04	-4.50	0	1	1.00	0.00	1	0.00	1
0.0241	30,744.39	29,874.29	31,813.60	-4.48	0	1	1.00	0.00	1	0.00	0
0.0183	32,042.09	31,098.13	33,025.78	-6.52	0	1	1.00	0.00	1	0.00	0
0.0139	33,390.51	32,333.55	34,422.46	-8.31	0	1	1.00	0.00	1	0.00	0
0.0102	34,006.52	33,044.29	34,886.78	-9.88	0	1	1.00	0.00	1	0.00	0
0.0071	37,470.32	36,065.28	38,878.84	-11.51	0	1	1.00	0.00	1	0.00	0
0.0045	35,058.24	33,963.46	36,147.08	-10.49	0	1	1.00	0.00	1	0.00	0
0.0021	44,657.72	43,443.47	45,981.40	-21.93	0	1	1.00	0.00	1	0.00	0
Period 13-60; H1 (sample mean = 59,236.30; CI = [57,383.16 ; 61,111.33]); Normal test is passed											
0.0461	57,556.31	55,850.66	59,336.09	1.26	0	0	0.10	0.90	0	0.23	0
0.0322	57,202.62	55,593.41	58,751.90	1.60	0	0	0.06	0.94	0	0.17	0
0.0241	57,508.72	55,702.23	59,334.28	1.27	0	0	0.10	0.90	0	0.27	0
0.0183	56,302.65	54,402.87	58,269.68	2.10	1	0	0.02	0.98	1	0.04	0
0.0139	56,023.71	54,279.13	57,716.59	2.44	1	0	0.01	0.99	1	0.03	0
0.0102	57,873.05	55,903.16	59,945.16	0.96	0	0	0.17	0.83	0	0.35	0
0.0071	61,268.59	59,230.55	63,442.47	-1.40	0	0	0.92	0.08	0	0.21	0
0.0045	59,020.39	57,150.08	60,939.16	0.16	0	0	0.44	0.56	0	0.90	0
0.0021	59,298.88	57,307.97	61,544.55	-0.04	0	0	0.52	0.48	0	0.77	1
Period 1-60; H1 (sample mean = 87,181.81; CI = [85,042.00 ; 89,547.36]); Normal test is passed											
0.0461	88,370.05	86,443.68	90,494.27	-0.76	0	0	0.78	0.22	0	0.39	0
0.0322	87,527.31	85,646.92	89,506.70	-0.23	0	0	0.59	0.41	0	0.72	0
0.0241	88,253.11	86,053.77	90,537.55	-0.65	0	0	0.74	0.26	0	0.36	0
0.0183	88,344.74	86,084.97	90,697.38	-0.70	0	0	0.76	0.24	0	0.44	0
0.0139	89,414.22	87,368.81	91,490.51	-1.42	0	0	0.92	0.08	1	0.10	0
0.0102	91,879.57	89,716.74	94,235.22	-2.84	0	1	1.00	0.00	1	0.01	0
0.0071	98,738.91	96,243.49	101,529.59	-6.49	0	1	1.00	0.00	1	0.00	0
0.0045	94,078.62	92,111.74	96,235.68	-4.38	0	1	1.00	0.00	1	0.00	1
0.0021	103,956.60	101,606.90	106,612.12	-9.69	0	1	1.00	0.00	1	0.00	1

Figure 3.8 shows BCa confidence intervals of aggregate costs. It compares aggregate costs of H1, labeled H1 No C2G, with scale H1 TD(0) labeled with a fraction of its midpoint effect.

Figures 3.9, 3.10, and 3.11 show RBF outputs when using a scale of 0.0461 (1/10-midpoint), 0.0139 (5/10-midpoint) and 0.0021 (9/10-midpoint) respectively. In the top left 3D plot, x is the inventory level, D_p is the previous demand and a vertical axis is RBF output when all weights are equal to one. The top right plot is a contour plot. The bottom right plot is a contour plot with RBF centers as asterisks. The bottom left plot is a histogram of RBF output.

RBF center spacing experiments: Tables 3.4, 3.2 and 3.5 display significance test results of H1 TD(0) with gaps of 5, 10 and 15, respectively, compared to H1.

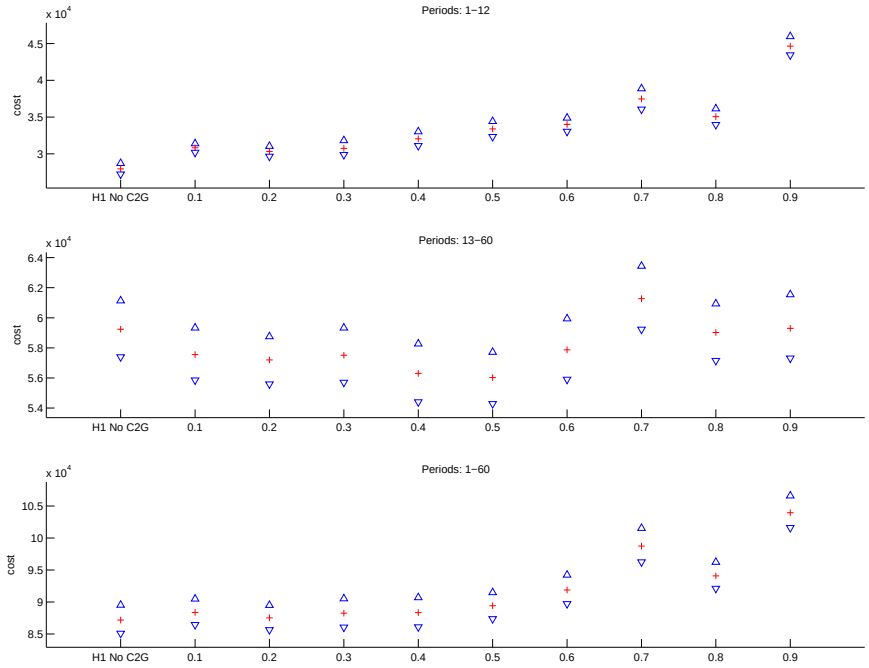

Figure 3.8: Midpoint comparisons

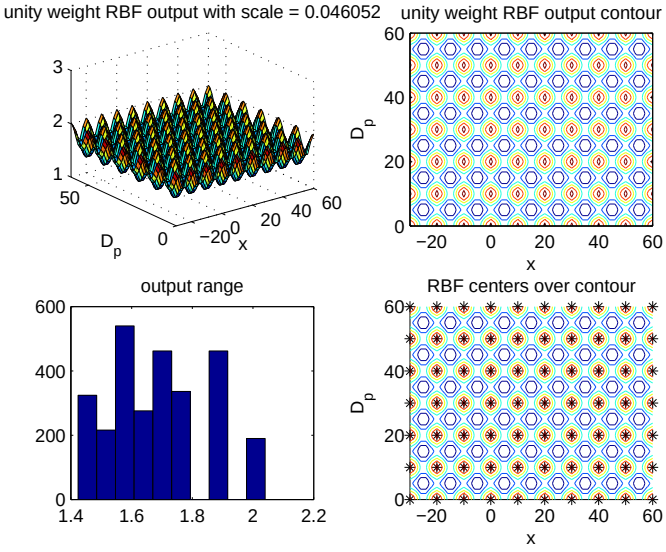

Figure 3.9: RBF bases with unity weight: 1/10-midpoint

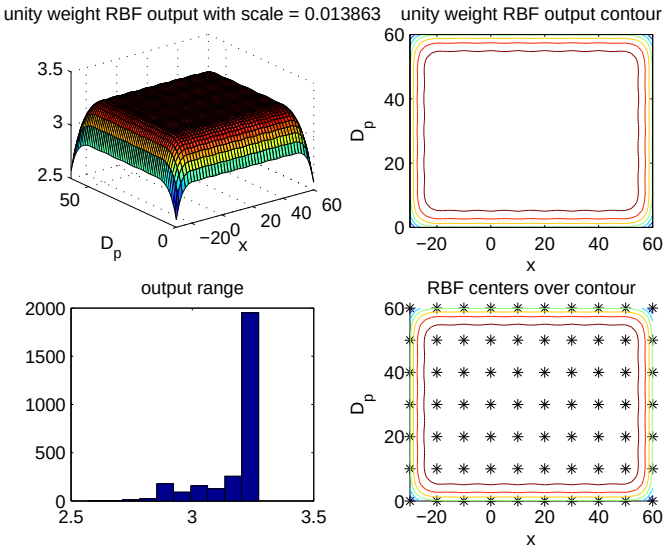

Figure 3.10: RBF bases with unity weight: 1/2-midpoint

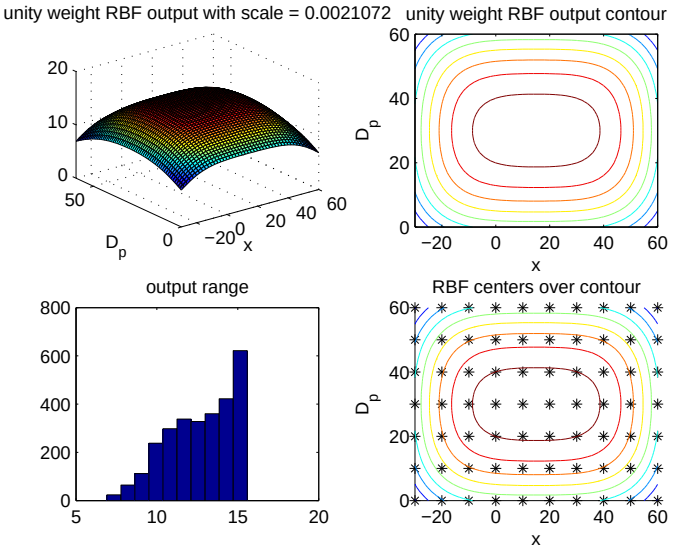

Figure 3.11: RBF bases with unity weight: 9/10-midpoint

Table 3.4: Significance tests: H1 and H1 TD(0) with center gap of 5

treatment	sample	BCa interval		test	reject H_0		p value		Rank sum		Normal
β	mean	LCI	UCI	stat.	H_{a+}	H_{a-}	H_{a+}	H_{a-}	H_*	p val.	
Period 1-12; H1 (sample mean = 27,945.51; CI = [27,216.45 ; 28,723.41]); Normal test is passed											
0.90	30,699.01	29,820.25	31,608.59	-4.54	0	1	1.00	0.00	1	0.00	0
0.95	30,729.30	30,020.12	31,501.48	-5.14	0	1	1.00	0.00	1	0.00	0
0.96	30,699.32	30,040.50	31,423.11	-5.20	0	1	1.00	0.00	1	0.00	0
0.97	30,544.92	29,652.06	31,416.99	-4.39	0	1	1.00	0.00	1	0.00	0
0.99	30,895.72	29,977.35	31,864.04	-4.70	0	1	1.00	0.00	1	0.00	0
Period 13-60; H1 (sample mean = 59,236.30; CI = [57,428.56 ; 61,221.31]); Normal test is passed											
0.90	58,070.28	56,178.54	60,167.56	0.82	0	0	0.21	0.79	0	0.39	0
0.95	57,811.53	55,965.00	59,674.92	1.04	0	0	0.15	0.85	0	0.35	0
0.96	58,176.84	56,417.72	60,021.68	0.78	0	0	0.22	0.78	0	0.43	0
0.97	58,567.34	56,718.75	60,532.27	0.49	0	0	0.31	0.69	0	0.64	0
0.99	57,949.32	56,059.71	59,975.40	0.92	0	0	0.18	0.82	0	0.39	0
Period 1-60; H1 (sample mean = 87,181.81; CI = [85,056.66 ; 89,563.37]); Normal test is passed											
0.90	88,769.29	86,643.50	91,192.69	-0.96	0	0	0.83	0.17	0	0.36	0
0.95	88,540.83	86,447.22	90,828.79	-0.84	0	0	0.80	0.20	0	0.35	0
0.96	88,876.17	86,910.74	91,120.72	-1.07	0	0	0.86	0.14	0	0.26	0
0.97	89,112.26	86,942.07	91,454.39	-1.18	0	0	0.88	0.12	0	0.23	0
0.99	88,845.04	86,599.10	91,206.65	-1.01	0	0	0.84	0.16	0	0.24	0

Table 3.5: Significance tests: H1 and H1 TD(0) with center gap of 15

treatment	sample	BCa interval		test	reject H_0		p value		Rank sum		Normal
β	mean	LCI	UCI	stat.	H_{a+}	H_{a-}	H_{a+}	H_{a-}	H_*	p val.	
Period 1-12; H1 (sample mean = 27,945.51; CI = [27,231.16 ; 28,706.11]); Normal test is passed											
0.90	34,724.47	33,949.73	35,515.76	-12.13	0	1	1.00	0.00	1	0.00	0
0.95	34,344.83	33,306.64	35,327.53	-9.76	0	1	1.00	0.00	1	0.00	1
0.96	33,846.46	33,010.50	34,686.85	-10.14	0	1	1.00	0.00	1	0.00	0
0.97	33,918.59	33,002.98	34,808.41	-9.91	0	1	1.00	0.00	1	0.00	0
0.99	33,674.43	32,932.66	34,432.33	-10.39	0	1	1.00	0.00	1	0.00	0
Period 13-60; H1 (sample mean = 59,236.30; CI = [57,423.55 ; 61,215.04]); Normal test is passed											
0.90	56,292.20	54,621.22	57,987.03	2.25	1	0	0.01	0.99	1	0.05	0
0.95	55,952.86	54,032.47	57,965.65	2.32	1	0	0.01	0.99	1	0.03	0
0.96	56,185.19	54,458.28	57,960.89	2.30	1	0	0.01	0.99	1	0.04	0
0.97	56,241.49	54,213.05	58,312.68	2.10	1	0	0.02	0.98	1	0.04	0
0.99	56,442.72	54,627.08	58,279.24	2.06	1	0	0.02	0.98	1	0.06	0
Period 1-60; H1 (sample mean = 87,181.81; CI = [84,958.70 ; 89,599.73]); Normal test is passed											
0.90	91,016.67	89,178.58	93,164.61	-2.48	0	1	0.99	0.01	1	0.01	0
0.95	90,297.70	88,037.29	92,541.79	-1.90	0	1	0.97	0.03	1	0.04	0
0.96	90,031.66	87,937.24	92,228.63	-1.78	0	1	0.96	0.04	1	0.07	0
0.97	90,160.07	87,954.67	92,584.41	-1.78	0	1	0.96	0.04	1	0.07	0
0.99	90,117.15	88,116.10	92,277.22	-1.84	0	1	0.97	0.03	1	0.07	0

Figure 3.12 shows BCa intervals of aggregate cost obtained from RBFs with gaps of 5 ($\beta = 0.95$), 10 ($\beta = 0.96$), and 15 ($\beta = 0.96$), as indicated on x-axis.

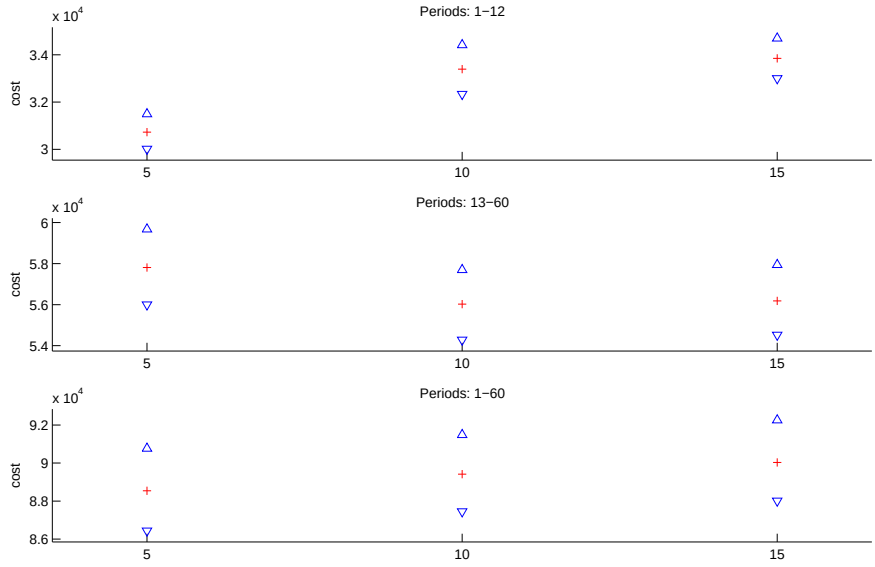

Figure 3.12: Center gap comparisons

3.6 Discussions and Conclusions

Learning rate: Using learning rates between 0.7 and 0.99 do not give significantly different results. Table 3.2 shows that the sample means obtained from these learning rates are within any BCa interval obtained from any learning rate between 0.7 and 0.99. H1 TD(0) with learning rate 0.8 and any learning rate from 0.95 and higher produces significantly better performance than H1 in the later 48 periods. This data illustrates that TD(0) is working well in later periods. Figure 3.7 shows that TD(0) increases average inventory level after the first few periods. As is typical of a learning-based method, TD(0) requires several periods to tune its cost-to-go approximation. Then, once the cost-to-go approximation is well tuned, it helps H1 TD(0) choose more appropriate actions. Consequently, TD(0) helps lower inventory cost.

RBF scale: Table 3.3 and Figure 3.8 show that in the later 48 periods midpoint effects around 0.5 perform better than midpoint effects near the extreme ends.

Figure 3.9 shows that, for a 1/10-midpoint scale, RBF produces a pronounced many peaked structure. A profile of the 3D surface plot on the top left of Fig. 3.9 shows many bumps, with each peak at an RBF center. This bumpy profile indicates potentially insufficient coverage of the state

space. A contour plot, top right, displays low value points located in the middle of four surrounding peaks. These points are at the midpoint locations, where combining basis function value is smallest. In the lower right plot, peaks are located at each RBF center. Figure 3.11 shows that for a 9/10-midpoint scale RBF produces a smooth structure. The 3D surface plot displays a smooth surface with a single gently rising peak in the middle. The single gently rising peak indicates a potential over-coverage of the state space. The single peak in the middle may be a result of overlapping effects from several wide spreading radial bases. Figure 3.10 shows that the 1/2-midpoint scale RBF has a sufficient degree of smoothness. The 3D surface plot displays a smooth flat surface. This profile indicates a high likelihood of adequate coverage of the state space. The histogram of outputs shows the majority of values being in the flat top. Half midpoint effect is intuitive in that it has a combining effect that sufficiently covers the area without excessive overlap. The half midpoint is also an effective RBF scale setting for a TD(0) application to inventory management, because it has low aggregate costs in later 48 periods (Table 3.3 and Figure 3.8).

RBF center: Tables 3.4, 3.2, 3.5 and Figure 3.12 show that in the later 48 periods H1 TD(0) with RBF center gaps of 10 and 15 perform better than those with a center gap of 5. The explanation may lie in the variance error. (See Geman et al. [44] for discussion about approximation error.) Using RBF gap of 5 results in having too many parameters, therefore more samples are required for RBF to converge.

This issue is common in model selection. It is a trade-off between a large variance error associated with too many parameters, and a large bias error associated with too few parameters. As discussed in Section 2.4.2, Akaike Information Criteria (AIC) can be used to determine this trade-off. AIC values of different controllers are displayed in Figure 3.13. For each controller, the numerical labels indicate gap size and the letters indicate a learning rate. For example, a: $\beta = 0.90$, b: $\beta = 0.95$, c: $\beta = 0.96$, d: $\beta = 0.97$, e: $\beta = 0.99$. For example, label 10c indicates an inventory control using TD(0) with an RBF gap size of 10 and $\beta = 0.96$. Among three center gaps investigated, a center gap of 10 yields the lowest AIC value, which means it gives the best trade-off between approximation power and model complexity.

AIC is calculated by first assigning states visited to groups, in which each group is characterized by an RBF center. A state is assigned to a group where the RBF center is closest to the state (Equation 3.13). Then the residual sum of squares of each group is calculated (Equation 3.14). Then AIC is the summation of the total residual sum of squares (Equation 3.15) and the model complexity (Equation 3.16).

It should be noted that multiple group assignment is possible when there is a state located in the middle between two centers. However, this will result in the same AIC value, because the centroid of each group is the RBF center, which is fixed. This is different from the application of K-means (as discussed in Section 2.4.2), where an assignment can change a value of the centroid of the group which the data point is assigned to.

$$m_s = \arg\min_{m'} \|\vec{s} - \vec{v}_{m'}\| \quad \text{for all } \vec{s} \tag{3.13}$$

$$\text{RSS}_m = \sum_{\vec{s}:m_s=m} \|\vec{s} - \vec{v}_m\|^2 \tag{3.14}$$

$$\text{RSS}(M) = \sum_{m=1}^{M} \text{RSS}_m \tag{3.15}$$

$$\text{AIC}(M) = \text{RSS}(M) + 2 \cdot M \cdot D \tag{3.16}$$

where $\vec{s}$ is a state visited, $\vec{v}_m$ is the m^{th} radial basis center, m_s is the group assignment of state $\vec{s}$, RSS_m is the RSS value of the m^{th} group, $\text{RSS}(M)$ is the summation of RSS values of M groups and $\text{AIC}(M)$ is the AIC value of RBF having M centers.

In a clustering application, AIC can be calculated based on the same data set. However, states visited in inventory control application depend on an inventory policy, which is in this case, H1 TD(0) with RBF. Therefore, with a different RBF, inventory may be controlled differently, which may result in different states visited. Figure 3.13 shows AIC values of each controller.

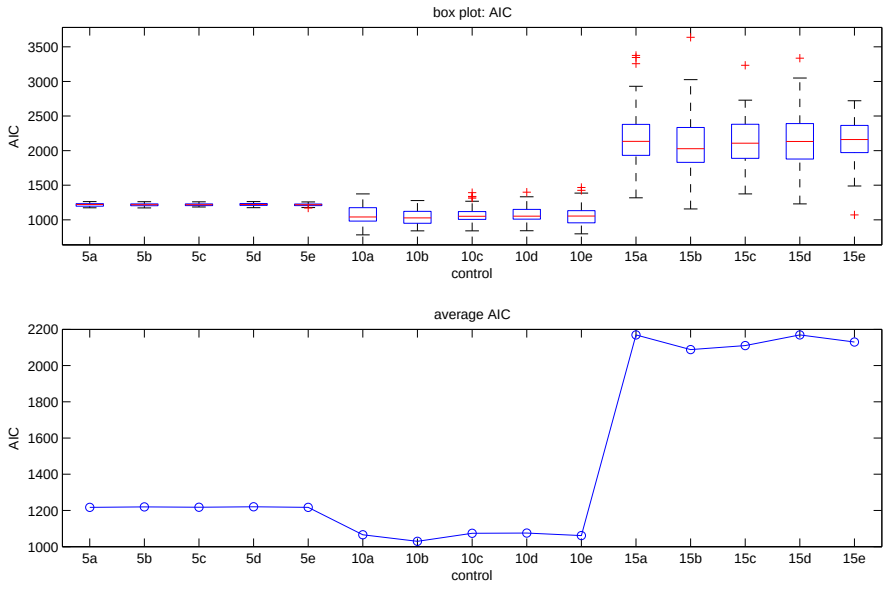

Figure 3.13: Boxplot and average AIC's of controllers with different center gap sizes

Each AIC value is calculated with corresponding data: for example, AIC of RBF with gap of 5 is calculated with data obtained from H1 TD(0) with RBF gap of 5. Using data obtained from each controller makes this analysis available only after each controller has been deployed and its results have been recorded.

Instead of using data obtained from each controller, common data may be used for AIC calculation to determine RBF structure for inventory control before implementing it. Figure 3.14 displays AIC values of different gap sizes.

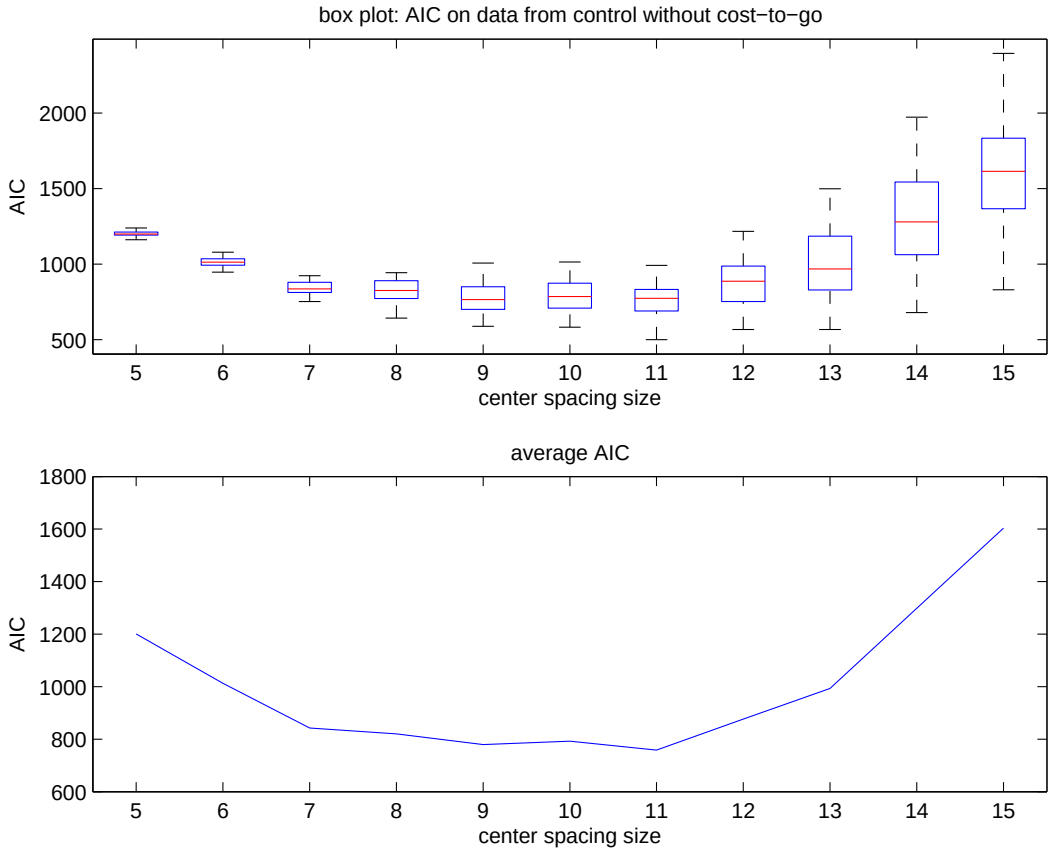

Figure 3.14: Boxplot and average common-data AIC's of different center spacing size.

All AIC values shown in this figure are calculated with common data obtained from H1. In practice, a history of an inventory system can provide this common data and, in a similar manner, a proper gap size can be determined from AIC prior to deployment of learning-based ADPs. Figure 3.14 shows that gap sizes of 9, 10, and 11 give the lowest AIC values indicating they are better settings than other gap sizes.

CHAPTER 4

LEARNING BASED CONTROLLERS

This chapter investigates solutions to inventory problems using the Temporal Difference (TD) learning techniques Sarsa, Sarsa(λ) and a Residual Gradient method. A new extension to the Residual Gradient method has also been developed as part of this research and is investigated. A Look-Ahead method is used as a benchmark for performance comparisons. These approaches are used both in a zero leadtime inventory problem and a one-period leadtime inventory problem. The investigation of a Sarsa(λ) application to an inventory problem in this research is also new. The development of Direct Credit Back is new as well.

4.1 Residual Gradient Method

The Residual Gradient method, introduced by Baird [7], is a learning-based ADP method similar to Sarsa, but it is designed to be used with various approximation functions, including nonlinear functions. Sarsa was originally designed to be implemented using a look-up table and later was extended for use with an approximation function. The Residual Gradient method is based on the temporal difference learning technique which uses information obtained at different times to update the state value approximation. Parameters $\vec{\theta}$ of a state cost approximation are updated with the gradient descent method. Equation 4.1 shows a total mean squared error of the approximation as a function of $\vec{\theta}$. The update formula is shown in Equation 4.3.

$$\xi(\vec{\theta}) = \sum_{t=1}^{T} \xi_t(\vec{\theta}) \tag{4.1}$$

$$\xi_t(\vec{\theta}) = \frac{1}{2}\left(Q(s_t) - \tilde{Q}(s_t|\vec{\theta})\right)^2 \tag{4.2}$$

$$\begin{aligned} \vec{\theta} &\leftarrow \vec{\theta} - \beta \cdot \nabla_{\vec{\theta}} \xi_t(\vec{\theta}) \\ &\leftarrow \vec{\theta} - \beta \cdot \left(Q(s_t) - \tilde{Q}(s_t|\vec{\theta})\right) \nabla_{\vec{\theta}} \left(Q(s_t) - \tilde{Q}(s_t|\vec{\theta})\right) \\ &\leftarrow \vec{\theta} + \beta \cdot \left(Q(s_t) - \tilde{Q}(s_t|\vec{\theta})\right) \nabla_{\vec{\theta}} \tilde{Q}(s_t|\vec{\theta}) \end{aligned} \tag{4.3}$$

where $Q(s_t)$ is an actual state cost, $\tilde{Q}(s_t|\vec{\theta})$ is an approximate state cost obtained with parameter $\vec{\theta}$, and β is a learning rate.

A linear parameter function is a function having a linear relationship between parameters and output. An approximate state cost can be calculated with a linear parameter function as $\tilde{Q}(s_t|\vec{w}) = \vec{w}^T \cdot \vec{\phi}(s_t)$, where $\vec{w}$ is a parameter vector and $\vec{\phi}(s_t)$ is a basis vector. This family of approximation functions includes the linear-mode RBF technique as mentioned in Section 2.4.2. Parameters $\vec{w}$ can be updated as shown in Equation 4.4.

$$\vec{w}' = \vec{w} + \beta \left(Q(s_t) - \tilde{Q}(s_t|\vec{w})\right) \vec{\phi}(s_t) \tag{4.4}$$

Since the real value $Q(s_t)$ is not known, the TD update formula approximates $Q(s_t)$ with $r_{t+1} + \alpha\tilde{Q}(s_{t+1}|\vec{w})$. The TD update formulas for state cost and state-action cost are shown in Equation 4.5 and Equation 4.6, respectively.

$$\vec{w}' = \vec{w} + \beta\left(r_{t+1} + \alpha\tilde{Q}(s_{t+1}|\vec{w}) - \tilde{Q}(s_t|\vec{w})\right)\vec{\phi}(s_t) \tag{4.5}$$

$$\vec{w}' = \vec{w} + \beta\left(r_{t+1} + \alpha\tilde{Q}(s_{t+1}, a_{t+1}|\vec{w}) - \tilde{Q}(s_t, a_t|\vec{w})\right)\vec{\phi}(s_t, a_t) \tag{4.6}$$

where $\vec{w}'$ is a vector of parameter values after an update.

The update formula of the Residual Gradient method is represented similarly. Rather than delaying the approximation of $Q(s_t)$ until after the gradient is determined, as in Equation 4.3, the Residual Gradient method approximates a state cost when $Q(s_t)$ first appears in Equation 4.2. The Residual Gradient error at time t is shown in Equation 4.7.

$$\tilde{\xi}_t(\vec{\theta}) = \frac{1}{2}\left(r_{t+1} + \alpha\tilde{Q}(s_{t+1}|\vec{\theta}) - \tilde{Q}(s_t|\vec{\theta})\right)^2 \tag{4.7}$$

The Residual Gradient update formulas for a linear parameter function for state cost and state-action cost are shown in Equation 4.8 and Equation 4.9, respectively.

$$\vec{w}' = \vec{w} - \beta\left(r_{t+1} + \alpha\tilde{Q}(s_{t+1}|\vec{w}) - \tilde{Q}(s_t|\vec{w})\right)\cdot\left(\alpha\vec{\phi}(s_{t+1}) - \vec{\phi}(s_t)\right) \tag{4.8}$$

$$\vec{w}' = \vec{w} - \beta\left(r_{t+1} + \alpha\tilde{Q}(s_{t+1}, a_{t+1}|\vec{w}) - \tilde{Q}(s_t, a_t|\vec{w})\right)\cdot\left(\alpha\vec{\phi}(s_{t+1}, a_{t+1}) - \vec{\phi}(s_t, a_t)\right) \tag{4.9}$$

The Residual Gradient method is used to control inventory in the same way as Sarsa, but the approximation function parameters are updated with the Residual Gradient formula (Equation 4.9) instead of Sarsa's.

4.2 Direct Credit Back

The inventor, Baird [7], claims that the Residual Gradient method always converges. However, it has been reported that the Residual Gradient method converges at a slower rate than Sarsa, whenever Sarsa converges. To improve the convergence rate of the Residual Gradient method, Direct Credit Back was developed in our research. It is seen to enhance the performance of the Residual Gradient method by using the trajectory of prior states, or prior state-action pairs.

When new information, such as a period cost r_{t+1}, is obtained, it can be used to update an approximate cost of the most recent state or other prior states. When the new information is used to update only an approximate cost of the most recent state, it is the update formula of the Residual Gradient method (Equation 4.9). However, if we apply this new information further back in time, then we will get a new updating formula, utilizing the trajectory of visited states.

For simplicity, consider state cost approximation first. Later the result can be extended to state-action cost approximation. Given a period cost r_{t+1}, a temporal difference of state s_t is noted as $\psi(s_t|r_{t+1})$ and can be calculated as shown in Equation 4.10.

$$\psi(s_t|r_{t+1}) = r_{t+1} + \alpha\tilde{Q}(s_{t+1}|\vec{w}_t) - \tilde{Q}(s_t|\vec{w}_t) \tag{4.10}$$

Temporal Differences of prior states can be derived in the same way, as shown in Equation 4.11. Equation 4.12 shows the recursive form of a temporal difference update.

$$\begin{aligned}
\psi(s_{t-1}|r_{t+1}) &= r_t + \alpha\cdot(r_{t+1} + \alpha\tilde{Q}(s_{t+1}|\vec{w}_t)) - \tilde{Q}(s_{t-1}|\vec{w}_t) \\
&= r_t + \alpha\cdot(\psi(s_t|r_{t+1}) + \tilde{Q}(s_t|\vec{w}_t)) - \tilde{Q}(s_{t-1}|\vec{w}_t) \\
\psi(s_{t-2}|r_{t+1}) &= r_{t-1} + \alpha\cdot(r_t + \alpha r_{t+1} + \alpha^2\tilde{Q}(s_{t+1}|\vec{w}_t)) - \tilde{Q}(s_{t-2}|\vec{w}_t) \\
&= r_{t-1} + \alpha\cdot(\psi(s_{t-1}|r_{t+1}) + \tilde{Q}(s_{t-1}|\vec{w}_t)) - \tilde{Q}(s_{t-2}|\vec{w}_t) \\
&\vdots \\
\psi(s_{t-i}|r_{t+1}) &= r_{t-i+1} + \alpha\cdot(r_{t-i+2} + \alpha r_{t-i+3} + \ldots + \alpha^{i-1}r_{t+1} + \alpha^i\tilde{Q}(s_{t+1}|\vec{w}_t)) - \tilde{Q}(s_{t-i}|\vec{w}_t) \\
&= r_{t-i+1} + \sum_{j=1}^{i}\alpha^j r_{t-i+j+1} + \alpha^{i+1}\tilde{Q}(s_{t+1}|\vec{w}_t) - \tilde{Q}(s_{t-i}|\vec{w}_t) \qquad (4.11) \\
&= r_{t-i+1} + \alpha\cdot\Big(\psi(s_{t-i+1}|r_{t+1}) + \tilde{Q}(s_{t-i+1}|\vec{w}_t)\Big) - \tilde{Q}(s_{t-i}|\vec{w}_t) \qquad (4.12)
\end{aligned}$$

where $i = 1, 2, \ldots, t-1$.

The temporal error, ξ_t, can be written as shown in Equation 4.13.

$$\xi_t = \frac{1}{2}\cdot\sum_{i=0}^{t-1}\psi^2(s_{t-i}|r_{t+1}) \tag{4.13}$$

Instead of directly crediting period cost backward, a credit back weight, λ, is introduced to control the back crediting effect as shown in Equation 4.14. The weighted temporal error is denoted as ξ_λ.

$$\xi_\lambda = \frac{1}{2}\cdot\sum_{i=0}^{t-1}\lambda^i\cdot\psi^2(s_{t-i}|r_{t+1}) \tag{4.14}$$

where a credit back weight $\lambda \in (0, 1]$.

When λ equals one, Equation 4.14 is equivalent to Equation 4.13. When λ equals zero, Equation 4.14 is equivalent to the Residual Gradient method.

Parameter values $\vec{w}$ can be corrected with a gradient descent method using the gradient of the total temporal error as shown in Equation 4.15.

$$\begin{aligned}
\vec{w}_{t+1} &= \vec{w}_t - \beta\cdot\nabla_{\vec{w}}\xi_\lambda \\
w_m^{(t+1)} &= w_m^{(t)} - \beta\sum_{i=0}^{t-1}\lambda^i\cdot\frac{\partial\frac{1}{2}\psi^2(s_{t-i}|r_{t+1})}{\partial w_m}
\end{aligned} \tag{4.15}$$

When the state cost is approximated with a linear parameter function $\tilde{Q}(s|\vec{w}) = \vec{w}^T \cdot \vec{\phi}(s)$, where $\phi_0(s) = 1$, the update formula can be written as shown in Equations 4.16 and 4.17.

$$w_0^{(t+1)} = w_0^{(t)} - \beta \sum_{i=0}^{t-1} \lambda^i \cdot \psi(s_{t-i}|r_{t+1}) \cdot \left(\alpha^{i+1} - 1\right) \tag{4.16}$$

$$w_m^{(t+1)} = w_m^{(t)} - \beta \sum_{i=0}^{t-1} \lambda^i \cdot \psi(s_{t-i}|r_{t+1}) \cdot \left(\alpha^{i+1} \cdot \phi_m(s_{t+1}) - \phi_m(s_{t-i})\right) \quad \text{for } m = 1, ..., M \tag{4.17}$$

where $0 < \lambda \leq 1$.

The implementation of Equation 4.16 and 4.17 requires computation and memory for the entire trajectory of states visited, making this approach inefficient for a long horizon problem. Truncation may be used as a simple expedient. Instead of crediting back to the beginning, a period cost can be credited back to a certain number of prior states. With variable N_{cb} denoted as the number of prior states to be credited, equation 4.18 shows a modified formula of the truncated Direct Credit Back method.

$$\vec{w}_{t+1} = \vec{w}_t - \beta \sum_{i=0}^{\min\{N_{cb}, t-1\}} \lambda^i \cdot \psi(s_{t-i}|r_{t+1}) \cdot \left(\alpha^{i+1} \cdot \vec{\phi}(s_{t+1}) - \vec{\phi}(s_{t-i})\right) \tag{4.18}$$

where $\vec{\phi}(s) = [\ 1\ ;\ \phi_1(s)\ ;\ \phi_2(s)\ ;\ \ldots\ ;\ \phi_M(s)\ ]$.

$$\psi(s_{t-i}|r_{t+1}) = r_{t-i+1} + \sum_{j=1}^{i} \alpha^j \cdot r_{t-i+j+1} + \alpha^{i+1} \tilde{Q}(s_{t+1}|\vec{w}_t) - \tilde{Q}(s_{t-i}|\vec{w}_t) \tag{4.19}$$

$$= r_{t-i+1} + \alpha \cdot \left(\psi(s_{t-i+1}|r_{t+1}) + \tilde{Q}(s_{t-i+1}|\vec{w}_t)\right) - \tilde{Q}(s_{t-i}|\vec{w}_t) \tag{4.20}$$

for $i = 0, 1, 2, \ldots, \min\{N_{cb}, t-1\}$ and $N_{cb} \in \{0, 1, 2, ...\}$.

This method requires memory storage for s_t, s_{t-1}, ..., $s_{\max\{1, t-N_{cb}\}}$ and r_{t+1}, r_t, ..., $r_{\max\{2, t-N_{cb}+1\}}$. When $N_{cb} = 0$, this update formula becomes the Residual Gradient update formula.

The update formula of state-action cost can be derived in a similar manner, as is shown in Equation 4.23.

$$\psi(s_t, a_t|r_{t+1}) = r_{t+1} + \alpha \cdot \tilde{Q}(s_{t+1}, a_{t+1}|\vec{w}_t) - \tilde{Q}(s_t, a_t|\vec{w}_t) \tag{4.21}$$

$$\psi(s_{t-i}, a_{t-i}|r_{t+1}) = r_{t-i+1} + \alpha \cdot \left(\psi(s_{t-i+1}, a_{t-i+1}|r_{t+1}) + \tilde{Q}(s_{t-i+1}, a_{t-i+1}|\vec{w}_t)\right) - \tilde{Q}(s_{t-i}, a_{t-i}|\vec{w}_t) \quad \text{for } i = 1, ..., \min\{t-1, N_{cb}\} \tag{4.22}$$

$$\vec{w}_{t+1} = \vec{w}_t - \beta \sum_{i=0}^{\min\{N_{cb}, t-1\}} \lambda^i \cdot \psi(s_{t-i}, a_{t-i}|r_{t+1}) \cdot \left(\alpha^{i+1} \cdot \vec{\phi}(s_{t+1}, a_{t+1}) - \vec{\phi}(s_{t-i}, a_{t-i})\right) \tag{4.23}$$

where $\vec{\phi}(s, a) = [\ 1\ ,\ \phi_1(s, a)\ ,\ \phi_2(s, a)\ ,\ \ldots\ ,\ \phi_M(s, a)\ ]$ and $N_{cb} \in \{0, 1, 2, \ldots\}$.

Direct Credit Back(λ, N_{cb}) can be used to control inventory in the same way as Sarsa. The procedure is summarized in Table 4.1. The algorithm shown in Table 4.1 is similar to Sarsa(λ) but the update formula is changed to the Direct Credit Back formula.

Table 4.1: Direct Credit Back with linear RBF

(1)	Set $t = 1$
	Set up RBF
	a) determine M
	b) determine $\vec{v}_m$ and $\mathbf{Z}_m$ for $m = 1, ..., M$
	c) determine $\vec{w}_t$
(2)	Observe state s_t
(3)	Decide action, $a_t \leftarrow \pi(s_t \vert \tilde{Q})$
(4)	Take action (place replenishment order if decide to do so) according to action a_t
	Observe the consequence:
	a) record actual consequence and speculate next state s_{t+1}
	b) evaluate period cost r_{t+1}
	c) speculate next action $a_{t+1} \leftarrow \pi(s_{t+1} \vert \tilde{Q})$
(5)	Update an approximation function
	a) Compute credit back TD
	i) $\psi(s_t, a_t \vert r_{t+1}) = r_{t+1} + \alpha \cdot \tilde{Q}(s_{t+1}, a_{t+1} \vert \vec{w}_t) - \tilde{Q}(s_t, a_t \vert \vec{w}_t)$
	ii) $\psi(s_{t-i}, a_{t-i} \vert r_{t+1}) = r_{t-i+1} + \alpha \cdot \Big(\psi(s_{t-i+1}, a_{t-i+1} \vert r_{t+1}) + \tilde{Q}(s_{t-i+1}, a_{t-i+1} \vert \vec{w}_t)\Big)$ $-\tilde{Q}(s_{t-i}, a_{t-i} \vert \vec{w}_t)$ for $i = 1, ..., \min\{t-1, N_{cb}\}$
	b) Update weights
	$\vec{w} \leftarrow \vec{w} - \beta \sum_{i=0}^{\min\{N_{cb}, t-1\}} \lambda^i \psi(s_{t-i}, a_{t-i} \vert r_{t+1}) \Big(\alpha^{i+1} \cdot \vec{\phi}(s_{t+1}, a_{t+1}) - \vec{\phi}(s_{t-i}, a_{t-i})\Big)$
(6)	Transition (to next period)
	a) Set $t \leftarrow t + 1$
	b) Repeat step (4) to (6)

4.3 Experiments: a zero leadtime problem

This section discusses the application of Sarsa, Residual Gradient, Sarsa(λ) and Direct Credit Back to a zero leadtime inventory problem. Simulation-based experiments were conducted to evaluate how each method performs. Each experiment is repeated 50 times. Each repetition has a 60-period horizon. The problem is set up with K_t =\$80, c_t = \$100/unit, h_t = \$0.05/unit, and b_t = \$180/unit. This is a discounted problem with $\alpha = 0.95$ and zero leadtime, $L = 0$. The demand is modeled with AR1, where $a_0 = 2$, $a_1 = 0.8$, and the demand noise is normally distributed with a variance of 2. Each experiment is initialized at $D_0 = 50$ and $x_1 = 10$. With zero leadtime, state s_t has only two dimensions: previous demand D_{t-1} and on-site inventory level x_t.

Look-Ahead: The Look-Ahead method used in this section is based on average projection, assuming zero noise $e_t = 0$. Look-Ahead controllers are used with 1 to 5 look-ahead periods.

Sarsa: As introduced in Section 2.4, Sarsa is an algorithm to control processes with the temporal difference learning technique and cost-to-go approximation. Four learning rates 0.01, 0.1, 1 and 10 are used. The decision is chosen based on approximate state-action costs. These experiments utilize simulated annealing to search for an action having the minimum state-action cost for the given state. The degree of exploration is assumed to be covered by zero initialization of approximate state-action costs and heuristic nature of simulated annealing. Since this is a minimization problem and state-action costs are non-negative values, initialization to zero of approximate state-action costs will result in actions not previously taken being picked. A random start and the suboptimal nature of simulated annealing also add a greater degree of exploration to the action selection process.

Simulated annealing, introduced by Kirkpatrick et al. [72], is a probabilistic search technique for optimization problems. This technique was inspired by a metal annealing process developed to control metal ductility and hardness by scheduling cool down time. While searching, the algorithm uses an analog to time-temperature scheduling to control the degree of exploration and to provide a mechanism to prevent the program from getting stuck at a local optimum. Schneider and Kirkpatrick [104] explained that when the time-temperature schedule is slow enough, simulated annealing will find the global optimal solution. Table 4.2 shows a simulated annealing algorithm. This table is based on Russell and Norvig [102].

Table 4.2: Simulated Annealing

	problem: find $x = \arg\min_x f(x)$ with subject to $x \in \Omega$ input: a) objective function $f(x)$ b) search space Ω c) simulated annealing parameters: time-temperature mapping, i.e., $\tau = 1/\log(t)$
(1)	Initialize x Set $t = 1$
(2)	Calculate annealing temperature τ Pick x' from a neighborhood function $\Phi(x)$ Set $\Delta f = f(x') - f(x)$
(3)	If $\Delta f < 0$ Then set $x \leftarrow x'$ Else set $x \leftarrow x'$ with probability $\exp(-\Delta f/\mathring{T})$
(4)	Set $t \leftarrow t + \Delta t$ Repeat step (2) to (3) until t has reached the maximum time $t_{\max}$

The time-temperature mapping τ can be defined such that $\tau = 1/\log(t)$. At each step, a possible solution x' is picked from the neighborhood function, $\Phi(x)$. In this study, the neighborhood function

$\Phi(x)$ is a finite (truncated) Normal distributed function (Section 7.1) that is a feasible value chosen to be close to x. A new solution x is chosen and the annealing time t is incremented by a time step Δt. The algorithm repeats this process until the final annealing time $t_{\max}$ is reached. The new solution is chosen from either a current solution x or a candidate solution x'. A candidate solution will be chosen under two circumstances: first, when it gives a lower cost than the current solution, and second, if the candidate solution cannot give a lower cost, it will be chosen with probability such that the candidate is more likely to be chosen if its cost is closer to the current solution cost. Such a probability function can be formulated as $\exp(-\Delta f/\tau)$ where $\Delta f = f(x') - f(x)$. The probability function $\exp(-\Delta f/\tau)$ will equal one when a candidate results in the same cost function as the current solution. Given $\Delta f \geq 0$, the exponential function will ensure that the probability will be between 0 and 1. The non-negative annealing temperature τ controls sensitivity of the probability to Δf. Simulated annealing in this study is set up with time step Δt of 0.01 and final time $t_{\max}$ of 6. (See Russell and Norvig [102] and Schneider and Kirkpatrick [104] for a discussion of Simulated Annealing)

RBF is used as a state-action cost approximation function and it is set up with fixed centers and scales. Centers are three dimensional and evenly distributed. The three dimensions of a RBF center correspond to previous demand, the on-site inventory level and the inventory level after replenishment. The RBF structure has centers at each combination of $\{0, 10, 20, ..., 60\} \times \{-30, -20, -10, 0, 10, ..., 60\} \times \{0, 10, 20, ..., 60\}$. All RBF scales are set to 0.0139.

Eligibility Trace: An eligibility trace technique[1] is an extension of Sarsa that utilizes a state-action trajectory to improve learning speed. Instead of using new information to correct only the most recent approximate state-action cost, The eligibility trace technique partially credits backward through multiple prior state-action pairs.

Experiments here use a Sarsa version of an eligibility trace, Sarsa(λ), which is used with eligibility factors λ of 0, 0.5 and 1. It should be noted that $\lambda = 0$ is equivalent to Sarsa (without eligibility trace). Learning rates of 0.01, 0.1, 1 and 10 are investigated. Other parameters are specified as Sarsa controllers (as mentioned in Sarsa paragraph).

Residual Gradient: Residual Gradient is implemented with the same settings as Sarsa. Four learning rates β = 0.01, 0.1, 1 and 10 are investigated and RBF is used as a state-action cost approximation function, as also was done with the Sarsa set up.

[1] As discussed in Sutton and Barto [114], an eligibility trace technique is introduced by Watkins [123]. The version used here is based on Sutton and Barto [114], which in turn contributed by Jaakkola et al. [63].

Direct Credit Back: Direct Credit Back controllers are investigated for the credit back horizon $N_{cb} = 0$, 1, 10 and 100 with credit back weights $\lambda = 0$, 0.5 and 1. It should be noted that when $N_{cb} = 0$, the calculation is equivalent to the Residual Gradient method. Thus credit back weight λ has no effect on the approximate cost. When $N_{cb} = 100$, the period cost will be credited back through the entire trajectory[2]. Learning rates of 0.01, 0.1, 1 and 10 are used. RBF is used as a state-action cost approximation function, also with the Sarsa set up.

4.4 Experimental results: a zero leadtime problem

The results of the significance tests of Look-Ahead, Sarsa, Sarsa(λ), Residual Gradient and Direct Credit Back are shown in Tables 4.3 , 4.4 , 4.5 , 4.6 and 4.7, respectively.

Table 4.3: Significance tests: Look-Ahead

	sample	BCa interval		test	reject H_0		p value		Rank sum		Normal
treatment	mean	LCI	UCI	stat.	H_{a^+}	H_{a^-}	H_{a^+}	H_{a^-}	H_*	p val.	
Period 1-12; H1 (sample mean = 27,789.58; CI = [27,148.48 ; 28,330.76]); normal test is passed.											
H2	27,858.82	27,252.50	28,487.54	-0.16	0	0	0.56	0.44	0	0.91	0
H3	27,363.45	26,661.20	28,084.05	0.89	0	0	0.19	0.81	0	0.22	0
H4	27,032.24	26,366.66	27,695.39	1.65	0	0	0.05	0.95	1	0.07	0
H5	27,350.67	26,774.71	27,880.05	1.06	0	0	0.15	0.85	0	0.15	0
Period 13-60; H1 (sample mean = 58,060.49; CI = [56,219.24 ; 59,944.04]); normal test is passed.											
H2	57,182.68	55,265.67	59,053.78	0.64	0	0	0.26	0.74	0	0.54	0
H3	56,605.75	54,861.59	58,494.28	1.09	0	0	0.14	0.86	0	0.29	0
H4	55,562.06	53,915.40	57,357.28	1.90	1	0	0.03	0.97	1	0.08	0
H5	54,266.68	52,607.94	55,940.83	2.92	1	0	0.00	1.00	1	0.01	0

Table 4.4: Significance tests: Sarsa

treatment	sample	BCa interval		test	reject H_0		p value		Rank sum		Normal
β	mean	LCI	UCI	stat.	H_{a^+}	H_{a^-}	H_{a^+}	H_{a^-}	H_*	p val.	
Period 1-12; H1 (sample mean = 27,789.58; CI = [27,141.21 ; 28,323.89]); normal test is passed.											
0.01	35,407.23	34,418.94	36,397.73	-12.89	0	1	1.00	0.00	1	0.00	0
0.1	34,632.42	33,719.20	35,565.06	-12.19	0	1	1.00	0.00	1	0.00	0
1	35,542.00	34,520.53	36,521.42	-12.76	0	1	1.00	0.00	1	0.00	0
10	34,981.11	33,906.57	36,043.67	-11.34	0	1	1.00	0.00	1	0.00	0
Period 13-60; H1 (sample mean = 58,060.49; CI = [56,209.72 ; 60,027.56]; normal test is passed.											
0.01	53,143.27	51,605.57	54,942.06	3.80	1	0	0.00	1.00	1	0.00	0
0.1	51,535.10	49,579.96	53,645.14	4.59	1	0	0.00	1.00	1	0.00	1
1	51,213.10	49,279.76	53,309.58	4.83	1	0	0.00	1.00	1	0.00	0
10	54,160.33	52,496.55	56,019.67	2.95	1	0	0.00	1.00	1	0.01	0

The first column of Table 4.3, 4.4 and 4.6, the first two columns of Table 4.5 and the first three columns of Table 4.7 indicate the settings of corresponding inventory control methods. The tables also provide sample means and confidence intervals. The bias corrected and accelerated percentile method (BCa) is used to obtain a confidence interval. The lower and upper levels of

[2] Since the problem is set for a 60-period horizon, any $N_{cb} \geq 60$ would result in the same learning process.

Table 4.5: Significance tests: Sarsa(λ)

treatment		sample	BCa interval		test	reject H_0		p value		Rank sum		Normal
λ	β	mean	LCI	UCI	stat.	H_{a^+}	H_{a^-}	H_{a^+}	H_{a^-}	H_*	p val.	
Period 1-12; H1 (sample mean = 27,789.58; CI = [27,143.54 ; 28,323.55]); normal test is passed.												
0	a	35,407.23	34,401.06	36,393.19	-12.89	0	1	1.00	0.00	1	0.00	0
0	b	34,632.42	33,730.15	35,564.37	-12.19	0	1	1.00	0.00	1	0.00	0
0	c	35,542.00	34,472.75	36,543.96	-12.76	0	1	1.00	0.00	1	0.00	0
0	d	34,981.11	33,898.95	36,082.69	-11.34	0	1	1.00	0.00	1	0.00	0
0.5	a	35,063.91	34,158.20	36,049.02	-12.77	0	1	1.00	0.00	1	0.00	0
0.5	b	36,297.45	35,322.81	37,374.35	-13.85	0	1	1.00	0.00	1	0.00	0
0.5	c	34,948.13	34,041.26	35,862.37	-12.88	0	1	1.00	0.00	1	0.00	0
0.5	d	35,006.06	34,148.65	36,049.04	-12.57	0	1	1.00	0.00	1	0.00	0
1	a	34,964.63	34,107.02	35,934.46	-12.79	0	1	1.00	0.00	1	0.00	1
1	b	35,913.01	35,105.20	36,872.71	-14.99	0	1	1.00	0.00	1	0.00	0
1	c	35,936.48	34,945.79	36,967.32	-13.44	0	1	1.00	0.00	1	0.00	0
1	d	36,247.73	35,455.49	37,110.80	-16.18	0	1	1.00	0.00	1	0.00	0
Period 13-60; H1 (sample mean = 58,060.49; CI = [56,236.47 ; 59,933.75]); normal test is passed.												
0	a	53,143.27	51,638.36	54,958.73	3.80	1	0	0.00	1.00	1	0.00	0
0	b	51,535.10	49,578.76	53,632.41	4.59	1	0	0.00	1.00	1	0.00	1
0	c	51,213.10	49,277.47	53,342.58	4.83	1	0	0.00	1.00	1	0.00	0
0	d	54,160.33	52,475.18	55,954.37	2.95	1	0	0.00	1.00	1	0.01	0
0.5	a	54,491.72	52,769.12	56,226.62	2.70	1	0	0.00	1.00	1	0.01	0
0.5	b	53,875.38	52,275.92	55,500.08	3.28	1	0	0.00	1.00	1	0.00	0
0.5	c	52,816.64	51,247.53	54,407.37	4.13	1	0	0.00	1.00	1	0.00	0
0.5	d	53,929.55	52,248.13	55,576.51	3.19	1	0	0.00	1.00	1	0.00	0
1	a	54,732.18	53,271.92	56,242.39	2.69	1	0	0.00	1.00	1	0.01	0
1	b	54,825.50	53,322.65	56,397.54	2.59	1	0	0.01	0.99	1	0.01	0
1	c	54,274.57	52,741.28	55,900.43	2.99	1	0	0.00	1.00	1	0.00	0
1	d	52,163.76	50,605.74	53,756.90	4.67	1	0	0.00	1.00	1	0.00	0
Remark β coding: 'a' for $\beta = 0.01$, 'b' for $\beta = 0.1$, 'c' for $\beta = 1$, and 'd' for $\beta = 10$												

Table 4.6: Significance tests: Residual Gradient

treatment	sample	BCa interval		test	reject H_0		p value		Rank sum		Normal
β	mean	LCI	UCI	stat.	H_{a^+}	H_{a^-}	H_{a^+}	H_{a^-}	H_*	p val.	
Period 1-12; H1 (sample mean = 27,789.58; CI = [27,155.83 ; 28,341.06]); normal test is passed.											
0.01	33,994.76	32,895.20	36,981.22	-6.77	0	1	1.00	0.00	1	0.00	1
0.1	33,755.77	32,819.06	35,101.80	-9.17	0	1	1.00	0.00	1	0.00	1
1	34,940.99	33,154.22	38,113.58	-5.80	0	1	1.00	0.00	1	0.00	1
10	34,014.70	32,729.09	36,150.75	-6.98	0	1	1.00	0.00	1	0.00	1
Period 13-60; H1 (sample mean = 58,060.49; CI = [56,171.34 ; 60,031.17]); normal test is passed.											
0.01	55,587.89	53,615.89	57,903.67	1.69	1	0	0.05	0.95	1	0.06	0
0.1	54,700.96	52,900.14	57,335.16	2.27	1	0	0.01	0.99	1	0.01	0
1	57,498.39	54,208.76	64,295.62	0.22	0	0	0.41	0.59	1	0.03	1
10	56,820.45	54,100.01	66,549.99	0.48	0	0	0.31	0.69	1	0.02	1

the 95% confidence interval are shown in column LCI and UCI. T-test statistics are shown in a column labeled "test stat". One-sided test results are shown in columns H_{a^+} and H_{a^-}, under the label "reject H_0". An entry of 1 means H_0 can be rejected in favor of an alternative hypothesis. An entry of 0 means H_0 cannot be rejected. The rejection is at a significance level of 0.05. An entry of 1 in the null hypothesis H_0 column indicates that two means of aggregate cost show no significant differences. An entry of 1 in the alternative hypothesis H_{a^+} column indicates that the mean of the control, H1, is significantly higher than the mean of the treatment. An entry of 1 in

Table 4.7: Significance tests: Direct Credit Back

treatment			sample	BCa interval		test	reject H_0		p value		Rank sum		Normal
N_{cb}	λ	β	mean	LCI	UCI	stat.	H_{a+}	H_{a-}	H_{a+}	H_{a-}	H_*	p val.	
Period 1-12; H1 (sample mean = 27,789.58; CI = [27,147.57 ; 28,320.51]); normal test is passed.													
0		a	33,994.76	32,851.90	36,868.85	-6.77	0	1	1.00	0.00	1	0.00	1
0		b	33,755.77	32,821.09	35,168.64	-9.17	0	1	1.00	0.00	1	0.00	1
0		c	34,940.99	33,143.84	38,177.51	-5.80	0	1	1.00	0.00	1	0.00	1
0		d	34,014.70	32,724.35	36,180.87	-6.98	0	1	1.00	0.00	1	0.00	1
1	0.5	a	34,662.19	33,520.75	36,967.37	-7.87	0	1	1.00	0.00	1	0.00	1
1	0.5	b	34,638.65	33,313.37	37,220.83	-7.07	0	1	1.00	0.00	1	0.00	1
1	0.5	c	34,445.41	33,327.88	36,211.47	-8.60	0	1	1.00	0.00	1	0.00	1
1	0.5	d	36,779.08	35,588.14	38,554.10	-11.39	0	1	1.00	0.00	1	0.00	1
1	1	a	35,042.44	33,230.72	38,908.59	-5.51	0	1	1.00	0.00	1	0.00	1
1	1	b	34,772.67	33,750.64	37,022.29	-8.52	0	1	1.00	0.00	1	0.00	1
1	1	c	33,251.07	32,112.88	35,115.79	-6.94	0	1	1.00	0.00	1	0.00	1
1	1	d	36,206.79	35,165.39	37,568.92	-12.21	0	1	1.00	0.00	1	0.00	1
10	0.5	a	34,696.94	33,273.57	37,823.74	-6.43	0	1	1.00	0.00	1	0.00	1
10	0.5	b	34,401.69	33,285.30	36,216.70	-8.44	0	1	1.00	0.00	1	0.00	1
10	0.5	c	35,840.59	33,902.85	38,789.38	-6.35	0	1	1.00	0.00	1	0.00	1
10	0.5	d	37,007.59	36,096.72	38,111.27	-15.42	0	1	1.00	0.00	1	0.00	0
10	1	a	33,894.88	32,782.27	35,665.88	-7.83	0	1	1.00	0.00	1	0.00	1
10	1	b	33,297.36	32,327.45	34,693.63	-8.18	0	1	1.00	0.00	1	0.00	0
10	1	c	37,686.39	36,203.34	40,871.63	-8.83	0	1	1.00	0.00	1	0.00	1
10	1	d	36,607.43	35,696.05	37,663.84	-14.89	0	1	1.00	0.00	1	0.00	1
100	0.5	a	33,674.03	32,610.34	36,218.58	-6.84	0	1	1.00	0.00	1	0.00	1
100	0.5	b	34,173.42	33,048.01	35,615.84	-8.99	0	1	1.00	0.00	1	0.00	1
100	0.5	c	34,983.94	33,703.25	36,757.53	-8.62	0	1	1.00	0.00	1	0.00	1
100	0.5	d	36,284.28	35,192.89	37,609.53	-12.33	0	1	1.00	0.00	1	0.00	1
100	1	a	33,911.69	32,805.75	35,813.93	-7.78	0	1	1.00	0.00	1	0.00	1
100	1	b	35,397.40	33,960.78	37,679.13	-7.75	0	1	1.00	0.00	1	0.00	1
100	1	c	37,057.67	35,818.47	39,131.61	-10.71	0	1	1.00	0.00	1	0.00	1
100	1	d	35,880.47	34,876.03	37,194.22	-12.08	0	1	1.00	0.00	1	0.00	1
Period 13-60; H1 (sample mean = 58,060.49; CI = [56,242.83 ; 59,941.70]); normal test is passed.													
0		a	55,587.89	53,650.13	57,926.67	1.69	1	0	0.05	0.95	1	0.06	0
0		b	54,700.96	52,914.53	57,267.32	2.27	1	0	0.01	0.99	1	0.01	0
0		c	57,498.39	54,248.41	64,125.98	0.22	0	0	0.41	0.59	1	0.03	1
0		d	56,820.45	54,063.00	66,043.75	0.48	0	0	0.31	0.69	1	0.02	1
1	0.5	a	55,779.59	53,454.00	58,997.83	1.35	0	0	0.09	0.91	1	0.02	1
1	0.5	b	55,659.78	53,584.87	60,991.67	1.27	0	0	0.10	0.90	1	0.01	1
1	0.5	c	56,138.67	54,130.99	60,493.63	1.09	0	0	0.14	0.86	1	0.05	1
1	0.5	d	54,252.52	52,664.92	55,805.66	3.02	1	0	0.00	1.00	1	0.00	0
1	1	a	56,821.23	54,146.38	62,302.00	0.58	0	0	0.28	0.72	1	0.09	1
1	1	b	55,233.58	53,419.52	57,643.19	1.99	1	0	0.02	0.98	1	0.02	0
1	1	c	55,931.36	54,199.29	57,808.99	1.59	0	0	0.06	0.94	0	0.13	0
1	1	d	54,060.46	52,591.25	55,729.11	3.15	1	0	0.00	1.00	1	0.00	0
10	0.5	a	55,946.79	53,896.58	60,175.03	1.20	0	0	0.12	0.88	1	0.03	1
10	0.5	b	56,797.08	54,976.69	59,566.73	0.85	0	0	0.20	0.80	0	0.17	1
10	0.5	c	58,607.22	56,068.21	62,165.90	-0.30	0	0	0.62	0.38	0	0.55	1
10	0.5	d	53,940.14	52,493.96	55,498.34	3.31	1	0	0.00	1.00	1	0.00	0
10	1	a	57,187.90	55,460.16	59,407.31	0.63	0	0	0.27	0.73	0	0.40	0
10	1	b	55,924.30	54,215.39	57,709.85	1.62	0	0	0.05	0.95	0	0.14	0
10	1	c	53,724.07	52,186.04	55,231.45	3.48	1	0	0.00	1.00	1	0.00	0
10	1	d	53,083.35	51,619.26	54,660.67	3.97	1	0	0.00	1.00	1	0.00	0
100	0.5	a	54,580.73	52,678.24	58,340.47	2.11	1	0	0.02	0.98	1	0.00	1
100	0.5	b	55,498.03	53,617.82	57,578.78	1.81	1	0	0.04	0.96	1	0.06	0
100	0.5	c	56,478.41	54,494.86	59,105.59	1.03	0	0	0.15	0.85	0	0.13	0
100	0.5	d	54,142.44	52,639.18	55,756.76	3.13	1	0	0.00	1.00	1	0.00	0
100	1	a	56,404.90	54,379.51	59,137.07	1.08	0	0	0.14	0.86	0	0.14	0
100	1	b	58,472.02	56,148.53	61,744.61	-0.24	0	0	0.60	0.40	0	0.56	1
100	1	c	52,944.13	51,432.91	54,518.54	4.07	1	0	0.00	1.00	1	0.00	0
100	1	d	52,686.71	51,084.22	54,386.88	4.16	1	0	0.00	1.00	1	0.00	0
β coding: 'a' for $\beta = 0.01$, 'b' for $\beta = 0.1$, 'c' for $\beta = 1$, and 'd' for $\beta = 10$													

the alternative hypothesis H_{a^-} column indicates that the mean of the control is significantly lower than the mean of the treatment. The two columns under label "p value" show the p values of the test, given the alternative hypothesis H_{a^+} and H_{a^-}. The two-sided Wilcoxon Rank Sum test results with corresponding p values are shown in columns "H_*" and "p val" located under the label "Rank sum". The Wilcoxon Rank Sum test is calculated at a 0.10 significance level. Similarly, an entry of 1 in column H_* indicates the median of the control is significantly different than the median of the treatment. An entry of 0 indicates that the medians are not significantly different. Normality of the data is tested at a 0.05 significance level with both χ^2 goodness-of-fit and Lilliefors tests. The normality test result is shown as 1 if either the χ^2 goodness-of-fit or Lilliefors test can reject a normality hypothesis. Otherwise it is shown as 0. The section heading indicates an aggregate interval, a method used as the control, the control's sample mean, confidence interval in parentheses and normality test result.

In addition to the significance tests between the control and treatments, cross comparisons of significance test results of various treatments are provided in Table 4.8, Table 4.9, Table 4.10, Table 4.11 and Table 4.12 for Look-Ahead, Sarsa, Sarsa(λ), Residual Gradient and Direct Credit Back data, respectively.

Table 4.8: Cross significance tests: Look-Ahead

treatment	sample mean	H1	H2	H3	H4	H5
Period 1-12						
H1	27,789.58	0	0	0	0	0
H2	27,858.82	0	0	0	-1	0
H3	27,363.45	0	0	0	0	0
H4	27,032.24	0	1	0	0	0
H5	27,350.67	0	0	0	0	0
Period 13-60						
H1	58,060.49	0	0	0	-1	-1
H2	57,182.68	0	0	0	0	-1
H3	56,605.75	0	0	0	0	-1
H4	55,562.06	1	0	0	0	0
H5	54,266.68	1	1	1	0	0

The details of test statistics are omitted in these tables, showing test conclusions only. An entry of '0' means that average aggregate costs of row and column treatments are not significantly different. An entry of '1' means that an average aggregate cost of a row treatment is significantly lower than an average aggregate cost of a column treatment. An entry of '-1' means that average aggregate cost of a row treatment is significantly higher than an average aggregate cost of a column treatment.

Table 4.9: Cross significance tests: Sarsa

treatment	sample mean	β=0.01	β=0.1	β=1	β=10
Period 1-12					
β=0.01	35,407.23	0	0	0	0
β=0.1	34,632.42	0	0	0	0
β=1	35,542.00	0	0	0	0
β=10	34,981.11	0	0	0	0
Period 13-60					
β=0.01	53,143.27	0	0	0	0
β=0.1	51,535.10	0	0	0	1
β=1	51,213.10	0	0	0	1
β=10	54,160.33	0	-1	-1	0

Table 4.10: Cross significance tests: Sarsa(λ)

treatment		sample	$\lambda = 0$				$\lambda = 0.5$				$\lambda = 1$			
λ	β	mean	a	b	c	d	a	b	c	d	a	b	c	d
Period 1-12														
0	0.01	35,407.23	0	0	0	0	0	0	0	0	0	0	0	0
0	0.1	34,632.42	0	0	0	0	0	1	0	0	0	1	1	1
0	1	35,542.00	0	0	0	0	0	0	0	0	0	0	0	0
0	10	34,981.11	0	0	0	0	0	1	0	0	0	0	0	1
0.5	0.01	35,063.91	0	0	0	0	0	1	0	0	0	0	0	1
0.5	0.1	36,297.45	0	-1	0	-1	-1	0	-1	-1	-1	0	0	0
0.5	1	34,948.13	0	0	0	0	0	1	0	0	0	0	0	1
0.5	10	35,006.06	0	0	0	0	0	1	0	0	0	0	0	1
1	0.01	34,964.63	0	0	0	0	0	1	0	0	0	1	1	1
1	0.1	35,913.01	0	-1	0	0	0	0	0	0	-1	0	0	0
1	1	35,936.48	0	-1	0	0	0	0	0	0	-1	0	0	0
1	10	36,247.73	0	-1	0	-1	-1	0	-1	-1	-1	0	0	0
Period 13-60														
0	0.01	53,143.27	0	0	0	0	0	0	0	0	0	0	0	0
0	0.1	51,535.10	0	0	0	1	1	0	0	1	1	1	1	0
0	1	51,213.10	0	0	0	1	1	1	0	1	1	1	1	0
0	10	54,160.33	0	-1	-1	0	0	0	0	0	0	0	0	0
0.5	0.01	54,491.72	0	-1	-1	0	0	0	0	0	0	0	0	-1
0.5	0.1	53,875.38	0	0	-1	0	0	0	0	0	0	0	0	0
0.5	1	52,816.64	0	0	0	0	0	0	0	0	1	1	0	0
0.5	10	53,929.55	0	-1	-1	0	0	0	0	0	0	0	0	0
1	0.01	54,732.18	0	-1	-1	0	0	0	-1	0	0	0	0	-1
1	0.1	54,825.50	0	-1	-1	0	0	0	-1	0	0	0	0	-1
1	1	54,274.57	0	-1	-1	0	0	0	0	0	0	0	0	-1
1	10	52,163.76	0	0	0	0	1	0	0	0	1	1	1	0

Table 4.11: Cross significance tests: Residual Gradient

treatment	sample mean	β=0.01	β=0.1	β=1	β=10
Period 1-12					
β=0.01	33,994.76	0	0	0	0
β=0.1	33,755.77	0	0	0	0
β=1	34,940.99	0	0	0	0
β=10	34,014.70	0	0	0	0
Period 13-60					
β=0.01	55,587.89	0	0	0	0
β=0.1	54,700.96	0	0	0	0
β=1	57,498.39	0	0	0	0
β=10	56,820.45	0	0	0	0

Table 4.12: Cross significance tests: Direct Credit Back

	sample mean	$N_{cb}=0$				$N_{cb}=1$								$N_{cb}=10$								$N_{cb}=100$							
						$\lambda=0.5$				$\lambda=1$				$\lambda=0.5$				$\lambda=1$				$\lambda=0.5$				$\lambda=1$			
		a	b	c	d	a	b	c	d	a	b	c	d	a	b	c	d	a	b	c	d	a	b	c	d	a	b	c	d
Period 1-12																													
0a	33,994.76	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0	1	1	0	0	0	1	0	0	1	1
0b	33,755.77	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0	1	1	0	0	0	1	0	0	1	1
0c	34,940.99	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0	1	1	0	0	0	1	0	0	1	1
0d	34,014.70	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0	1	1	0	0	0	1	0	0	1	1
Aa	34,662.19	0	0	0	0	0	0	0	1	0	0	-1	1	0	0	0	1	0	0	1	1	0	0	0	1	0	0	1	1
Ab	34,638.65	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0	1	1	0	0	0	1	0	0	1	1
Ac	34,445.41	0	0	0	0	0	0	0	1	0	0	-1	1	0	0	0	1	0	0	1	1	0	0	0	1	0	0	1	1
Ad	36,779.08	-1	-1	-1	-1	-1	-1	-1	0	-1	-1	-1	0	-1	-1	-1	0	-1	-1	0	0	-1	-1	-1	0	-1	-1	0	0
Ba	35,042.44	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0	1	1	0	0	0	1	0	0	1	1
Bb	34,772.67	0	0	0	0	0	0	0	1	0	0	-1	1	0	0	0	1	0	0	1	1	0	0	0	1	0	0	1	1
Bc	33,251.07	0	0	0	0	1	0	1	1	0	1	0	1	0	1	0	1	0	0	1	1	0	0	1	1	0	1	1	1
Bd	36,206.79	-1	-1	-1	-1	-1	-1	-1	0	-1	-1	-1	0	-1	-1	-1	1	-1	-1	0	0	-1	-1	-1	0	-1	-1	0	0
Ca	34,696.94	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0	1	1	0	0	0	1	0	0	1	1
Cb	34,401.69	0	0	0	0	0	0	0	1	0	0	-1	1	0	0	0	1	0	0	1	1	0	0	0	1	0	0	1	1
Cc	35,840.59	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0	1	1	0	0	0	1	0	0	1	1
Cd	37,007.59	-1	-1	-1	-1	-1	-1	-1	0	-1	-1	-1	-1	-1	-1	-1	0	-1	-1	0	0	-1	-1	-1	0	-1	-1	0	-1
Da	33,894.88	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0	1	1	0	0	0	1	0	1	1	1
Db	33,297.36	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0	1	1	0	0	0	1	0	1	1	1
Dc	37,686.39	-1	-1	-1	-1	-1	-1	-1	0	-1	-1	-1	0	-1	-1	-1	0	-1	-1	0	0	-1	-1	-1	0	-1	-1	0	0
Dd	36,607.43	-1	-1	-1	-1	-1	-1	-1	0	-1	-1	-1	0	-1	-1	-1	0	-1	-1	0	0	-1	-1	-1	0	-1	-1	0	0
Ea	33,674.03	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0	1	1	0	0	0	1	0	1	1	1
Eb	34,173.42	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0	1	1	0	0	0	1	0	0	1	1
Ec	34,983.94	0	0	0	0	0	0	0	1	0	0	-1	1	0	0	0	1	0	0	1	1	0	0	0	1	0	0	1	1
Ed	36,284.28	-1	-1	-1	-1	-1	-1	-1	0	-1	-1	-1	0	-1	-1	-1	0	-1	-1	0	0	-1	-1	-1	0	-1	-1	0	0
Fa	33,911.69	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0	1	1	0	0	0	1	0	0	1	1
Fb	35,397.40	0	0	0	0	0	0	0	1	0	0	-1	1	0	0	0	1	-1	-1	1	1	-1	0	0	1	0	0	1	0
Fc	37,057.67	-1	-1	-1	-1	-1	-1	-1	0	-1	-1	-1	0	-1	-1	-1	0	-1	-1	0	0	-1	-1	-1	0	-1	-1	0	0
Fd	35,880.47	-1	-1	-1	-1	-1	-1	-1	0	-1	-1	-1	0	-1	-1	-1	1	-1	-1	0	0	-1	-1	-1	0	-1	0	0	0
Period 13-60																													
0a	55,587.89	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	0	0	0	0	0	0	-1	-1
0b	54,700.96	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	1	0	0
0c	57,498.39	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0d	56,820.45	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	-1
Aa	55,779.59	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	1	0	0
Ab	55,659.78	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	1	0	0
Ac	56,138.67	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	0	0	0	0	0	0	-1	-1
Ad	54,252.52	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	1	0	0
Ba	56,821.23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	0	0	0	0	0	0	-1	-1
Bb	55,233.58	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	-1	-1
Bc	55,931.36	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	-1	0	0	0	0	0	0	-1	-1
Bd	54,060.46	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0	1	0	0	1	0	0
Ca	55,946.79	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	-1
Cb	56,797.08	0	0	0	0	0	0	0	0	0	0	0	-1	0	0	0	-1	0	0	-1	-1	-1	0	0	-1	0	0	-1	-1
Cc	58,607.22	0	-1	0	0	-1	-1	0	-1	0	0	0	-1	0	0	0	-1	0	0	-1	-1	-1	0	0	-1	0	0	-1	-1
Cd	53,940.14	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	1	0	0	0	0	1	0	1	1	0	0
Da	57,187.90	0	-1	0	0	-1	-1	0	-1	0	0	0	-1	0	0	0	-1	0	0	-1	-1	-1	0	0	-1	0	0	-1	-1
Db	55,924.30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	0	0	-1	-1	0	0	0	0	0	0	-1	-1
Dc	53,724.07	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1	0	1	1	0	0	0	0	1	0	1	1	0	0
Dd	53,083.35	1	0	0	0	0	0	1	0	1	0	1	0	0	1	1	0	1	1	0	0	0	1	1	0	1	1	0	0
Ea	54,580.73	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0	1	0	0
Eb	55,498.03	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	0	0	0	0	0	0	-1	-1
Ec	56,478.41	0	0	0	0	0	0	0	0	0	0	0	-1	0	0	0	-1	0	0	-1	-1	0	0	0	0	0	0	-1	-1
Ed	54,142.44	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0	1	0	0
Fa	56,404.90	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	0	0	-1	-1	0	0	0	0	0	0	-1	-1
Fb	58,472.02	0	-1	0	0	-1	-1	0	-1	0	-1	0	-1	0	0	0	-1	0	0	-1	-1	-1	0	0	-1	0	0	-1	-1
Fc	52,944.13	1	0	0	1	0	0	1	0	1	1	1	0	1	1	1	0	1	1	0	0	0	1	1	0	1	1	0	0
Fd	52,686.71	1	0	0	1	0	0	1	0	1	1	1	0	1	1	1	0	1	1	0	0	0	1	1	0	1	1	0	0

treatment coding 'Xx': the first letter encodes Direct Credit Back parameters and the second encodes a learning rate.
First letter '0': $N_{cb}=0$; 'A': $N_{cb}=1$, $\lambda=0.5$; 'B': $N_{cb}=1$, $\lambda=1$
'C': $N_{cb}=10$, $\lambda=0.5$; 'D': $N_{cb}=10$, $\lambda=1$; 'E': $N_{cb}=100$, $\lambda=0.5$; 'F': $N_{cb}=100$, $\lambda=1$
Second letter 'a' for $\beta=0.01$, 'b' for $\beta=0.1$, 'c' for $\beta=1$, and 'd' for $\beta=10$

Remark: in Period 13-60, the significance tests show that the result of treatment 0c is not significantly different from the treaments Fc and Fd and the treatment 0d has significantly higher costs than the treaments Fc and Fd, while the treatment 0d's sample mean is lower than the treatment 0c. This is because both data are skewed and fail normality tests. Therefore, the conclusions are drawn from Wilcoxson Ranksum tests that do not use sample means.

Plots of sample means and BCa confidence intervals are provided in Figures 4.1, 4.2, 4.3, 4.4 and 4.5 respectively. To avoid cluttering in the plots, Direct Credit Back parameters (N_{cb} and λ) are labeled with two-letter codes. The first letter indicates the direct credit back setting: '0' means $N_{cb} = 0$; 'A' means $N_{cb} = 1, \lambda = 0.5$; 'B' means $N_{cb} = 1, \lambda = 1$; 'C' means $N_{cb} = 10, \lambda = 0.5$; 'D' means $N_{cb} = 10, \lambda = 1$; 'E' means $N_{cb} = 100, \lambda = 0.5$; 'F' means $N_{cb} = 100, \lambda = 1$. The second letter indicates a learning rate: 'a' means $\beta = 0.01$; 'b' means $\beta = 0.1$; 'c' means $\beta = 1$; and 'd' means $\beta = 10$.

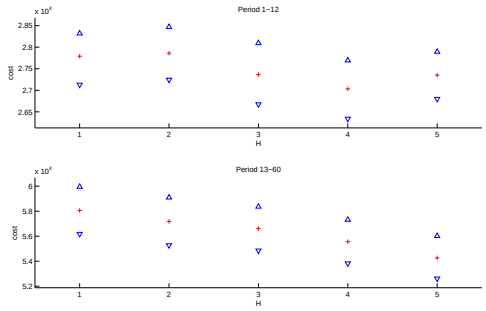

Figure 4.1: Average aggregate costs obtained from Look-Ahead on L0

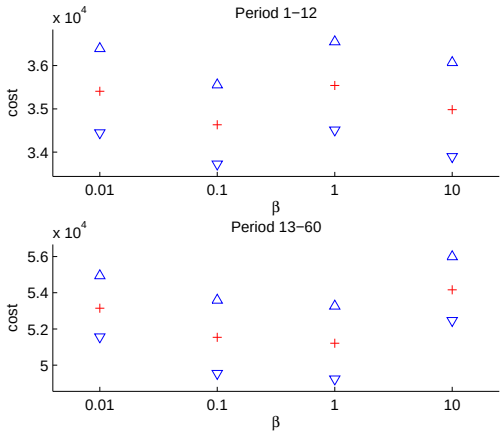

Figure 4.2: Average aggregate costs obtained from Sarsa on L0

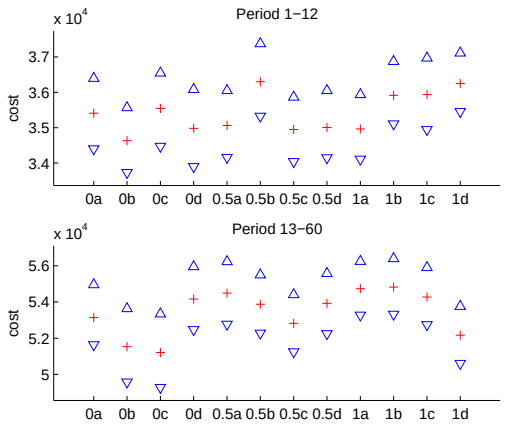

Figure 4.3: Average aggregate costs obtained from Sarsa(0), Sarsa(0.5), and Sarsa(1) on L0

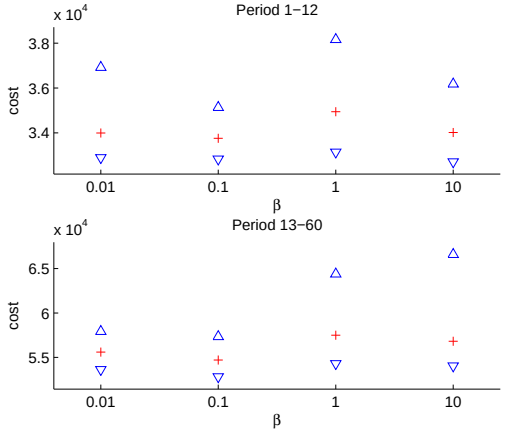

Figure 4.4: Average aggregate costs obtained from Residual Gradient on L0

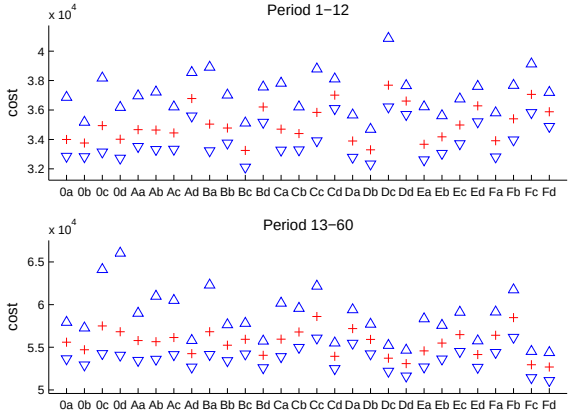

Figure 4.5: Average aggregate costs obtained from Direct Credit Back on L0

Comparison of Look-Ahead, Sarsa and Residual Gradient methods: Table 4.13 shows a cross comparison of Look-Ahead, Sarsa and Residual Gradient data.

Table 4.13: Cross comparison of different methods

	sample	Look-Ahead					Sarsa				Residual Gradient				Direct Credit Back			
	mean	H1	H2	H3	H4	H5	Sa	Sb	Sc	Sd	Ra	Rb	Rc	Rd	Ca	Cb	Cc	Cd
Period 1-12																		
H1	27,789.58	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1
H2	27,858.82	0	0	0	-1	0	1	1	1	1	1	1	1	1	1	1	1	1
H3	27,363.45	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1
H4	27,032.24	0	1	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1
H5	27,350.67	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1
Sa	35,407.23	-1	-1	-1	-1	-1	0	0	0	0	-1	-1	-1	-1	-1	0	0	0
Sb	34,632.42	-1	-1	-1	-1	-1	0	0	0	0	-1	-1	1	-1	-1	0	1	0
Sc	35,542.00	-1	-1	-1	-1	-1	0	0	0	0	-1	-1	-1	-1	-1	0	0	0
Sd	34,981.11	-1	-1	-1	-1	-1	0	0	0	0	-1	-1	-1	-1	-1	0	0	0
Ra	33,994.76	-1	-1	-1	-1	-1	1	1	1	1	0	0	0	0	0	0	1	1
Rb	33,755.77	-1	-1	-1	-1	-1	1	1	1	1	0	0	0	0	0	0	1	1
Rc	34,940.99	-1	-1	-1	-1	-1	1	-1	1	1	0	0	0	0	0	0	1	1
Rd	34,014.70	-1	-1	-1	-1	-1	1	1	1	1	0	0	0	0	0	0	1	1
Ca	33,911.69	-1	-1	-1	-1	-1	1	1	1	1	0	0	0	0	0	0	1	1
Cb	35,397.40	-1	-1	-1	-1	-1	0	0	0	0	0	0	0	0	0	0	1	0
Cc	37,057.67	-1	-1	-1	-1	-1	0	-1	0	0	-1	-1	-1	-1	-1	-1	0	0
Cd	35,880.47	-1	-1	-1	-1	-1	0	0	0	0	-1	-1	-1	-1	-1	0	0	0
Period 13-60																		
H1	58,060.49	0	0	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	0	-1	-1
H2	57,182.68	0	0	0	0	-1	-1	-1	-1	-1	0	-1	1	-1	0	0	-1	-1
H3	56,605.75	0	0	0	0	-1	-1	-1	-1	-1	0	0	0	0	0	0	-1	-1
H4	55,562.06	1	0	0	0	0	-1	-1	-1	0	0	0	0	0	0	0	-1	-1
H5	54,266.68	1	1	1	0	0	0	-1	-1	0	0	0	0	0	0	1	0	0
Sa	53,143.27	1	1	1	1	0	0	0	0	0	1	0	0	0	1	1	0	0
Sb	51,535.10	1	1	1	1	1	0	0	0	1	1	1	1	1	1	1	0	0
Sc	51,213.10	1	1	1	1	1	0	0	0	1	1	1	1	1	1	1	0	0
Sd	54,160.33	1	1	1	0	0	0	-1	-1	0	0	0	0	0	0	1	0	0
Ra	55,587.89	1	0	0	0	0	-1	-1	-1	0	0	0	0	0	0	0	-1	-1
Rb	54,700.96	1	1	0	0	0	0	-1	-1	0	0	0	0	0	0	1	0	0
Rc	57,498.39	1	-1	0	0	0	0	-1	-1	0	0	0	0	0	0	0	0	0
Rd	56,820.45	1	1	0	0	0	0	-1	-1	0	0	0	0	0	0	0	-1	-1
Ca	56,404.90	0	0	0	0	0	-1	-1	-1	0	0	0	0	0	0	0	-1	-1
Cb	58,472.02	0	0	0	0	-1	-1	-1	-1	-1	0	-1	0	0	0	0	-1	-1
Cc	52,944.13	1	1	1	1	0	0	0	0	0	1	0	0	1	1	1	0	0
Cd	52,686.71	1	1	1	1	0	0	0	0	0	1	0	0	1	1	1	0	0

Remark
'H': Look-Ahead, 'S': Sarsa, 'R': Residual Gradient, and 'C': Direct Credit Back ($N_{cb} = 100$, $\lambda = 1$)
'a' for $\beta = 0.01$, 'b' for $\beta = 0.1$, 'c' for $\beta = 1$, and 'd' for $\beta = 10$

Look-Ahead is labeled 'H' with the following number indicating the number of period(s) looking ahead. The Sarsa and Residual Gradient methods are labeled 'S' and 'RG' with the following lower-case letter indicating a learning rate: 'a' means $\beta = 0.01$; 'b' means $\beta = 0.1$; 'c' means $\beta = 1$; and 'd' means $\beta = 10$. Figure 4.6 shows averages and confidence intervals of the Look-Ahead, Sarsa and Residual Gradient data.

4.5 Discussions: a zero leadtime problem

Experimental results show that Sarsa is an effective method to control a zero leadtime inventory problem compared to other methods. As expected, using a Look-Ahead method with a longer look-

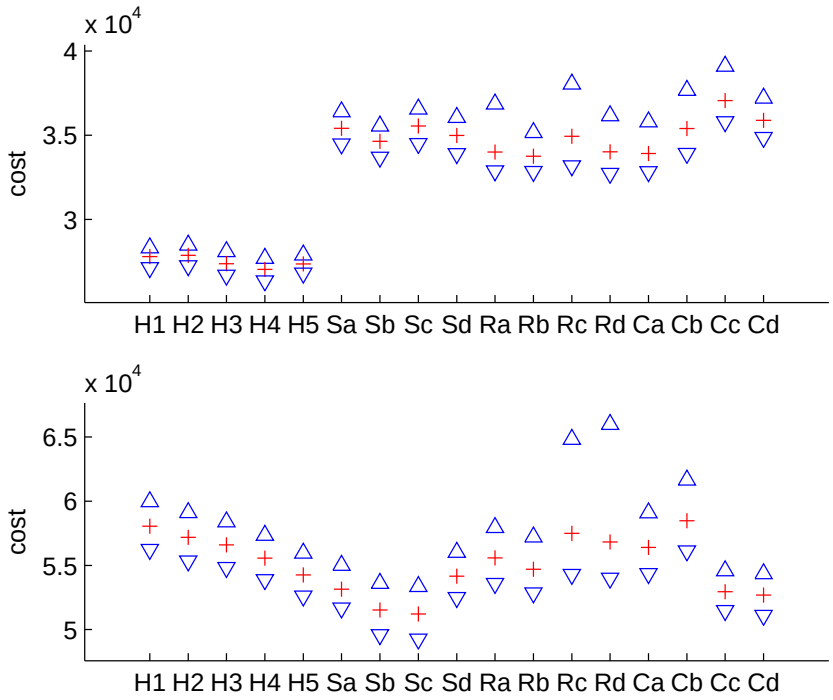

Figure 4.6: Average aggregate costs obtained from different methods on L0

ahead horizon results in lower aggregate costs than using it with a shorter horizon. However, the pairwise differences of aggregate costs of using one, two and three periods are not significant. Sarsa performs significantly better than a one-period Look-Ahead method in the later 48 periods. Learning rates of 0.1 and 1 produce lower aggregate costs than learning rates of 0.01 and 10. The significance test confirms that using Sarsa with learning rates of 0.1 and 1 results in better performance than using Sarsa with a learning rate of 10. Using an eligibility trace technique yields no improvement compared to Sarsa. On the other hand, Sarsa ($\lambda = 0$) yields lower aggregate costs than Sarsa($\lambda > 0$). The explanation for this behavior may lie in the nature of this problem, where a period cost is immediately observed, and the final outcome of an inventory problem is a summation of all period costs. Tesauro [116] successfully applied an eligibility trace technique to a backgammon game, but this situation is different from an inventory problem. While the final outcome of a zero leadtime inventory problem is a combination of all period costs, where each is observed instantly after taking an action, the final outcome of a Backgammon game is not known until the end. Therefore, bootstrapping a learning process with eligibility trace that improves learning speed in a delayed-reward problem such as Backgammon may not be suitable for a fast-return problem such as a zero leadtime inventory problem. In addition, Sutton and Barto [114] suggested that an eligibility trace technique should be

applied only to problems with long delayed reward and recommended TD(0) for problems with fast return.

The effect of eligibility trace on ADP performance is nevertheless worth further investigation. Sarsa, Sarsa(0.5) and Sarsa(1), each with two best performing parameters, are investigated further as shown in Figure 4.7.

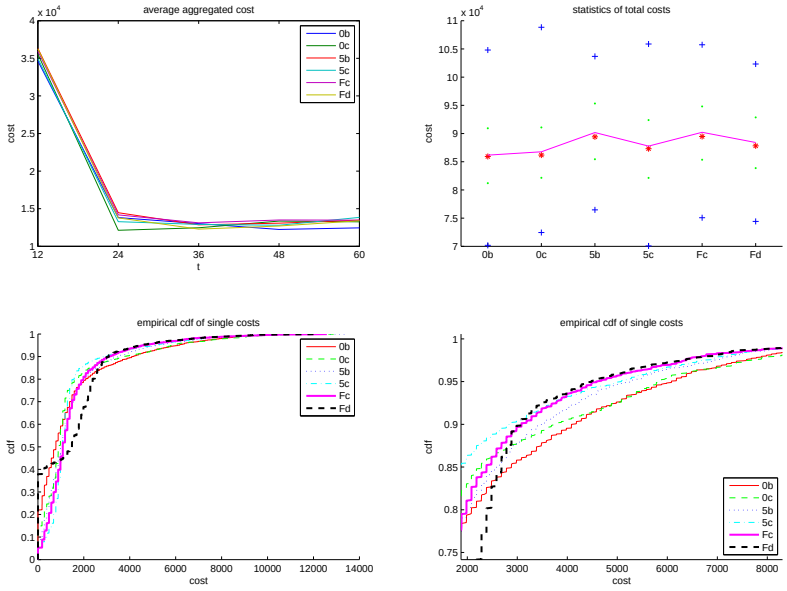

Figure 4.7: Results of Sarsa and Sarsa(λ); L0

Labels 0b, 0c, 5b, 5c, Fc and Fd represent Sarsa with $\beta = 0.1$, Sarsa with $\beta = 1$, Sarsa(0.5) with $\beta = 0.1$, Sarsa(0.5) with $\beta = 1$, Sarsa(1) with $\beta = 1$ and Sarsa(1) with $\beta = 10$, respectively. The upper left plot of Figure 4.7 shows 12-period aggregate costs. On this plot, the x axis tick marks indicate the last period of each 12-period aggregation. Sarsa with β of 0.1 and 1 deliver the lowest average aggregate costs in most aggregate periods, except Period 25-36. These results confirm the earlier conclusion that eligibility trace is not suitable for this type of problem. The upper right plot displays minimum, 1st Quartile, median, 3rd Quartile, and maximum of period costs with labels '+', '.', '*', '.' and '+' respectively. The average period costs are displayed in solid lines.

These statistics show that although Sarsa's mean and median period costs are lower than values of Sarsa(0.5) and Sarsa(1), the maximum period costs of Sarsa(0.5) and Sarsa(1) are lower than that of Sarsa. Since the maximum period cost implies the worst-case performance, this result may indicate that an eligibility trace may improve the worst-case performance of Sarsa. To examine this characteristic, CDF plots are also provided in figure 4.7. The left and right lower plots show an

empirical CDF[3] of period costs. The lower left plot displays the CDF for the complete range of period costs. The lower right plot displays the CDF for a selected range of period costs. These plots show that at high period costs the CDF values of Sarsa($\lambda > 0$) are higher than Sarsa. Since the greater CDF value at a greater period cost implies a better worst-case performance of Sarsa($\lambda > 0$), this may imply that Sarsa($\lambda > 0$) improves the worst-case performance of Sarsa.

To investigate the relation of an eligibility factor λ to inventory control performance Figure 4.8 shows plots of an average on-site inventory level, an average period cost and a maximum period cost versus an eligibility factor.

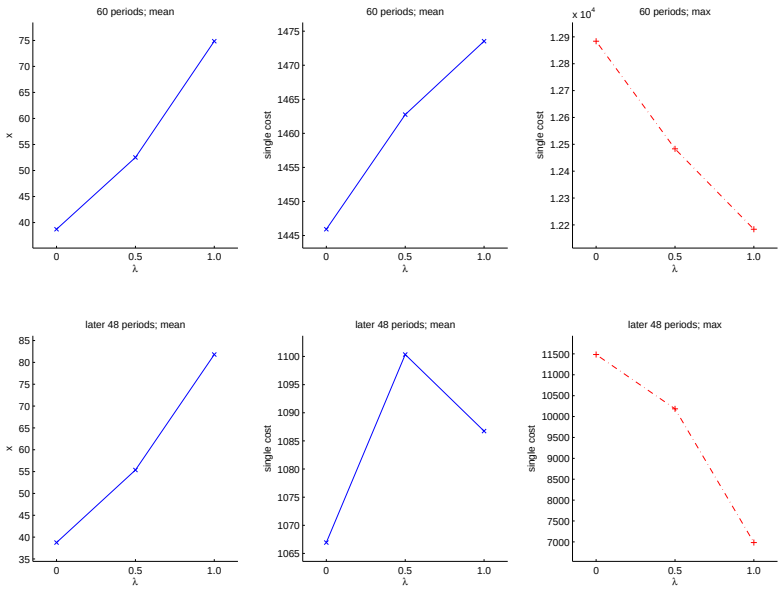

Figure 4.8: Inventory and period costs of Sarsa and Sarsa(λ); L0

The plots are drawn with the best performing learning rates (Sarsa with β of 1, Sarsa(0.5) with β of 1 and Sarsa(1) with β of 10). The upper plots show performance over 60 periods, and the lower plots show performance of the later 48 periods. Although average period costs may seem to be non-monotonic in the later 48 periods, with Sarsa showing the lowest average period costs, the upward trend of the average inventory levels and downward trend of the maximum period costs are apparent and may imply conservative behavior of Sarsa(λ). This conservative behavior is a result of a higher eligibility factor and a higher inventory level.

In addition to Sarsa and its extension Sarsa(λ), the Direct Credit Back equation was developed to improve Residual Gradient performance. Residual Gradient methods with learning rates of 0.01

[3] These CDF values are obtained from MATLAB's `ecdf` function, Revision 1.3.6.7, that uses a Kaplan-Meier estimate, as discussed in Kaplan and Meier [67].

and 0.1 show significantly better performance than a one-period Look-Ahead method in the later 48 periods. Using a Direct Credit Back method with the full credit back setting ($N_{cb} = 100$ and $\lambda = 1$) results in lower aggregate costs than the Residual Gradient method alone. Table 4.12 and Figure 4.5 show experimental results when using Residual Gradient (labeled 0a, 0b, 0c and 0d) and Direct Credit Back (labeled AA to Fd). Table 4.12 suggests that full Direct Credit Back improved Residual Gradient performance, but significance tests could not confirm the difference.

Comparison between Sarsa and Residual Gradient: As might be expected, Table 4.13 and Figure 4.6 show that the performance of a Look-Ahead method is better than that of the learning-based methods in the first 12 periods. This is because the learning-based methods require initial time to learn the process. In the later 48 periods, Sarsa, with learning rates of 0.1 and 1.0, performs significantly better than the Look-Ahead and Residual gradient methods. The full Direct Credit Back methods with learning rates of 1 and 10 improve Residual Gradient performance, as indicated in lower aggregate costs. The performances of the full Direct Credit Back methods are not significantly different from the performances of Sarsa.

4.6 Experiments: one-period leadtime problem

This section addresses the application of Sarsa, Sarsa(λ), Residual Gradient, Direct Credit Back, an (s,S) policy and a Roll-out method to an inventory problem with a one-period leadtime. Having a non-zero leadtime requires the inclusion of in-transit inventory. The state s_t is three-dimensional and is composed of previous demand D_{t-1}, on-site inventory level x_t and in-transit inventory $B_1^{(t)} \in \{0, 1, 2, ...\}$. A replenishment leadtime is set to one, $L = 1$ with other problem settings as mentioned in Section 4.3.

Look-Ahead: The Look-Ahead experiment is based on an average projection, using one to five future periods.

(s,S) policy: An (s,S) inventory management policy is based on a decision rule of a reordering point s and an order-up-to-level S. Section 2.1.2 discussed the (s,S) policy in detail.

The (s,S) policy investigated here has its parameters re-determined in each period after new demand is observed. Both a reordering level s and an order-up-to-level S are initially set to 50, which is also the initial value of previous demand. This is equivalent to a decision to fill an inventory level up to 50 units. This set up provides initial values for both parameters. The parameters s and S are revised each period using the Economic Order Quantity model (EOQ). The EOQ model requires

a demand rate, so for simplicity, the most recently observed demand will be treated as a demand rate in unit(s) per period. Therefore, after demand is observed for each period, a reordering level s is re-determined by Equations 2.3 and 2.4 and an order-upto-level S is re-determined by Equation 2.2. The factor Z and standard deviation σ in Equation 2.4 are set to 3 and $\sqrt{2}$, respectively. The inclusive inventory Y, which is a combination of both on-site and in-transit inventories, is used as the inventory level to determine a replenishment order.

Although the EOQ model may be somewhat inaccurate in determining (s,S) parameters, it may be used as a base policy in the Rollout method. Despite simple application of EOQ to a stochastic case, the EOQ model is easy to implement and it does not require much computation. This makes an EOQ-parameterized (s,S) policy suitable for using as a base policy of the Rollout method.

Sarsa: Sarsa is used with RBF as its state-action cost approximation. A structure of RBF has centers at each combination of {0, 10, 20, ..., 60} × {-30, -20, -10, ..., 60} × {0, 10, 20, ..., 60} × {0, 10, 20, ..., 150}, corresponding to four state-action variables (previous demand D_{t-1}, on-site inventory level x_t, in-transit inventory $B_1^{(t)}$ and after-replenishment on-site inventory level y_t). All RBF scales are 0.0139. Learning rates are 0.01, 0.1, 1, and 10.

Sarsa(λ): Eligibility factors of 0.5 and 1 are investigated. Since Sarsa (without eligibility trace) is equivalent to Sarsa with an eligibility factor of 0, experimental results of Sarsa will be labeled as Sarsa(0). Other parameters are set as in the above Sarsa experiments.

Residual Gradient: Residual Gradient with learning rates of 0.01, 0.1, 1 and 10 are investigated with RBF being used as a state-action cost approximation function. RBF parameters are set as in the above Sarsa experiments.

Direct Credit Back: Direct Credit Back is used with the credit back horizon N_{cb} = 1, 10 and 100, and credit back weight λ = 0.5 and 1. Since Residual Gradient is equivalent to Direct Credit Back with N_{cb} = 0, experimental results of Residual Gradient will be labeled as N_{cb} = 0. Other parameters are set as in the Residual Gradient experiments.

Rollout: An EOQ-parameterized (s,S) policy is used as a base policy for Rollout because an EOQ-parameterized (s,S) policy is easy to calculate. Since Rollout generally runs many simulations, an easy computation of the Rollout base policy is favorable in term of time expended in the simulation. The (s,S) parameters are initialized and redetermined as mentioned in the above (s,S) policy experiments. The Rollout simulation horizon T of 12 periods is used. The numbers of simulations N = 1, 10 and 100 are used.

4.7 Experimental results: one-period leadtime problem

As in the zero leadtime problem, conclusions are mainly based on significance tests and are presented in the same way. Significance tests of the Look-Ahead and (s,S) policy approaches are shown in Table 4.14. Since one-period Look-Ahead does not work well in the one-period-leadtime problem, a two-period Look-Ahead method, H2, is used as the control in other tables. Tables 4.15, 4.16, 4.17, 4.18 and 4.19 show significance test results of Sarsa, Sarsa(λ), Residual Gradient, Direct Credit Back and Rollout, respectively.

Table 4.14: Significance tests: Look-Ahead and (s,S) policies on one-period leadtime case

	sample	BCa interval		test	reject H_0		p value		Rank sum		Normal
	mean	LCI	UCI	stat.	H_{a+}	H_{a-}	H_{a+}	H_{a-}	H_*	p val.	
Period 1-12; H1 (sample mean = 72,673.60; CI = [71,040.80 ; 74,291.60]; normal test is passed.											
H1	72,673.60	71,086.77	74,360.00	0.00	0	0	0.50	0.50	0	1.00	0
H2	35,630.78	34,649.16	36,689.66	37.29	1	0	0.00	1.00	1	0.00	0
H3	34,762.85	33,965.52	35,625.46	39.90	1	0	0.00	1.00	1	0.00	0
H4	34,385.86	33,634.27	35,192.61	40.94	1	0	0.00	1.00	1	0.00	0
H5	34,645.59	33,872.19	35,460.49	40.51	1	0	0.00	1.00	1	0.00	0
(s,S)	61,463.21	60,836.22	62,045.12	12.48	1	0	0.00	1.00	1	0.00	0
Period 13-60; H1 (sample mean = 142,765.60; CI = [138,018.71 ; 147,670.80]; normal test is passed.											
H1	142,765.60	138,058.14	147,677.97	0.00	0	0	0.50	0.50	0	1.00	0
H2	63,706.82	61,421.19	66,138.16	28.82	1	0	0.00	1.00	1	0.00	0
H3	59,672.99	57,620.37	61,777.54	30.99	1	0	0.00	1.00	1	0.00	0
H4	56,975.87	55,048.89	59,110.36	32.04	1	0	0.00	1.00	1	0.00	0
H5	55,407.55	53,636.86	57,137.10	33.30	1	0	0.00	1.00	1	0.00	0
(s,S)	39,153.62	36,976.74	41,336.25	38.27	1	0	0.00	1.00	1	0.00	1
Period 1-60; H1 (sample mean = 215,439.20 ; CI = [210,115.78 ; 221,150.82]; normal test is passed.											
H1	215,439.20	210,162.22	220,968.06	0.00	0	0	0.50	0.50	0	1.00	0
H2	99,337.60	96,648.06	102,246.70	36.67	1	0	0.00	1.00	1	0.00	0
H3	94,435.84	92,123.61	96,992.73	39.11	1	0	0.00	1.00	1	0.00	0
H4	91,361.73	89,101.71	93,818.95	40.39	1	0	0.00	1.00	1	0.00	0
H5	90,053.13	88,116.54	92,264.27	41.55	1	0	0.00	1.00	1	0.00	0
(s,S)	100,616.83	98,272.78	103,166.20	37.14	1	0	0.00	1.00	1	0.00	1

Table 4.15: Significance tests: Sarsa on one-period leadtime case

	sample	BCa interval		test	reject H_0		p value		Rank sum		Normal
β	mean	LCI	UCI	stat.	H_{a+}	H_{a-}	H_{a+}	H_{a-}	H_*	p val.	
Period 1-12; H2 (sample mean = 35,630.78; CI = [34,667.94 ; 36,673.67]; normal test is passed.											
0.01	42,862.17	41,829.19	43,995.05	-9.46	0	1	1.00	0.00	1	0.00	0
0.1	42,886.75	41,844.51	44,049.53	-9.51	0	1	1.00	0.00	1	0.00	0
1	42,066.42	41,027.37	43,380.32	-8.06	0	1	1.00	0.00	1	0.00	1
10	43,000.04	42,092.51	44,131.26	-10.01	0	1	1.00	0.00	1	0.00	0
Period 13-60; H2 (sample mean = 63,706.82; CI = [61,457.47 ; 66,137.34]; normal test is passed.											
0.01	53,199.16	51,766.05	54,584.75	7.46	1	0	0.00	1.00	1	0.00	0
0.1	52,213.12	50,562.59	53,937.92	7.72	1	0	0.00	1.00	1	0.00	0
1	51,757.42	50,150.16	53,557.94	8.06	1	0	0.00	1.00	1	0.00	0
10	53,691.24	52,058.63	55,366.00	6.77	1	0	0.00	1.00	1	0.00	0

Table 4.16: Significance tests: Sarsa(λ) on one-period leadtime case

		sample	BCa interval		test	reject H_0		p value		Rank sum		Normal
λ	β	mean	LCI	UCI	stat.	H_{a+}	H_{a-}	H_{a+}	H_{a-}	H_*	p val.	
Period 1-12; H2 (sample mean = 35,630.78; CI = [34,666.77 ; 36,710.18]; normal test is passed.												
0	a	42,862.17	41,897.61	44,053.98	-9.46	0	1	1.00	0.00	1	0.00	0
0	b	42,886.75	41,852.66	44,012.81	-9.51	0	1	1.00	0.00	1	0.00	0
0	c	42,066.42	40,998.00	43,318.64	-8.06	0	1	1.00	0.00	1	0.00	1
0	d	43,000.04	42,106.61	44,132.35	-10.01	0	1	1.00	0.00	1	0.00	0
0.5	a	42,674.55	41,765.45	43,835.03	-9.48	0	1	1.00	0.00	1	0.00	0
0.5	b	42,237.61	41,127.92	43,504.22	-8.24	0	1	1.00	0.00	1	0.00	1
0.5	c	41,717.66	40,667.80	42,900.75	-7.85	0	1	1.00	0.00	1	0.00	0
0.5	d	42,227.65	41,284.23	43,339.82	-8.92	0	1	1.00	0.00	1	0.00	1
1	a	42,513.96	41,590.70	43,548.28	-9.55	0	1	1.00	0.00	1	0.00	0
1	b	42,602.56	41,543.07	43,838.00	-8.81	0	1	1.00	0.00	1	0.00	0
1	c	42,729.48	41,849.28	43,671.88	-10.08	0	1	1.00	0.00	1	0.00	0
1	d	43,031.49	42,049.97	44,095.19	-9.89	0	1	1.00	0.00	1	0.00	0
Period 13-60; H2 (sample mean = 63,706.82; CI = [61,424.18 ; 66,136.73]; normal test is passed.												
0	a	53,199.16	51,797.36	54,621.32	7.46	1	0	0.00	1.00	1	0.00	0
0	b	52,213.12	50,586.74	53,977.45	7.72	1	0	0.00	1.00	1	0.00	0
0	c	51,757.42	50,185.24	53,519.84	8.06	1	0	0.00	1.00	1	0.00	0
0	d	53,691.24	52,029.20	55,375.97	6.77	1	0	0.00	1.00	1	0.00	0
0.5	a	53,768.84	52,242.59	55,282.39	6.91	1	0	0.00	1.00	1	0.00	0
0.5	b	53,860.34	52,353.84	55,498.64	6.75	1	0	0.00	1.00	1	0.00	1
0.5	c	53,211.24	51,889.25	54,676.29	7.48	1	0	0.00	1.00	1	0.00	0
0.5	d	54,006.68	52,307.92	55,653.21	6.52	1	0	0.00	1.00	1	0.00	0
1	a	54,561.77	52,838.48	56,297.46	6.07	1	0	0.00	1.00	1	0.00	0
1	b	54,394.90	52,886.28	55,993.07	6.40	1	0	0.00	1.00	1	0.00	0
1	c	53,864.51	52,274.98	55,569.68	6.68	1	0	0.00	1.00	1	0.00	0
1	d	52,898.46	51,281.67	54,564.44	7.32	1	0	0.00	1.00	1	0.00	0

Table 4.17: Significance tests: Residual Gradient on one-period leadtime case

	sample	BCa interval		test	reject H_0		p value		Rank sum		Normal
treatment	mean	LCI	UCI	stat.	H_{a+}	H_{a-}	H_{a+}	H_{a-}	H_*	p val.	
Period 1-12; H2 (sample mean = 35,630.78; CI = [34,656.50 ; 36,699.02]; normal test is passed.											
0.01	41,220.53	40,365.30	42,181.01	-7.94	0	1	1.00	0.00	1	0.00	0
0.1	41,728.49	40,944.44	42,880.84	-8.58	0	1	1.00	0.00	1	0.00	1
1	41,993.03	40,787.53	43,448.58	-7.40	0	1	1.00	0.00	1	0.00	1
10	42,734.09	41,740.03	44,127.82	-8.89	0	1	1.00	0.00	1	0.00	0
Period 13-60; H2 (sample mean = 63,706.82; CI = [61,354.24 ; 66,158.33]; normal test is passed.											
0.01	54,379.33	52,735.93	56,155.95	6.25	1	0	0.00	1.00	1	0.00	0
0.1	53,756.88	52,215.30	55,390.28	6.82	1	0	0.00	1.00	1	0.00	0
1	53,913.60	52,331.19	55,523.32	6.69	1	0	0.00	1.00	1	0.00	0
10	53,546.41	51,968.19	55,265.56	6.87	1	0	0.00	1.00	1	0.00	0

Table 4.18: Significance tests: Direct Credit Back on one-period leadtime case

treatment			sample	BCa interval		test	reject H_0		p value		Rank sum		Normal
N_{cb}	λ	β	mean	LCI	UCI	stat.	H_{a^+}	H_{a^-}	H_{a^+}	H_{a^-}	H_*	p val.	
Period 1-12; H2 (sample mean = 35,630.78; CI = [34,670.63 ; 36,728.55]; normal test is passed.													
0		a	41,220.53	40,353.40	42,173.06	-7.94	0	1	1.00	0.00	1	0.00	0
0		b	41,728.49	40,938.41	42,894.32	-8.58	0	1	1.00	0.00	1	0.00	1
0		c	41,993.03	40,773.89	43,465.71	-7.40	0	1	1.00	0.00	1	0.00	1
0		d	42,734.09	41,731.70	44,167.90	-8.89	0	1	1.00	0.00	1	0.00	0
1	0.5	a	41,496.50	40,564.26	42,459.08	-8.18	0	1	1.00	0.00	1	0.00	1
1	0.5	b	41,478.03	40,561.60	42,562.35	-7.94	0	1	1.00	0.00	1	0.00	0
1	0.5	c	42,119.94	41,340.61	42,854.63	-9.98	0	1	1.00	0.00	1	0.00	0
1	0.5	d	42,024.35	41,011.98	43,208.80	-8.26	0	1	1.00	0.00	1	0.00	1
1	1	a	42,380.75	41,341.73	43,530.92	-8.76	0	1	1.00	0.00	1	0.00	0
1	1	b	41,510.89	40,655.26	42,638.40	-8.12	0	1	1.00	0.00	1	0.00	0
1	1	c	41,242.89	40,405.05	42,650.34	-7.35	0	1	1.00	0.00	1	0.00	1
1	1	d	42,728.62	41,753.30	43,802.52	-9.56	0	1	1.00	0.00	1	0.00	0
10	0.5	a	41,617.49	40,731.59	42,562.86	-8.51	0	1	1.00	0.00	1	0.00	0
10	0.5	b	40,877.86	39,976.83	41,848.78	-7.41	0	1	1.00	0.00	1	0.00	0
10	0.5	c	41,952.28	41,068.39	42,772.25	-9.26	0	1	1.00	0.00	1	0.00	0
10	0.5	d	43,219.26	42,095.01	44,526.77	-9.27	0	1	1.00	0.00	1	0.00	1
10	1	a	41,711.66	40,823.11	42,602.52	-8.73	0	1	1.00	0.00	1	0.00	0
10	1	b	40,957.00	39,969.09	41,949.21	-7.28	0	1	1.00	0.00	1	0.00	0
10	1	c	43,012.52	41,877.94	44,702.65	-8.41	0	1	1.00	0.00	1	0.00	1
10	1	d	42,959.12	41,899.67	44,390.41	-9.03	0	1	1.00	0.00	1	0.00	0
100	0.5	a	42,065.56	40,969.59	43,554.89	-7.66	0	1	1.00	0.00	1	0.00	0
100	0.5	b	41,579.76	40,544.58	42,650.89	-7.81	0	1	1.00	0.00	1	0.00	0
100	0.5	c	41,730.99	40,943.56	42,801.60	-8.70	0	1	1.00	0.00	1	0.00	1
100	0.5	d	42,997.33	42,077.39	44,255.02	-9.64	0	1	1.00	0.00	1	0.00	0
100	1	a	42,114.12	41,090.35	43,137.61	-8.77	0	1	1.00	0.00	1	0.00	0
100	1	b	42,401.19	41,297.12	43,854.70	-8.14	0	1	1.00	0.00	1	0.00	1
100	1	c	43,728.34	42,532.09	45,178.26	-9.40	0	1	1.00	0.00	1	0.00	1
100	1	d	42,871.54	41,940.52	44,087.97	-9.59	0	1	1.00	0.00	1	0.00	1
Period 13-60; H2 (sample mean = 63,706.82; CI = [61,481.80 ; 66,204.13]; normal test is passed.													
0		a	54,379.33	52,738.62	56,150.66	6.25	1	0	0.00	1.00	1	0.00	0
0		b	53,756.88	52,210.66	55,409.56	6.82	1	0	0.00	1.00	1	0.00	0
0		c	53,913.60	52,342.37	55,597.10	6.69	1	0	0.00	1.00	1	0.00	0
0		d	53,546.41	51,959.35	55,245.20	6.87	1	0	0.00	1.00	1	0.00	0
1	0.5	a	54,180.98	52,519.50	56,103.78	6.25	1	0	0.00	1.00	1	0.00	0
1	0.5	b	54,390.28	52,734.67	56,086.30	6.28	1	0	0.00	1.00	1	0.00	1
1	0.5	c	53,113.71	51,421.07	54,781.96	7.12	1	0	0.00	1.00	1	0.00	0
1	0.5	d	53,889.64	52,259.42	55,618.04	6.60	1	0	0.00	1.00	1	0.00	0
1	1	a	53,393.71	51,803.45	55,129.41	6.99	1	0	0.00	1.00	1	0.00	0
1	1	b	54,064.22	52,429.70	55,797.30	6.44	1	0	0.00	1.00	1	0.00	0
1	1	c	53,767.45	52,204.41	55,421.15	6.74	1	0	0.00	1.00	1	0.00	0
1	1	d	53,851.08	52,284.81	55,627.02	6.61	1	0	0.00	1.00	1	0.00	0
10	0.5	a	54,501.87	52,666.41	56,798.84	5.77	1	0	0.00	1.00	1	0.00	1
10	0.5	b	55,380.61	53,580.04	57,292.49	5.40	1	0	0.00	1.00	1	0.00	0
10	0.5	c	54,181.77	52,621.73	55,785.89	6.54	1	0	0.00	1.00	1	0.00	0
10	0.5	d	53,159.79	51,686.64	54,695.83	7.31	1	0	0.00	1.00	1	0.00	0
10	1	a	53,429.03	51,597.32	55,417.66	6.59	1	0	0.00	1.00	1	0.00	0
10	1	b	54,210.30	52,599.84	55,890.55	6.44	1	0	0.00	1.00	1	0.00	0
10	1	c	53,751.83	52,278.16	55,236.14	6.96	1	0	0.00	1.00	1	0.00	1
10	1	d	52,360.43	50,793.10	53,983.73	7.75	1	0	0.00	1.00	1	0.00	0
100	0.5	a	54,225.79	52,784.91	55,703.31	6.64	1	0	0.00	1.00	1	0.00	0
100	0.5	b	54,787.97	53,237.64	56,441.04	6.11	1	0	0.00	1.00	1	0.00	0
100	0.5	c	53,776.81	52,318.75	55,299.78	6.94	1	0	0.00	1.00	1	0.00	0
100	0.5	d	53,540.84	51,970.52	55,218.08	6.93	1	0	0.00	1.00	1	0.00	0
100	1	a	52,741.89	51,182.94	54,411.64	7.50	1	0	0.00	1.00	1	0.00	0
100	1	b	54,899.91	53,290.65	56,646.51	5.93	1	0	0.00	1.00	1	0.00	0
100	1	c	52,600.06	51,044.17	54,198.59	7.64	1	0	0.00	1.00	1	0.00	0
100	1	d	52,926.75	51,293.73	54,601.60	7.34	1	0	0.00	1.00	1	0.00	0

Table 4.19: Significance tests: Rollout on one-period leadtime case

N_{sim}	sample mean	BCa interval LCI	BCa interval UCI	test stat.	reject H_0 H_{a+}	reject H_0 H_{a-}	p value H_{a+}	p value H_{a-}	Rank sum H_*	Rank sum p val.	Normal
Period 1-12; H2 (sample mean = 35,468.76; CI = [34,520.06 ; 36,500.66]; normal test is passed.											
1	41,459.08	40,853.92	42,082.70	-9.95	0	1	1.00	0.00	1	0.00	0
10	41,468.17	41,043.71	41,887.68	-10.73	0	1	1.00	0.00	1	0.00	0
100	38,659.19	38,001.00	39,321.34	-5.17	0	1	1.00	0.00	1	0.00	0
Period 13-60; H2 (sample mean = 64,207.21; CI = [61,871.35 ; 66,602.66]; normal test is passed.											
1	52,418.65	50,989.97	53,939.56	8.24	1	0	0.00	1.00	1	0.00	0
10	53,425.59	52,213.94	54,718.66	7.84	1	0	0.00	1.00	1	0.00	0
100	53,167.47	51,892.10	54,326.28	8.12	1	0	0.00	1.00	1	0.00	1
Period 1-60; H2 (sample mean = 99,675.97; CI = [96,993.50 ; 102,578.79]; normal test is passed.											
1	93,877.73	92,161.51	95,635.34	3.44	1	0	0.00	1.00	1	0.00	0
10	94,893.76	93,674.04	96,255.56	3.03	1	0	0.00	1.00	1	0.01	1
100	91,826.66	90,491.79	93,118.44	4.97	1	0	0.00	1.00	1	0.00	0

Figures 4.9, 4.10, 4.11, 4.12, 4.13 and 4.14 show mean and confidence intervals of aggregate costs obtained from Look-Ahead and (s,S) policy, Sarsa, Sarsa(λ), Residual Gradient, Direct Credit Back and Rollout, respectively.

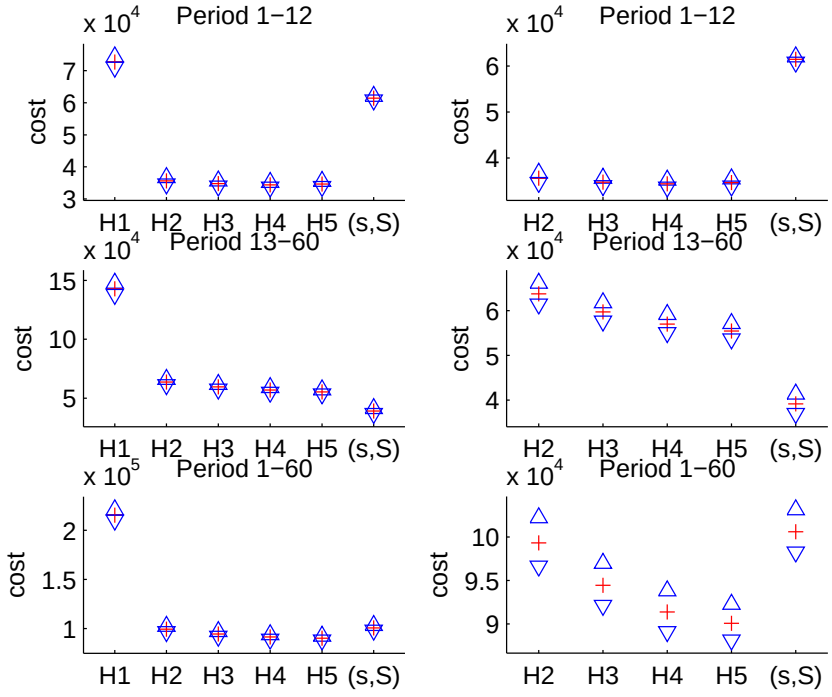

Figure 4.9: Average aggregate costs obtained from Look-Ahead and (s,S) on L1

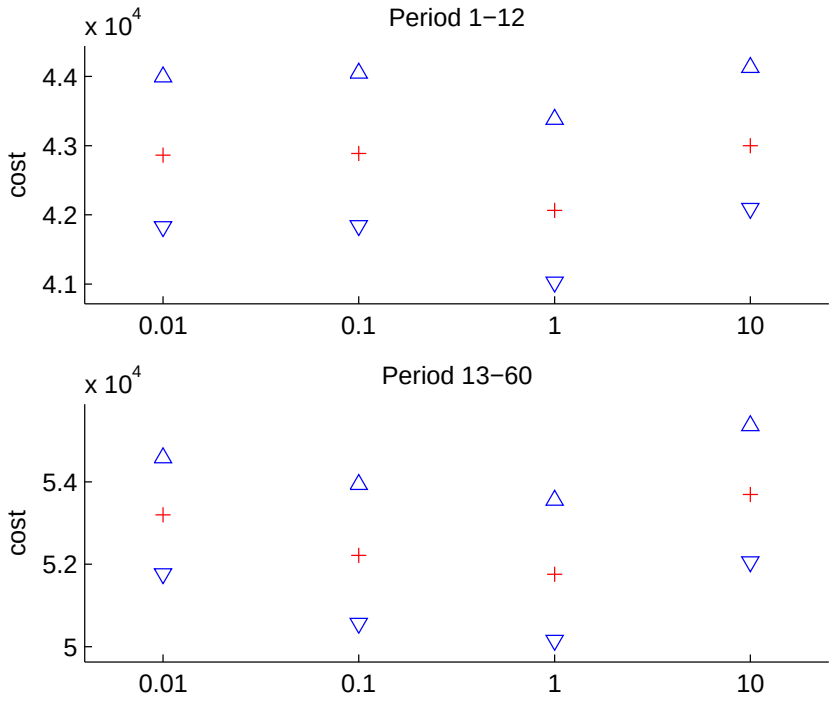

Figure 4.10: Average aggregate costs obtained from Sarsa on L1

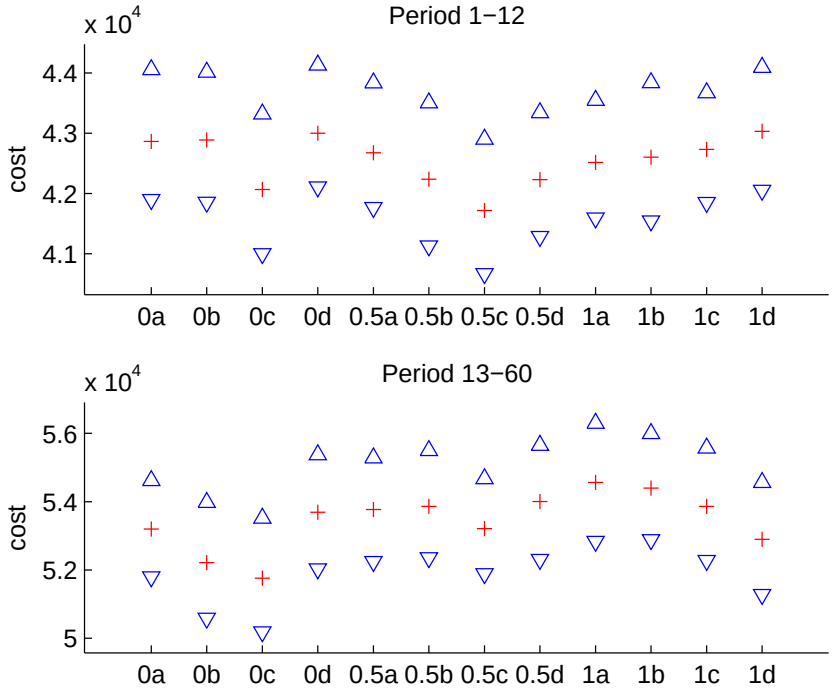

Figure 4.11: Average aggregate costs obtained from Sarsa(0), Sarsa(0.5), and Sarsa(1) on L1

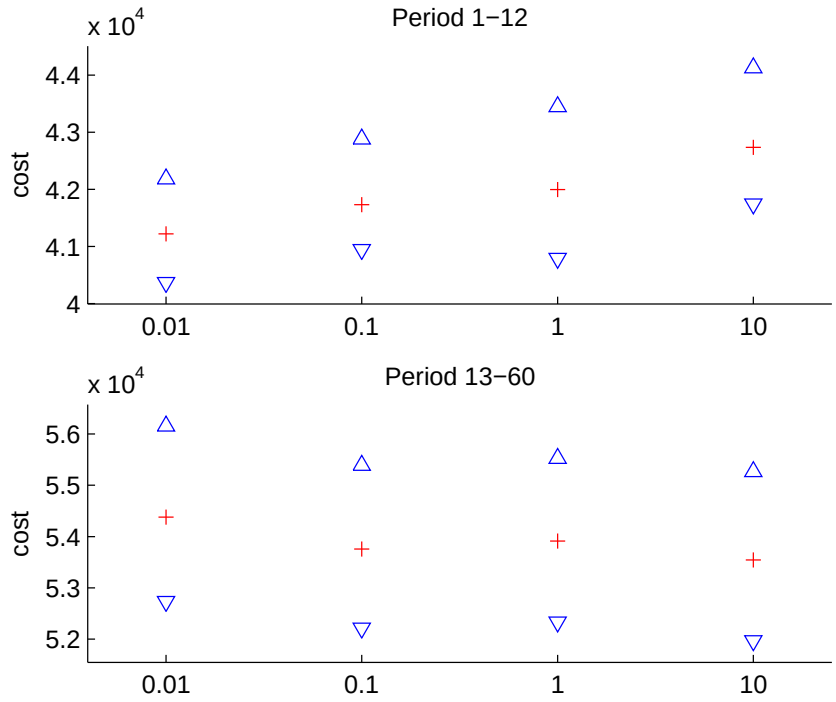

Figure 4.12: Average aggregate costs obtained from Residual Gradient on L1

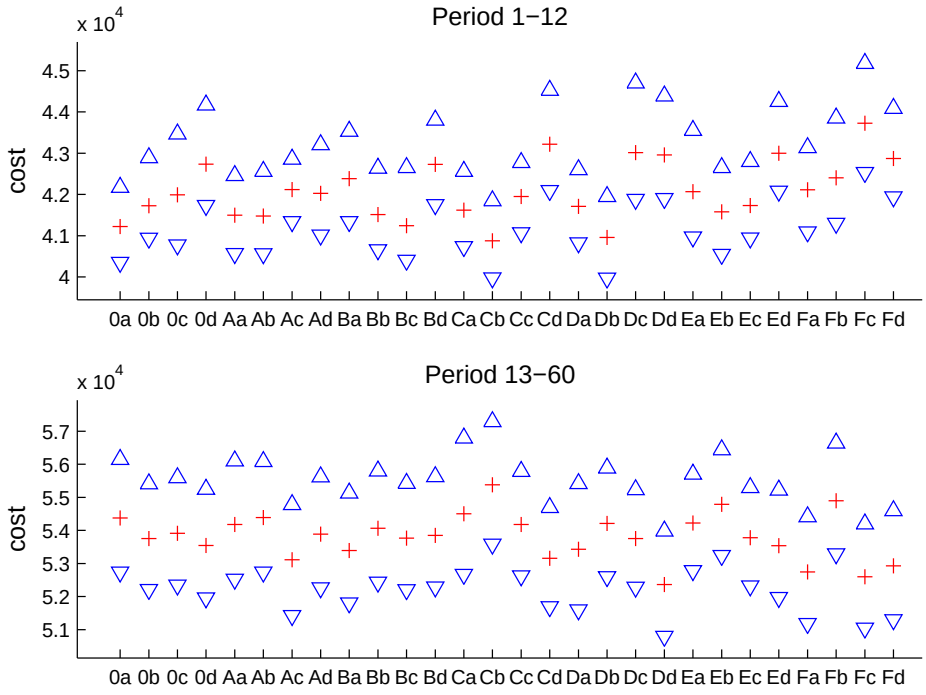

Figure 4.13: Average aggregate costs obtained from Direct Credit Back on L1

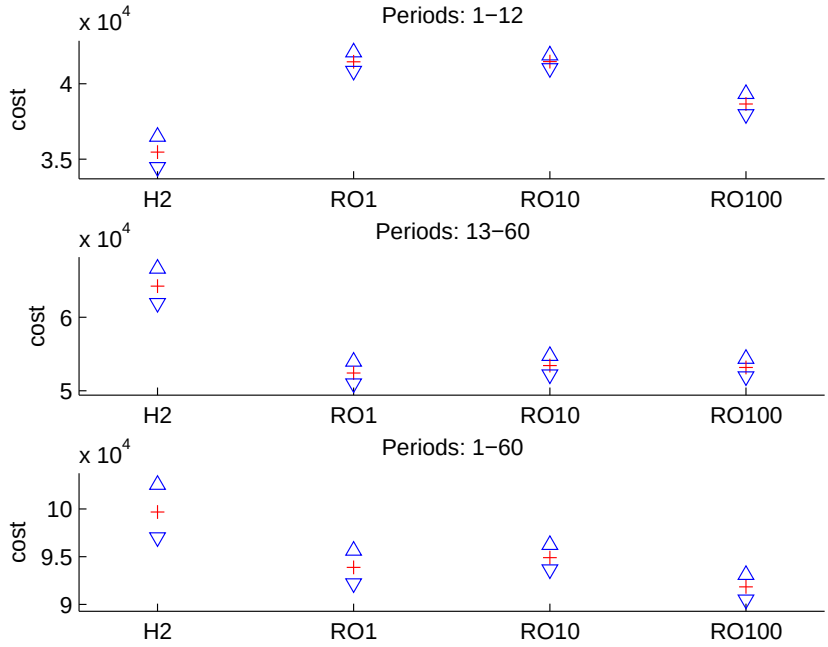

Figure 4.14: Average aggregate costs obtained from Rollout on L1

Tables 4.20, 4.21, 4.22, 4.23 and 4.24 show a summary of cross significance tests of a Look-Ahead method and an (s,S) policy, Sarsa, Residual Gradient, Sarsa(λ) and Direct Credit Back. Table 4.25 shows a summary of cross significance tests among different methods including Rollout.

Table 4.20: Cross significance tests: Look-Ahead and (s,S) on one-period leadtime case

treatment	sample mean	H1	H2	H3	H4	H5	(s,S)
Period 1-12							
H1	72,673.60	0	-1	-1	-1	-1	-1
H2	35,630.78	1	0	0	-1	0	1
H3	34,762.85	1	0	0	0	0	1
H4	34,385.86	1	1	0	0	0	1
H5	34,645.59	1	0	0	0	0	1
(s,S)	61,463.21	1	-1	-1	-1	-1	0
Period 13-60							
H1	142,765.60	0	-1	-1	-1	-1	-1
H2	63,706.82	1	0	-1	-1	-1	-1
H3	59,672.99	1	1	0	-1	-1	-1
H4	56,975.87	1	1	1	0	0	-1
H5	55,407.55	1	1	1	0	0	-1
(s,S)	39,153.62	1	1	1	1	1	0
All Periods							
H1	215,439.20	0	-1	-1	-1	-1	-1
H2	99,337.60	1	0	-1	-1	-1	0
H3	94,435.84	1	1	0	-1	-1	1
H4	91,361.73	1	1	1	0	0	1
H5	90,053.13	1	1	1	0	0	1
(s,S)	100,616.83	1	0	-1	-1	-1	0

Table 4.21: Cross significance tests: Sarsa on one-period leadtime case

treatment	sample mean	0.01	0.1	1	10
Period 1-12					
0.01	42,862.17	0	0	0	0
0.1	42,886.75	0	0	0	0
1	42,066.42	0	0	0	0
10	43,000.04	0	0	0	0
Period 13-60					
0.01	53,199.16	0	0	0	0
0.1	52,213.12	0	0	0	0
1	51,757.42	0	0	0	0
10	53,691.24	0	0	0	0
Period 1-60					
0.01	96,061.33	0	0	0	0
0.1	95,099.86	0	0	0	0
1	93,823.84	0	0	0	1
10	96,691.28	0	0	-1	0

Table 4.22: Cross significance tests: Residual Gradient one one-period leadtime case

treatment	sample mean	0.01	0.1	1	10
Period 1-12					
0.01	41,220.53	0	0	0	1
0.1	41,728.49	0	0	0	0
1	41,993.03	0	0	0	0
10	42,734.09	-1	0	0	0
Period 13-60					
0.01	54,379.33	0	0	0	0
0.1	53,756.88	0	0	0	0
1	53,913.60	0	0	0	0
10	53,546.41	0	0	0	0
Period 1-60					
0.01	95,599.85	0	0	0	0
0.1	95,485.37	0	0	0	0
1	95,906.63	0	0	0	0
10	96,280.50	0	0	0	0

Table 4.23: Cross significance tests: Sarsa(λ) on one-period leadtime case

treatment		sample	$\lambda = 0$				$\lambda = 0.5$				$\lambda = 1$			
λ	β	mean	a	b	c	d	a	b	c	d	a	b	c	d
Period 1-12														
0	a	42,862.17	0	0	0	0	0	0	0	0	0	0	0	0
0	b	42,886.75	0	0	0	0	0	0	0	0	0	0	0	0
0	c	42,066.42	0	0	0	0	0	0	0	0	0	0	0	0
0	d	43,000.04	0	0	0	0	0	0	-1	0	0	0	0	0
0.5	a	42,674.55	0	0	0	0	0	0	0	0	0	0	0	0
0.5	b	42,237.61	0	0	0	0	0	0	0	0	0	0	0	0
0.5	c	41,717.66	0	0	0	1	0	0	0	0	0	0	0	1
0.5	d	42,227.65	0	0	0	0	0	0	0	0	0	0	0	0
1	a	42,513.96	0	0	0	0	0	0	0	0	0	0	0	0
1	b	42,602.56	0	0	0	0	0	0	0	0	0	0	0	0
1	c	42,729.48	0	0	0	0	0	0	0	0	0	0	0	0
1	d	43,031.49	0	0	0	0	0	0	-1	0	0	0	0	0
Period 13-60														
0	a	53,199.16	0	0	0	0	0	0	0	0	0	0	0	0
0	b	52,213.12	0	0	0	0	0	0	0	0	1	1	0	0
0	c	51,757.42	0	0	0	0	1	1	0	1	1	1	1	0
0	d	53,691.24	0	0	0	0	0	0	0	0	0	0	0	0
0.5	a	53,768.84	0	0	-1	0	0	0	0	0	0	0	0	0
0.5	b	53,860.34	0	0	-1	0	0	0	0	0	0	0	0	0
0.5	c	53,211.24	0	0	0	0	0	0	0	0	0	0	0	0
0.5	d	54,006.68	0	0	-1	0	0	0	0	0	0	0	0	0
1	a	54,561.77	0	-1	-1	0	0	0	0	0	0	0	0	0
1	b	54,394.90	0	-1	-1	0	0	0	0	0	0	0	0	0
1	c	53,864.51	0	0	-1	0	0	0	0	0	0	0	0	0
1	d	52,898.46	0	0	0	0	0	0	0	0	0	0	0	0
Period 1-60														
0	a	96,061.33	0	0	0	0	0	0	0	0	0	0	0	0
0	b	95,099.86	0	0	0	0	0	0	0	0	0	0	0	0
0	c	93,823.84	0	0	0	1	1	0	0	0	1	1	1	0
0	d	96,691.28	0	0	-1	0	0	0	0	0	0	0	0	0
0.5	a	96,443.40	0	0	-1	0	0	0	0	0	0	0	0	0
0.5	b	96,097.95	0	0	0	0	0	0	0	0	0	0	0	0
0.5	c	94,928.90	0	0	0	0	0	0	0	0	0	0	0	0
0.5	d	96,234.33	0	0	0	0	0	0	0	0	0	0	0	0
1	a	97,075.73	0	0	-1	0	0	0	0	0	0	0	0	0
1	b	96,997.46	0	0	-1	0	0	0	0	0	0	0	0	0
1	c	96,593.99	0	0	-1	0	0	0	0	0	0	0	0	0
1	d	95,929.94	0	0	0	0	0	0	0	0	0	0	0	0
Remark β coding: 'a' for $\beta = 0.01$, 'b' for $\beta = 0.1$ 'c' for $\beta = 1$, and 'd' for $\beta = 10$														

Table 4.24: Cross significance tests: Direct Credit Back on one-period leadtime case

	sample	$N_{CB}=0$				$N_{CB}=1$								$N_{CB}=10$								$N_{CB}=100$							
						$\lambda=0.5$				$\lambda=1$				$\lambda=0.5$				$\lambda=1$				$\lambda=0.5$				$\lambda=1$			
	mean	0a	0b	0c	0d	Aa	Ab	Ac	Ad	Ba	Bb	Bc	Bd	Ca	Cb	Cc	Cd	Da	Db	Dc	Dd	Ea	Eb	Ec	Ed	Fa	Fb	Fc	Fd
Period 1-12																													
0a	41,221	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	1	0	0	1	1	0	0	0	1	0	0	1	1
0b	41,728	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0
0c	41,993	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
0d	42,734	-1	0	0	0	0	0	0	0	0	0	-1	0	0	-1	0	0	0	-1	0	0	0	0	0	0	0	0	0	0
Aa	41,497	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
Ab	41,478	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	1	1
Ac	42,120	0	0	0	0	0	0	0	0	0	0	-1	0	0	-1	0	0	0	-1	0	0	0	0	0	0	0	0	0	0
Ad	42,024	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
Ba	42,381	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	0	0	0	-1	0	0	0	0	0	0	0	0	0	0
Bb	41,511	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	1	1
Bc	41,243	0	0	0	1	0	0	1	0	0	0	0	1	0	0	1	1	0	0	1	1	0	0	0	1	1	0	1	1
Bd	42,729	-1	-1	0	0	0	-1	0	0	0	-1	-1	0	0	-1	0	0	0	-1	0	0	0	0	0	0	0	0	0	0
Ca	41,617	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	1	0
Cb	40,878	0	0	0	1	0	0	1	0	1	0	0	1	0	0	0	1	0	0	1	1	0	0	0	1	1	0	1	1
Cc	41,952	0	0	0	0	0	0	0	0	0	0	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Cd	43,219	-1	0	0	0	0	-1	0	0	0	-1	-1	0	0	-1	0	0	0	-1	0	0	0	0	0	0	0	0	0	0
Da	41,712	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0
Db	40,957	0	0	0	1	0	0	1	0	1	0	0	1	0	0	0	1	0	0	1	1	0	0	0	1	0	0	1	1
Dc	43,013	-1	0	0	0	0	0	0	0	0	0	-1	0	0	-1	0	0	0	-1	0	0	0	0	0	0	0	0	0	0
Dd	42,959	-1	0	0	0	0	-1	0	0	0	-1	-1	0	-1	-1	0	0	0	-1	0	0	0	0	0	0	0	0	0	0
Ea	42,066	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
Eb	41,580	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0
Ec	41,731	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
Ed	42,997	-1	-1	0	0	0	-1	0	0	0	-1	-1	0	-1	-1	0	0	-1	-1	0	0	0	-1	0	0	0	0	0	0
Fa	42,114	0	0	0	0	0	0	0	0	0	0	-1	0	0	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Fb	42,401	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Fc	43,728	-1	-1	-1	0	-1	-1	0	-1	0	-1	-1	0	-1	-1	0	0	-1	-1	0	0	-1	-1	-1	0	0	0	0	0
Fd	42,872	-1	0	0	0	0	-1	0	0	0	-1	-1	0	0	-1	0	0	0	-1	0	0	0	0	0	0	0	0	0	0
Period 13-60																													
0a	54,379	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	0	0	0	0	0	0	0	0
0b	53,757	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0c	53,914	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0d	53,546	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Aa	54,181	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Ab	54,390	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	0	0	0	0	0	0	0	0
Ac	53,114	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Ad	53,890	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Ba	53,394	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bb	54,064	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bc	53,767	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bd	53,851	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Ca	54,502	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Cb	55,381	0	0	0	0	0	0	-1	0	0	0	0	0	0	0	0	-1	0	0	0	-1	0	0	0	0	-1	0	-1	-1
Cc	54,182	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Cd	53,160	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Da	53,429	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Db	54,210	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Dc	53,752	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Dd	52,360	1	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	1	0	0	0	1	0	0
Ea	54,226	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	0	0	0	0	0	0	0	0
Eb	54,788	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	0	0	0	0	-1	0	-1	0
Ec	53,777	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Ed	53,541	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Fa	52,742	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	1	0	0
Fb	54,900	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	0	0	0	0	-1	0	-1	0
Fc	52,600	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	1	0	0
Fd	52,927	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Label 'Xx' encodes Direct Credit Back parameters and learning rate.
First letter '0': $N_{cb}=0$; 'A': $N_{cb}=1$, $\lambda=0.5$; 'A': $N_{cb}=1$, $\lambda=1$
'C': $N_{cb}=10$, $\lambda=0.5$; 'D': $N_{cb}=10$, $\lambda=1$; 'E': $N_{cb}=100$, $\lambda=0.5$; 'A': $N_{cb}=100$, $\lambda=1$;
Second letter 'a' for $\beta=0.01$, 'b' for $\beta=0.1$, 'c' for $\beta=1$, and 'd' for $\beta=10$

Table 4.25: Cross significance tests: different methods on one-period leadtime case

	sample	Look-Ahead					Sarsa: β				R: β				full DCB: β				(s,S)	RO: N_{sim}		
	mean	H1	H2	H3	H4	H5	0.01	0.1	1	10	0.01	0.1	1	10	0.01	0.1	1	10		1	10	100
Period 1-12																						
H1	72,423	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
H2	35,469	1	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
H3	34,957	1	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
H4	34,386	1	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
H5	34,646	1	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Sa	42,862	1	-1	-1	-1	-1	0	0	0	0	-1	-1	0	0	0	0	0	0	1	-1	-1	-1
Sb	42,887	1	-1	-1	-1	-1	0	0	0	0	-1	-1	0	0	0	0	0	0	1	-1	-1	-1
Sc	42,066	1	-1	-1	-1	-1	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	-1
Sd	43,000	1	-1	-1	-1	-1	0	0	0	0	-1	-1	-1	0	0	0	0	0	1	-1	-1	-1
0a	41,221	1	-1	-1	-1	-1	1	1	0	1	0	0	0	1	0	0	1	1	1	0	0	-1
0b	41,728	1	-1	-1	-1	-1	1	1	0	1	0	0	0	0	0	0	1	0	1	0	0	-1
0c	41,993	1	-1	-1	-1	-1	0	0	0	1	0	0	0	0	0	0	1	0	1	0	0	-1
0d	42,734	1	-1	-1	-1	-1	0	0	0	0	-1	0	0	0	0	0	0	0	1	-1	-1	-1
Fa	42,114	1	-1	-1	-1	-1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	-1
Fb	42,401	1	-1	-1	-1	-1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	-1
Fc	43,728	1	-1	-1	-1	-1	0	0	-1	0	-1	-1	-1	0	0	0	0	0	1	-1	-1	-1
Fd	42,872	1	-1	-1	-1	-1	0	0	0	0	-1	0	0	0	0	0	0	0	1	0	0	-1
SS	61,934	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	-1	-1	-1
RO1	41,459	1	-1	-1	-1	-1	1	1	0	1	0	0	0	1	0	0	1	0	1	0	0	-1
RO2	41,468	1	-1	-1	-1	-1	1	1	0	1	0	0	0	1	0	0	1	0	1	0	0	-1
RO3	38,659	1	-1	-1	-1	-1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
Period 13-60																						
H1	142,522	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
H2	64,207	1	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
H3	60,193	1	1	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
H4	56,976	1	1	1	0	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	-1	-1	-1	-1	-1	-1
H5	55,408	1	1	1	0	0	-1	-1	-1	0	0	0	0	0	-1	0	-1	-1	-1	-1	-1	-1
Sa	53,199	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	-1	0	0	0
Sb	52,213	1	1	1	1	1	0	0	0	0	1	0	0	0	0	1	0	0	-1	0	0	0
Sc	51,757	1	1	1	1	1	0	0	0	0	1	1	1	0	0	1	0	0	-1	0	0	0
Sd	53,691	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	0	0	0
0a	54,379	1	1	1	1	0	0	-1	-1	0	0	0	0	0	0	0	0	0	-1	-1	0	0
0b	53,757	1	1	1	1	0	0	0	-1	0	0	0	0	0	0	0	0	0	-1	0	0	0
0c	53,914	1	1	1	1	0	0	0	-1	0	0	0	0	0	0	0	0	0	-1	0	0	0
0d	53,546	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	0	0	0
Fa	52,742	1	1	1	1	1	0	0	0	0	0	0	0	0	0	1	0	0	-1	0	0	0
Fb	54,900	1	1	1	0	0	0	-1	-1	0	0	0	0	0	-1	0	-1	0	-1	-1	0	0
Fc	52,600	1	1	1	1	1	0	0	0	0	0	0	0	0	0	1	0	0	-1	0	0	0
Fd	52,927	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	-1	0	0	0
SS	38,303	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1
RO1	52,419	1	1	1	1	1	0	0	0	0	1	0	0	0	0	1	0	0	-1	0	0	0
RO2	53,426	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	-1	0	0	0
RO3	53,167	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	-1	0	0	0
Period 1-60																						
H1	214,945	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
H2	99,676	1	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	-1	-1	0	-1	-1	-1
H3	95,150	1	1	0	-1	-1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	-1
H4	91,362	1	1	1	0	0	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	0
H5	90,053	1	1	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
Sa	96,061	1	1	0	-1	-1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	-1
Sb	95,100	1	1	0	-1	-1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	-1
Sc	93,824	1	1	0	0	-1	0	0	0	1	0	0	0	0	0	1	1	0	1	0	0	0
Sd	96,691	1	1	0	-1	-1	0	0	-1	0	0	0	0	0	0	0	0	0	1	-1	0	-1
0a	95,600	1	1	0	-1	-1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	-1
0b	95,485	1	1	0	-1	-1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	-1
0c	95,907	1	1	0	-1	-1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	-1
0d	96,281	1	1	0	-1	-1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	-1
Fa	94,856	1	1	0	-1	-1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	-1
Fb	97,301	1	0	0	-1	-1	0	0	-1	0	0	0	0	0	0	0	0	0	1	-1	0	-1
Fc	96,328	1	1	0	-1	-1	0	0	-1	0	0	0	0	0	0	0	0	0	1	0	0	-1
Fd	95,798	1	1	0	-1	-1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	-1
SS	100,237	1	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	-1	-1	-1
RO1	93,878	1	1	0	-1	-1	0	0	0	1	0	0	0	0	0	1	0	0	1	0	0	-1
RO2	94,894	1	1	0	-1	-1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	-1
RO3	91,827	1	1	1	0	0	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	0

Remark 'H#' is for Look-Ahead and # is number of period(s) looking ahead
'Sx', '0x' and 'Fx' are for Sarsa, Residual Gradient and full Direct Credit Back, respectively
'a' is for $\beta = 0.01$, 'b' is for $\beta = 0.1$, 'c' is for $\beta = 1$, 'd' is for $\beta = 10$
'SS' is for (s,S) policy
'RO#' is for Rollout and #: 1, 2, and 3 are for 1, 10, and 100 repetition(s), respectively

4.8 Discussions and Conclusions

One-period leadtime problem: A one-period Look-Ahead method, H1, did not work well compared to other methods. The explanation is that H1 does not look ahead far enough to see benefit of replenishment that has not yet come within the one period. Therefore, H1 chooses the minimum allowable replenishment order to minimize aggregate cost by minimizing the replenishment cost. This leads to extremely low inventory levels and an inventory shortage, resulting in a much higher cost. For Look-Ahead methods with one to four look-ahead periods, the monotonic trend is apparent in aggregate costs that are reduced as the look ahead time is increased. A lower aggregate cost indicates a better cost performance. Performance of a Look-Ahead method with 5 look ahead periods is significantly better than the method with 4 periods.

The cost performance of an (s,S) policy is comparable to H2 for a total of 60 periods. However, performance of an (s,S) policy in the first 12 periods and the last 48 periods are significantly different than other methods. An (s,S) policy seems to overstock in the first 12 periods, which results in a disproportionately high cost in the first 12 periods, but yields an equally disproportionately low cost in the last 48 periods. Overall cost performance of an (s,S) policy, which is considered for an entire horizon of 60 periods, is comparable to the H2 method.

Using Sarsa with learning rates of 0.1 and 1 results in lower total costs than with learning rates of 0.01 and 10. Using Sarsa(0), or Sarsa without the eligibility trace, yields lower total costs than Sarsa($\lambda > 0$).

As in the zero leadtime case, it is worth investigating further the conservative behavior of Sarsa(λ). Figure 4.15 shows average aggregate costs in the upper left plot. The upper right plot shows statistics of total costs (minimum, 1st quartile, median, 3rd quartile and maximum). Average total costs are shown as a solid line. The lower left plot displays the empirical CDF for the complete range of period costs. The lower right plot displays the empirical CDF for a selected range of period costs.

The CDF values of Sarsa(1), Sarsa(0.5) and Sarsa are respectively high to low at high period cost. This result implies that a higher value of an eligibility factor has better worst-case performance. Figure 4.16 also shows a monotonic trend between an inventory level and an eligibility factor. It supports the assumption that a higher eligibility factor causes more conservative behavior when inventory is kept at a higher level.

Residual Gradient and Direct Credit Back methods perform significantly better than H2. The experimental results indicate that the Direct Credit Back method has potential for better results than the Residual Gradient method, evidenced by the fact that the Direct Credit Back method

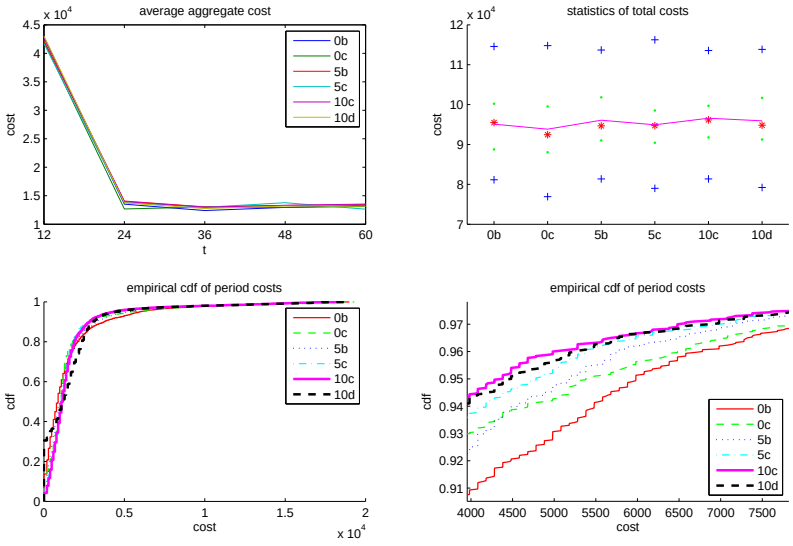

Figure 4.15: Results of Sarsa and Sarsa(λ); L1

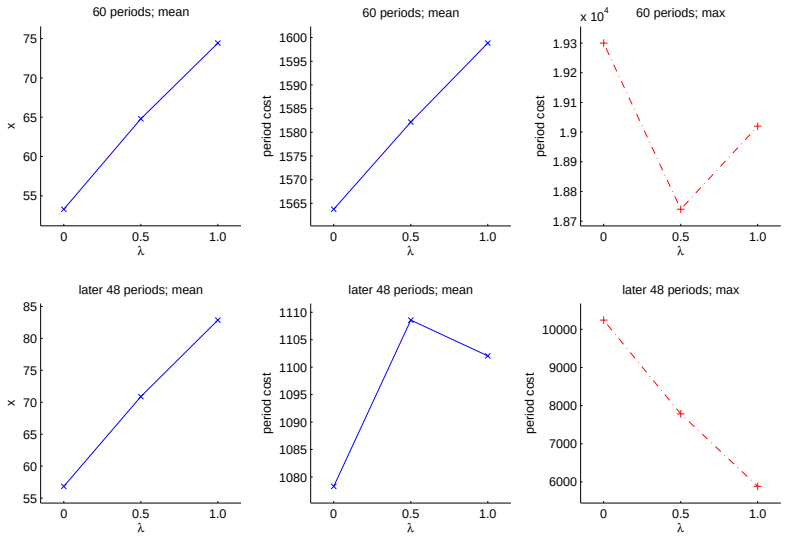

Figure 4.16: Inventory and period costs of Sarsa and Sarsa(λ); L1

yields lower average aggregate costs. However, this difference is not great enough to be confirmed by the significance tests. It should be noted that the benefit of the Direct Credit Back method may be more obvious for a delayed reward problem, but an investigation of longer leadtime inventory problems or other pure delayed-reward problems, such as Backgammon, is not attempted in this study.

For the last 48 periods, the Rollout method gives significantly better performance than the Look-Ahead method. For a total of 60 periods, the Rollout method using 1 or 10 simulations performs similarly to the three-period Look-Ahead method. The Rollout method using 100 simulations shows similar results to the four-period and five-period Look-Ahead methods. Rollout with 100 simulations has a total cost that is significantly lower than total costs obtained by using Rollout with 1 and 10 simulations. Table 4.26 shows means, standard deviations, minimums and maximums of total costs when using the Rollout method with 1, 10 and 100 simulations.

Table 4.26: Rollout numbers of simulations and total costs on one-period leadtime case

					N		
N	mean	std	min	max	1	10	100
1	93,878	6,298	81,560	109,654	0	0	-1
10	94,894	4,732	84,666	106,840	0	0	-1
100	91,827	4,753	81,083	103,087	1	1	0

Comparison among four controllers: The experimental results suggest that Sarsa has better inventory cost performance than other learning-based ADP methods in the last 48 periods. Sarsa performed significantly better than all Look-Ahead methods and most of the Residual Gradient methods. Using full Direct Credit Back or Rollout gives aggregate costs close to the costs achieved when using Sarsa. It should be noted that the linearity of the RBF may allow Sarsa to have this stability of inventory cost performance. Using a nonlinear approximation function may allow the Residual Gradient and Direct Credit Back methods to show a performance superiority over Sarsa. A nonlinear approximation function was not investigated in this study.

Regarding Rollout, the total cost of using the Rollout method with 100 simulations was not significantly different than the cost of using the Look-Ahead method with looking four or five periods ahead. The Rollout method performed significantly better than the Residual Gradient and full Direct Credit Back methods. With the best choice of parameters, using the Rollout method gave a lower total cost than Sarsa.

In conclusion, among learning-based methods, Sarsa outperformed the Sarsa(λ), Residual Gradient and Direct Credit Back methods for inventory problems. In addition to its requirement for more

computation time and memory, an eligibility trace technique seems not suitable for this fast-return problem structure. It had a higher aggregate cost as indicated by experimental results when using Sarsa(λ)

In addition to Sarsa, Rollout is a good alternative method when a model of the problem is available. Another benefit of Rollout is that it does not require a warm-up period and it performs consistently well from the outset.

CHAPTER 5

AN INVENTORY PROBLEM WITH HIGH VARIANCE DEMAND

Recently Zhang [130] found evidence of the GARCH(1,1) model in inventory data and showed that there are significant costs when GARCH(1,1) is not accounted for. He used an analytical approach to develop a formula for an order-up-to-level policy for an AR1/GARCH(1,1) problem without a setup cost, however, his analytical approach was too problem specific. A slight change to this problem structure requires rigorous reanalysis of the problem and subsequent redevelopment of the solution. The problem investigated in our study includes the setup cost, making the cost function highly non-linear, which in turn, makes an analytical solution difficult to achieve.

Approximate Dynamic Programming (ADP) has been shown to have generality and flexibility to overcome the shortcomings of most analytical approaches. As a result, ADP has gained much attention in inventory management research. However, none of the previous ADP studies have investigated an inventory problem with the GARCH(1,1) model structure included. The GARCH(1,1) model introduces two latent state variables that inadvertently will be left out if GARCH(1,1) is not accounted for. This has posed a challenge to a model-free property of a learning-based ADP method.

In addition to learning-based ADP, there is simulation-based ADP which has received less attention in inventory research. A simulation-based ADP method uses simulation, instead of a learning scheme, to provide an approximate state-action cost function. Since no analytical model of the problem is required, a learning-based ADP method needs only initial periods to attain good approximation of state-action costs. When a model of the problem is provided, the model can be integrated into the simulation-based ADP.

This chapter investigates the application of a learning-based method Sarsa and two simulation-based methods Rollout and Hindsight Optimization (HO) to an inventory problem with AR1/GARCH(1,1) demand. Inventory cost performance of each method is evaluated through simulation-based experiments.

5.1 An inventory problem with AR1/GARCH(1,1) demand

The problem investigated here is similar to the one-period leadtime inventory problem discussed in Chapter 4, but the demand is modeled with AR1/GARCH(1,1). An inventory problem with AR1/GARCH(1,1) demand has a system state s = $(D_t,\ z_t,\ \sigma_t^2,\ x_{t+1},\ B_1^{(t+1)},\ \cdots,\ B_L^{(t+1)})$. The state transition is shown in Equations 5.1, 5.2, 5.3, 5.4 and 5.5.

$$
\begin{aligned}
D_t &= a_0 + a_1 \cdot D_{t-1} + z_t \\
&= a_0 + a_1 \cdot D_{t-1} + e_t \cdot \sqrt{\nu + \alpha \cdot z_{t-1}^2 + \beta \cdot \sigma_{t-1}^2} && (5.1) \\
z_t &= e_t \cdot \sigma_t \\
&= e_t \cdot \sqrt{\nu + \alpha \cdot z_{t-1}^2 + \beta \cdot \sigma_{t-1}^2} && (5.2) \\
\sigma_t^2 &= \nu + \alpha \cdot z_{t-1}^2 + \beta \cdot \sigma_{t-1}^2 && (5.3) \\
x_{t+1} &= x_t + B_1^{(t)} - D_t \\
&= x_t + B_1^{(t)} - a_0 - a_1 \cdot D_{t-1} - e_t \cdot \sqrt{\nu + \alpha \cdot z_{t-1}^2 + \beta \cdot \sigma_{t-1}^2} && (5.4) \\
B_1^{(t+1)} &= B_2^{(t)} \\
\vdots &\ \vdots\ \vdots \\
B_{L-1}^{(t+1)} &= B_L^{(t)} \\
B_L^{(t+1)} &= u_t && (5.5)
\end{aligned}
$$

where e_t is white noise distributed $N(0,1)$, u_t is a replenishment order and $t = \{2, 3, 4, \ldots\}$.

The transition cost (Equation 3.8) is not affected by the GARCH(1,1) model.

5.2 Experiments

Simulation-based experiments are conducted to evaluate Sarsa, Rollout and HO. Each inventory controller is run for 50 replications of each of 60 time-unit-indeterminate periods. The problem is set up with K_t =\$80, c_t = \$100/unit, h_t = \$1/unit and b_t = \$180/unit. An undiscounted one-period leadtime problem ($\alpha = 1$ and $L = 1$) is investigated. The demand is modeled with AR1/GARCH(1,1), where the AR1's $a_0 = 2$ and $a_1 = 0.8$ and GARCH(1,1)'s $\nu_0 = 70$, $\nu_1 = 0.1$ and $\nu_2 = 0.8$. Each experiment is initialized at $D_0 = 50$, $z_0 = 10$, $\sigma_0^2 = 400$, $x_1 = 10$ and $B^{(1)} = 0$.

Look-Ahead: Look-Ahead is used for comparison to other methods. The cost and state projections in this experiment are average projections assuming zero demand noise.

Sarsa: Sarsa in this experiment is conducted with an RBF whose centers are located at each combination of {0, 100, 200, 300} × {-200, -100, 0, 100, 200} × {0, 800, 1600, 2400, 3200} × {-300, -200, -100, ..., 300 } × {0, 100, 200, 300, 400} × {0, 100, 200, 300, 400 }, corresponding to previous demand, demand noise, demand noise variance, on-site inventory, in-transit inventory and replenishment order respectively. The RBF scales are set up with a 1/2-midpoint strategy.

Therefore the RBF scales corresponding to previous demand, demand noise, on-site inventory, in-transit inventory and replenishment order are 0.4621×10^{-4} and the scales corresponding to demand noise variance are 0.0072×10^{-4}.

An exhaustive search is used to determine an action with the lowest Q-value for the given state. Consistent with Van Roy et al. [118], noise[1] is added to the optimal replenishment order found by the exhaustive search to create an explorative mechanism.

Sarsa without latent state variables: Demand D_t, on-site inventory x_{t+1} and in-transit inventory $\mathrm{B}^{(t+1)}$ of the AR1 model can be observed directly. The GARCH model introduces two extra variables z_{t-1} and σ^2_{t-1} and these two variables are less observable. Current literatures, including Das et al. [36], indicate a general belief that Sarsa and other learning-based ADP methods are model-free approaches that do not require an accurate model of the problem. However, the GARCH(1,1) variables will be left out if the GARCH(1,1) model is not to be accounted for. Our experiment investigates the application of Sarsa to AR1/GARCH(1,1) problems when the GARCH(1,1) model is not accounted for. Sarsa without the GARCH(1,1) variables is used with the RBF centers located at each combination of {0, 100, 200, 300} × {-300, -200, -100, ..., 300 } × {0, 100, 200, 300, 400 } × {0, 100, 200, 300, 400 }, corresponding to previous demand, on-site inventory, in-transit inventory and replenishment order. All RBF scales are 0.6931.

It should be noted that the extra state variables introduced by the GARCH(1,1) model differ from unobservable variables in a Partially Observable Markov Decision Problem (POMDP). When the GARCH(1,1) model is not accounted for, there is no extra effort made to handle the missing information. By contrast, POMDP provides a method to handle unknown values of unobservable variables.

Simulation-based ADP methods: Two simulation-based ADP methods, Rollout and Hindsight Optimization, are used. The Rollout method uses simulation to provide approximate state-action costs, which in turn are used for determining actions. The Rollout simulation requires a base policy to determine actions for the simulated events. This experiment uses an (s,S) policy for Rollout's base policy. The (s,S) policy parameters are determined by the Economic Order Quantity (EOQ) and safety stock calculation, as discussed in Section 4.6. The Hindsight Optimization (HO) is another simulation-based ADP method, HO does not require a base policy. HO uses a special simulation, which is discussed in Section 2.4.4. Both Rollout and HO used 1, 5, 10 and 50 simulations and 1, 3, 6 and 12 simulation horizons.

[1] Noise is limited normal, as being discussed in Section 7.1, and rounded to integer. According to experimental results from preliminary tests, this study uses 100 for standard deviation of this limited normal noise.

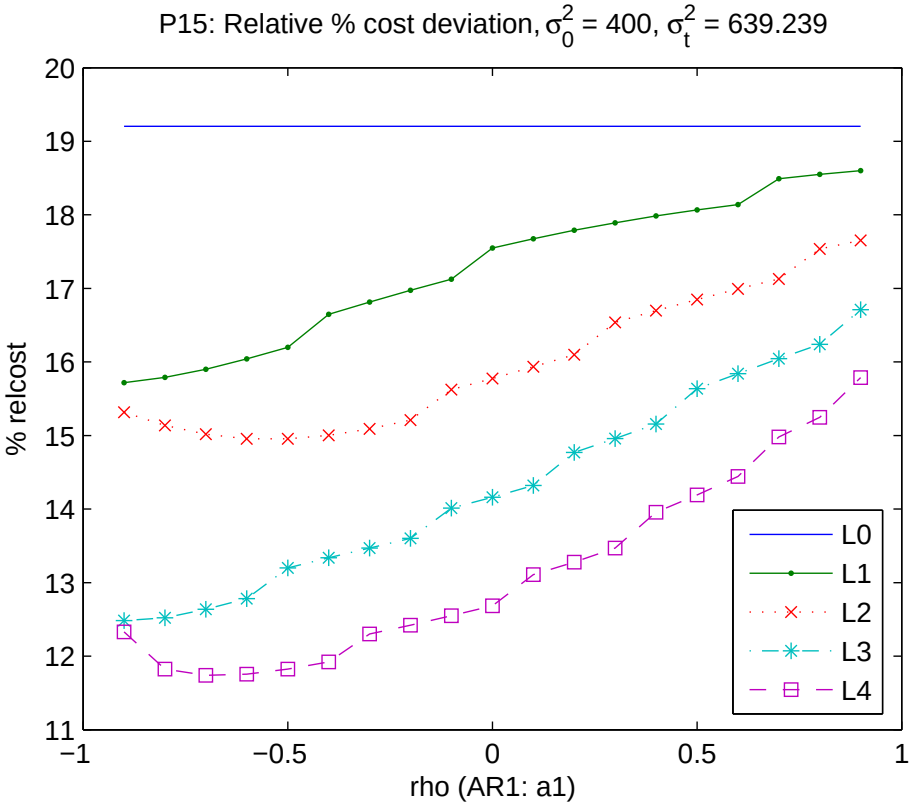

Figure 5.1: Relative cost deviation (%) showing GARCH significance

Evaluation: Aggregate costs are used as performance indicators. They are aggregated in the first 12 periods, the subsequent 48 periods (Period 13-60), and the total 60 periods. Statistical significance tests are used to compare results. A one-sided t test is used when the normality assumption of experimental results holds. The normality assumption is tested at a 5% significance level with both Lilliefors and χ^2 goodness-of-fit tests. When it is suspected that the normality assumption does not hold, a Wilcoxon rank sum test is used in place of the t test.

Problem selection: The problem in this Chapter is selected to investigate the effect of the GARCH(1,1) model on the performance of the ADP approach. Zhang's [130, Eq. 28, pp. 136] relative cost deviation is displayed in Figure 5.1. The relative cost deviation shows the estimated effect of the GARCH(1,1) model. The parameter σ_t^2 is set to 639.239, an average σ_t^2 obtained from a preliminary experiment. With AR1's $a_1 = 0.8$ and $L = 1$, the inventory cost is expected to be around 18% higher when the GARCH(1,1) model is not accounted for. It should be noted that Zhang's relative cost deviation formula is developed without a setup cost, but with the assumption that his formula is sufficient to show the significance of the GARCH(1,1) model in this problem.

5.3 Experimental Results

Tables 5.1, 5.2, 5.3 and 5.4 show the significance test results of Sarsa, Sarsa without GARCH(1,1) state variables, Rollout and Hindsight Optimization, respectively.

Table 5.1: Significance tests: Look-Ahead and Sarsa; GARCH(1,1)

	sample	BCa interval		test	reject H_0		p value		Rank sum		Normal
	mean	LCI	UCI	stat.	H_{a^+}	H_{a^-}	H_{a^+}	H_{a^-}	H_*	p val.	
Period 1-12; H2 (sample mean = 72,219.00; CI = [61,635.36 ; 85,434.64]; both normal tests are failed.											
H6	61,453.86	52,057.07	72,818.43	1.33	0	0	0.09	0.91	0	0.23	1
H12	69,539.66	62,858.25	77,525.38	0.37	0	0	0.35	0.65	0	0.64	0
S0.1	82,247.48	76,947.54	88,473.33	-1.49	0	0	0.93	0.07	1	0.05	0
S0.3	81,375.74	76,598.65	86,846.95	-1.38	0	0	0.91	0.09	1	0.06	1
S0.4	81,445.44	77,012.61	85,718.19	-1.43	0	0	0.92	0.08	1	0.06	0
S0.5	80,044.56	75,640.90	84,428.21	-1.21	0	0	0.88	0.12	1	0.09	0
S0.6	81,874.68	76,537.04	87,949.14	-1.43	0	0	0.92	0.08	1	0.04	0
S0.7	84,121.70	79,057.02	89,714.21	-1.79	0	1	0.96	0.04	1	0.03	0
S1.0	84,668.28	79,143.84	92,172.33	-1.80	0	1	0.96	0.04	1	0.03	1
Period 13-60; H2 (sample mean = 310,081.52; CI = [277,459.44 ; 346,678.20]; normal test is passed.											
H6	239,951.12	213,764.32	267,080.89	3.12	1	0	0.00	1.00	1	0.01	0
H12	200,561.66	177,978.85	226,373.99	5.04	1	0	0.00	1.00	1	0.00	0
S0.1	176,901.12	160,481.52	195,396.90	6.65	1	0	0.00	1.00	1	0.00	0
S0.3	176,856.26	160,289.96	194,486.32	6.68	1	0	0.00	1.00	1	0.00	0
S0.4	176,286.74	160,383.55	194,096.69	6.73	1	0	0.00	1.00	1	0.00	0
S0.5	176,637.76	160,971.83	194,546.18	6.72	1	0	0.00	1.00	1	0.00	0
S0.6	176,629.98	159,947.35	194,697.34	6.65	1	0	0.00	1.00	1	0.00	0
S0.7	175,695.92	160,340.18	193,554.60	6.77	1	0	0.00	1.00	1	0.00	0
S1.0	175,630.16	158,708.16	192,933.09	6.75	1	0	0.00	1.00	1	0.00	0
Period 1-60; H2 (sample mean = 382,300.52; CI = [345,731.40 ; 425,668.27]; normal test is passed.											
H6	301,404.98	273,115.26	333,919.71	3.13	1	0	0.00	1.00	1	0.00	0
H12	270,101.32	246,114.90	300,273.64	4.54	1	0	0.00	1.00	1	0.00	0
S0.1	259,148.60	240,910.38	279,723.02	5.42	1	0	0.00	1.00	1	0.00	0
S0.3	258,232.00	239,619.98	278,937.07	5.44	1	0	0.00	1.00	1	0.00	0
S0.4	257,732.18	240,081.09	277,395.21	5.51	1	0	0.00	1.00	1	0.00	0
S0.5	256,682.32	238,793.59	276,053.08	5.55	1	0	0.00	1.00	1	0.00	0
S0.6	258,504.66	239,196.04	280,014.28	5.40	1	0	0.00	1.00	1	0.00	0
S0.7	259,817.62	242,301.72	280,859.67	5.38	1	0	0.00	1.00	1	0.00	0
S1.0	260,298.44	241,840.00	282,450.66	5.32	1	0	0.00	1.00	1	0.00	0

Figure 5.2 shows BCa confidence intervals and means of 2-period Look-Ahead, 6-period Look-Ahead, 12-period Look-Ahead and Sarsa with learning rates of 0.1, 0.3, 0.4, 0.5, 0.6, 0.7 and 1.0. A Look-Ahead method is labeled "H#" where the suffix number indicates a number of period(s) looking ahead. Sarsa is labeled "S#" where the suffix number indicates a learning rate.

Table 5.5 shows a summary of the cross significance tests of aggregate costs obtained with Look-Ahead and Sarsa. The tables and figures are organized as described in Chapter 4.

Table 5.2: Significance tests: Sarsa w/o z & σ^2; GARCH(1,1)

β	sample mean	BCa interval LCI	BCa interval UCI	test stat.	reject H_0 H_a+	reject H_0 H_a-	p value H_a+	p value H_a-	Rank sum H_*	Rank sum p val.	Normal
Period 1-12; H2 (sample mean = 72,219.00; CI = [61,543.17 ; 85,084.63]; both normal tests are failed.											
0.1	83,090.02	77,930.31	89,536.62	-1.61	0	0	0.94	0.06	1	0.04	1
0.2	80,536.30	75,597.50	86,361.28	-1.25	0	0	0.89	0.11	1	0.05	0
0.3	80,929.28	76,053.04	86,377.47	-1.32	0	0	0.90	0.10	1	0.07	0
0.4	81,578.92	76,425.25	87,485.67	-1.40	0	0	0.92	0.08	1	0.05	0
0.5	81,300.24	76,236.28	86,914.50	-1.36	0	0	0.91	0.09	1	0.04	0
0.6	80,347.90	75,841.16	85,496.90	-1.24	0	0	0.89	0.11	1	0.08	1
0.7	81,980.48	76,901.17	88,182.61	-1.45	0	0	0.92	0.08	1	0.04	0
0.8	80,287.12	75,699.18	84,847.97	-1.24	0	0	0.89	0.11	1	0.07	0
0.9	79,466.22	74,409.33	84,936.58	-1.09	0	0	0.86	0.14	1	0.09	0
1.0	82,813.74	77,280.74	89,766.41	-1.55	0	0	0.94	0.06	1	0.04	1
Period 13-60; H2 (sample mean = 310,081.52; CI = [277,301.94 ; 346,957.31]; normal test is passed.											
0.1	175,832.94	159,935.69	193,806.61	6.74	1	0	0.00	1.00	1	0.00	0
0.2	175,862.76	159,558.54	192,986.43	6.73	1	0	0.00	1.00	1	0.00	0
0.3	177,015.94	161,040.13	195,081.39	6.65	1	0	0.00	1.00	1	0.00	0
0.4	176,668.10	159,656.01	194,492.90	6.66	1	0	0.00	1.00	1	0.00	0
0.5	176,597.12	159,917.07	195,650.91	6.66	1	0	0.00	1.00	1	0.00	0
0.6	176,949.54	160,130.27	194,464.77	6.67	1	0	0.00	1.00	1	0.00	0
0.7	176,977.76	160,411.47	195,557.06	6.63	1	0	0.00	1.00	1	0.00	0
0.8	173,607.68	157,701.36	190,363.00	6.91	1	0	0.00	1.00	1	0.00	0
0.9	176,304.38	159,047.17	194,841.89	6.69	1	0	0.00	1.00	1	0.00	0
1.0	176,714.20	160,429.65	194,898.88	6.67	1	0	0.00	1.00	1	0.00	0
Period 1-60; H2 (sample mean = 382,300.52; CI = [344,932.50 ; 425,016.47]; normal test is passed.											
0.1	258,922.96	239,988.88	279,664.00	5.39	1	0	0.00	1.00	1	0.00	0
0.2	256,399.06	238,177.11	275,617.06	5.57	1	0	0.00	1.00	1	0.00	0
0.3	257,945.22	239,505.39	278,329.29	5.46	1	0	0.00	1.00	1	0.00	0
0.4	258,247.02	239,215.03	279,068.45	5.42	1	0	0.00	1.00	1	0.00	0
0.5	257,897.36	239,607.37	278,761.04	5.45	1	0	0.00	1.00	1	0.00	0
0.6	257,297.44	239,455.99	277,267.63	5.50	1	0	0.00	1.00	1	0.00	0
0.7	258,958.24	239,961.61	279,510.56	5.39	1	0	0.00	1.00	1	0.00	0
0.8	253,894.80	237,033.02	271,632.70	5.74	1	0	0.00	1.00	1	0.00	0
0.9	255,770.60	237,566.39	276,135.57	5.54	1	0	0.00	1.00	1	0.00	0
1.0	259,527.94	241,685.66	279,388.09	5.41	1	0	0.00	1.00	1	0.00	0

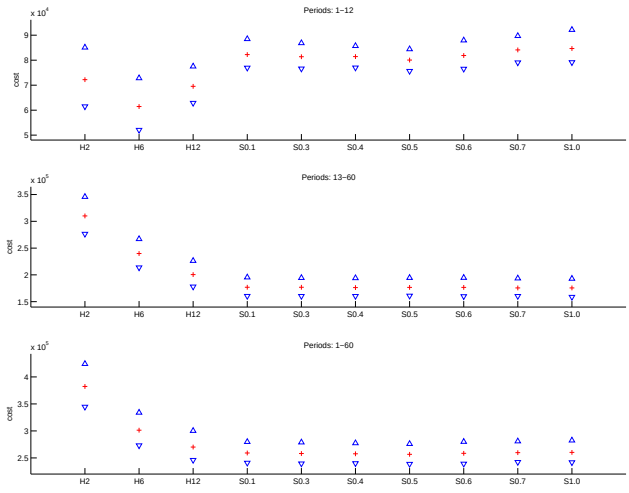

Figure 5.2: Average aggregate costs from Sarsa; GARCH(1,1)

Table 5.3: Significance tests: Rollout; GARCH(1,1)

treatment		sample mean	BCa interval LCI	BCa interval UCI	test stat.	reject H_0 H_{a+}	reject H_0 H_{a-}	p value H_{a+}	p value H_{a-}	Rank sum H_*	Rank sum p val.	Normal
Period 1-12; H2 sample mean = 72,219.00; CI = [61,331.91 ; 84,708.83]; both normal tests are failed.												
RN Rollout		62,495.86	53,789.17	72,589.31	1.25	0	0	0.11	0.89	0	0.39	0
N	T											
1	1	101,167.74	85,644.63	119,331.87	-2.76	0	1	1.00	0.00	1	0.02	1
1	3	80,196.92	68,595.30	94,978.69	-0.88	0	0	0.81	0.19	0	0.41	1
1	6	60,048.68	52,967.20	72,415.84	1.58	0	0	0.06	0.94	0	0.25	1
1	12	63,978.64	57,645.82	73,378.15	1.14	0	0	0.13	0.87	0	0.73	1
5	1	74,833.02	64,589.02	86,419.51	-0.32	0	0	0.62	0.38	0	0.57	0
5	3	59,826.46	53,577.24	66,963.86	1.77	1	0	0.04	0.96	0	0.30	1
5	6	61,492.16	56,607.65	67,877.79	1.59	0	0	0.06	0.94	0	0.69	1
5	12	74,891.28	69,368.26	80,874.52	-0.40	0	0	0.65	0.35	0	0.24	0
10	1	74,359.86	64,502.87	85,269.06	-0.26	0	0	0.60	0.40	0	0.73	0
10	3	59,364.14	53,654.57	66,269.97	1.86	1	0	0.03	0.97	0	0.38	1
10	6	66,114.00	60,809.95	72,200.82	0.90	0	0	0.18	0.82	0	0.87	1
10	12	81,970.96	77,328.27	87,131.96	-1.49	0	0	0.93	0.07	1	0.04	0
50	1	69,701.82	59,602.14	81,436.04	0.30	0	0	0.38	0.62	0	0.91	1
50	3	60,928.78	55,422.18	67,343.71	1.66	0	0	0.05	0.95	0	0.55	1
50	6	68,874.68	62,630.03	76,801.54	0.47	0	0	0.32	0.68	0	0.73	1
50	12	80,876.42	74,905.56	88,518.78	-1.24	0	0	0.89	0.11	1	0.07	1
Period 13-60; H2 (sample mean = 310,081.52; CI = [276,704.79 ; 346,140.32]; normal test is passed.												
RN Rollout		273,276.80	240,576.10	307,833.01	1.47	0	0	0.07	0.93	0	0.14	0
N	T											
1	1	342,712.88	304,913.08	387,441.28	-1.18	0	0	0.88	0.12	0	0.42	1
1	3	265,272.58	233,899.42	303,190.44	1.78	1	0	0.04	0.96	1	0.08	0
1	6	231,378.82	205,984.42	263,420.39	3.41	1	0	0.00	1.00	1	0.00	1
1	12	170,396.46	154,913.57	188,978.48	7.02	1	0	0.00	1.00	1	0.00	0
5	1	297,242.04	260,620.18	347,165.95	0.45	0	0	0.33	0.67	0	0.34	0
5	3	168,878.42	152,996.68	187,051.86	7.11	1	0	0.00	1.00	1	0.00	0
5	6	164,868.72	148,795.65	184,455.37	7.21	1	0	0.00	1.00	1	0.00	0
5	12	168,837.32	155,064.61	182,918.80	7.30	1	0	0.00	1.00	1	0.00	0
10	1	272,222.48	242,427.04	310,826.80	1.51	0	0	0.07	0.93	0	0.12	0
10	3	176,742.46	159,749.01	199,556.10	6.50	1	0	0.00	1.00	1	0.00	0
10	6	148,003.02	134,818.24	163,325.81	8.37	1	0	0.00	1.00	1	0.00	0
10	12	174,005.32	162,155.82	186,587.10	7.17	1	0	0.00	1.00	1	0.00	1
50	1	251,982.76	225,997.85	281,880.16	2.54	1	0	0.01	0.99	1	0.02	0
50	3	152,749.60	140,935.69	165,580.55	8.29	1	0	0.00	1.00	1	0.00	0
50	6	158,996.16	145,802.27	173,903.91	7.84	1	0	0.00	1.00	1	0.00	1
50	12	162,406.70	149,755.37	176,269.81	7.73	1	0	0.00	1.00	1	0.00	0
Period 1-60; H2 (sample mean = 382,300.52; CI = [345,012.17 ; 423,990.21]; normal test is passed.												
RN Rollout		335,772.66	299,997.40	378,289.64	1.62	0	0	0.05	0.95	1	0.09	0
N	T											
1	1	443,880.62	400,238.40	497,362.78	-1.91	0	1	0.97	0.03	0	0.13	1
1	3	345,469.50	309,325.83	386,845.09	1.30	0	0	0.10	0.90	0	0.23	0
1	6	291,427.50	265,349.22	323,351.94	3.57	1	0	0.00	1.00	1	0.00	1
1	12	234,375.10	215,665.99	255,694.78	6.44	1	0	0.00	1.00	1	0.00	0
5	1	372,075.06	332,107.19	427,676.78	0.32	0	0	0.37	0.63	0	0.43	1
5	3	228,704.88	210,359.91	249,961.56	6.72	1	0	0.00	1.00	1	0.00	1
5	6	226,360.88	208,955.94	246,999.12	6.86	1	0	0.00	1.00	1	0.00	1
5	12	243,728.60	229,419.67	259,066.10	6.33	1	0	0.00	1.00	1	0.00	1
10	1	346,582.34	314,236.17	384,091.16	1.31	0	0	0.10	0.90	0	0.24	0
10	3	236,106.60	217,680.55	260,235.21	6.32	1	0	0.00	1.00	1	0.00	1
10	6	214,117.02	198,562.47	231,452.24	7.58	1	0	0.00	1.00	1	0.00	0
10	12	255,976.28	242,279.18	270,050.07	5.82	1	0	0.00	1.00	1	0.00	0
50	1	321,684.58	293,918.89	350,986.10	2.41	1	0	0.01	0.99	1	0.04	0
50	3	213,678.38	201,285.17	226,734.96	7.84	1	0	0.00	1.00	1	0.00	0
50	6	227,870.84	213,815.00	245,490.95	7.02	1	0	0.00	1.00	1	0.00	1
50	12	243,283.12	229,993.16	258,372.03	6.39	1	0	0.00	1.00	1	0.00	0

Table 5.4: Significance tests: Hindsight Optimization; GARCH(1,1)

HO		sample	BCa interval		test	reject H_0		p value		Rank sum		Normal
N	T	mean	LCI	UCI	stat.	H_a+	H_a-	H_a+	H_a-	H_*	p val.	
Period 1-12; H2 sample mean = 72,219.00; CI = [61,328.00 ; 85,232.99]; both normal tests are failed.												
1	1	55,917.04	49,465.69	63,759.48	2.30	1	0	0.01	0.99	0	0.12	1
1	3	62,163.54	56,808.91	68,739.13	1.47	0	0	0.07	0.93	0	0.69	1
1	6	75,479.78	70,288.63	81,191.21	-0.49	0	0	0.69	0.31	0	0.24	0
1	12	79,341.64	75,085.30	84,086.20	-1.09	0	0	0.86	0.14	1	0.07	0
5	1	75,307.76	65,255.55	87,852.51	-0.37	0	0	0.64	0.36	0	0.48	1
5	3	77,061.54	73,457.84	81,639.81	-0.75	0	0	0.77	0.23	0	0.14	1
5	6	88,551.20	83,525.90	94,021.97	-2.46	0	1	0.99	0.01	1	0.00	0
5	12	86,208.66	81,674.15	92,063.01	-2.11	0	1	0.98	0.02	1	0.01	0
10	1	73,541.80	63,850.89	87,174.75	-0.16	0	0	0.56	0.44	0	0.59	1
10	3	80,412.50	76,238.45	85,358.87	-1.26	0	0	0.89	0.11	1	0.06	1
10	6	83,870.92	79,013.57	89,082.21	-1.76	0	1	0.96	0.04	1	0.03	0
10	12	84,325.66	79,719.89	90,354.48	-1.83	0	1	0.96	0.04	1	0.02	1
Period 13-60; H2 sample mean = 310,081.52; CI = [276,143.83 ; 345,924.87]; normal test is passed.												
1	1	202,294.48	182,515.68	226,967.17	5.10	1	0	0.00	1.00	1	0.00	1
1	3	173,871.12	155,304.29	199,159.49	6.47	1	0	0.00	1.00	1	0.00	1
1	6	170,902.52	155,128.02	187,639.86	7.04	1	0	0.00	1.00	1	0.00	0
1	12	171,478.82	158,147.75	184,534.25	7.23	1	0	0.00	1.00	1	0.00	0
5	1	189,142.30	167,370.30	217,095.47	5.53	1	0	0.00	1.00	1	0.00	1
5	3	168,023.52	155,883.18	182,672.19	7.40	1	0	0.00	1.00	1	0.00	0
5	6	160,976.50	146,823.85	175,322.22	7.71	1	0	0.00	1.00	1	0.00	0
5	12	155,991.80	144,912.01	167,930.05	8.17	1	0	0.00	1.00	1	0.00	0
10	1	185,780.54	166,081.82	209,378.16	5.89	1	0	0.00	1.00	1	0.00	1
10	3	160,253.60	149,513.35	171,946.95	7.97	1	0	0.00	1.00	1	0.00	0
10	6	168,517.98	157,229.55	181,539.75	7.47	1	0	0.00	1.00	1	0.00	0
10	12	164,548.96	150,218.44	179,740.53	7.47	1	0	0.00	1.00	1	0.00	0
Period 1-60; H2 sample mean = 382,300.52; CI = [345,064.13 ; 423,971.07]; normal test is passed.												
1	1	258,211.52	236,529.80	284,284.12	5.20	1	0	0.00	1.00	1	0.00	1
1	3	236,034.66	216,798.24	263,542.49	6.18	1	0	0.00	1.00	1	0.00	0
1	6	246,382.30	229,383.61	264,429.12	6.07	1	0	0.00	1.00	1	0.00	0
1	12	250,820.46	236,401.41	265,188.79	6.04	1	0	0.00	1.00	1	0.00	0
5	1	264,450.06	240,577.02	293,804.26	4.78	1	0	0.00	1.00	1	0.00	0
5	3	245,085.06	231,097.36	262,556.44	6.26	1	0	0.00	1.00	1	0.00	0
5	6	249,527.70	234,028.95	265,385.17	6.03	1	0	0.00	1.00	1	0.00	0
5	12	242,200.46	229,501.52	256,022.39	6.49	1	0	0.00	1.00	1	0.00	0
10	1	259,322.34	236,919.13	289,346.04	5.02	1	0	0.00	1.00	1	0.00	1
10	3	240,666.10	227,956.42	253,629.43	6.57	1	0	0.00	1.00	1	0.00	0
10	6	252,388.90	238,314.50	266,530.81	5.98	1	0	0.00	1.00	1	0.00	0
10	12	248,874.62	234,501.89	264,274.27	6.09	1	0	0.00	1.00	1	0.00	0

Table 5.5: Cross significance tests: Look-Ahead and Sarsa; GARCH(1,1)

	sample	Look-Ahead						Sarsa						
	mean	H2	H3	H4	H5	H6	H12	0.1	0.3	0.4	0.5	0.6	0.7	1
Period 1-12														
H2	72,219.00	0	0	0	0	0	0	1	1	1	1	1	1	1
H3	64,071.36	0	0	0	0	0	0	1	1	1	1	1	1	1
H4	62,253.06	0	0	0	0	0	0	1	1	1	1	1	1	1
H5	59,859.90	0	0	0	0	0	0	1	1	1	1	1	1	1
H6	61,453.86	0	0	0	0	0	1	1	1	1	1	1	1	1
H12	69,539.66	0	0	0	0	-1	0	1	1	1	1	1	1	1
S0.1	82,247.48	-1	-1	-1	-1	-1	-1	0	0	0	0	0	0	0
S0.3	81,375.74	-1	-1	-1	-1	-1	-1	0	0	0	0	0	0	0
S0.4	81,445.44	-1	-1	-1	-1	-1	-1	0	0	0	0	0	0	0
S0.5	80,044.56	-1	-1	-1	-1	-1	-1	0	0	0	0	0	0	0
S0.6	81,874.68	-1	-1	-1	-1	-1	-1	0	0	0	0	0	0	0
S0.7	84,121.70	-1	-1	-1	-1	-1	-1	0	0	0	0	0	0	0
S1.0	84,668.28	-1	-1	-1	-1	-1	-1	0	0	0	0	0	0	0
Period 13-60														
H2	310,081.52	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
H3	270,369.24	1	0	0	0	0	-1	-1	-1	-1	-1	-1	-1	-1
H4	254,758.30	1	0	0	0	0	-1	-1	-1	-1	-1	-1	-1	-1
H5	243,655.04	1	0	0	0	0	-1	-1	-1	-1	-1	-1	-1	-1
H6	239,951.12	1	0	0	0	0	-1	-1	-1	-1	-1	-1	-1	-1
H12	200,561.66	1	1	1	1	1	0	0	0	0	0	0	0	0
S0.1	176,901.12	1	1	1	1	1	0	0	0	0	0	0	0	0
S0.3	176,856.26	1	1	1	1	1	0	0	0	0	0	0	0	0
S0.4	176,286.74	1	1	1	1	1	0	0	0	0	0	0	0	0
S0.5	176,637.76	1	1	1	1	1	0	0	0	0	0	0	0	0
S0.6	176,629.98	1	1	1	1	1	0	0	0	0	0	0	0	0
S0.7	175,695.92	1	1	1	1	1	0	0	0	0	0	0	0	0
S1.0	175,630.16	1	1	1	1	1	0	0	0	0	0	0	0	0
Period 1-60														
H2	382,300.52	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
H3	334,440.60	1	0	0	0	0	-1	-1	-1	-1	-1	-1	-1	-1
H4	317,011.36	1	0	0	0	0	-1	-1	-1	-1	-1	-1	-1	-1
H5	303,514.94	1	0	0	0	0	0	-1	-1	-1	-1	-1	-1	-1
H6	301,404.98	1	0	0	0	0	0	-1	-1	-1	-1	-1	-1	-1
H12	270,101.32	1	1	1	0	0	0	0	0	0	0	0	0	0
S0.1	259,148.60	1	1	1	1	1	0	0	0	0	0	0	0	0
S0.3	258,232.00	1	1	1	1	1	0	0	0	0	0	0	0	0
S0.4	257,732.18	1	1	1	1	1	0	0	0	0	0	0	0	0
S0.5	256,682.32	1	1	1	1	1	0	0	0	0	0	0	0	0
S0.6	258,504.66	1	1	1	1	1	0	0	0	0	0	0	0	0
S0.7	259,817.62	1	1	1	1	1	0	0	0	0	0	0	0	0
S1.0	260,298.44	1	1	1	1	1	0	0	0	0	0	0	0	0

Sarsa does not show significant differences in performance for different learning rates. For simplicity of presentation, Sarsa with the learning rate of 0.5 is selected to represent Sarsa in Table 5.6 and Figure 5.3.

Table 5.6: Cross significance tests: Look-Ahead, Sarsa, and Sarsa w/o z & σ^2; GARCH(1,1)

	sample	Look-Ahead			S	Sarsa w/o GARCH(1,1) state variables									
treatment	mean	2	6	12		0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
Period 1-12															
H2	72,219.00	0	0	0	1	1	1	1	1	1	1	1	1	1	1
H6	61,453.86	0	0	1	1	1	1	1	1	1	1	1	1	1	1
H12	69,539.66	0	-1	0	1	1	1	1	1	1	1	1	1	1	1
S	80,044.56	-1	-1	-1	0	0	0	0	0	0	0	0	0	0	0
A0.1	83,090.02	-1	-1	-1	0	0	0	0	0	0	0	0	0	0	0
A0.2	80,536.30	-1	-1	-1	0	0	0	0	0	0	0	0	0	0	0
A0.3	80,929.28	-1	-1	-1	0	0	0	0	0	0	0	0	0	0	0
A0.4	81,578.92	-1	-1	-1	0	0	0	0	0	0	0	0	0	0	0
A0.5	81,300.24	-1	-1	-1	0	0	0	0	0	0	0	0	0	0	0
A0.6	80,347.90	-1	-1	-1	0	0	0	0	0	0	0	0	0	0	0
A0.7	81,980.48	-1	-1	-1	0	0	0	0	0	0	0	0	0	0	0
A0.8	80,287.12	-1	-1	-1	0	0	0	0	0	0	0	0	0	0	0
A0.9	79,466.22	-1	-1	-1	0	0	0	0	0	0	0	0	0	0	0
A1.0	82,813.74	-1	-1	-1	0	0	0	0	0	0	0	0	0	0	0
Period 13-60															
H2	310,081.52	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
H6	239,951.12	1	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
H12	200,561.66	1	1	0	0	0	0	0	0	0	0	0	-1	0	0
S	176,637.76	1	1	0	0	0	0	0	0	0	0	0	0	0	0
A0.1	175,832.94	1	1	0	0	0	0	0	0	0	0	0	0	0	0
A0.2	175,862.76	1	1	0	0	0	0	0	0	0	0	0	0	0	0
A0.3	177,015.94	1	1	0	0	0	0	0	0	0	0	0	0	0	0
A0.4	176,668.10	1	1	0	0	0	0	0	0	0	0	0	0	0	0
A0.5	176,597.12	1	1	0	0	0	0	0	0	0	0	0	0	0	0
A0.6	176,949.54	1	1	0	0	0	0	0	0	0	0	0	0	0	0
A0.7	176,977.76	1	1	0	0	0	0	0	0	0	0	0	0	0	0
A0.8	173,607.68	1	1	1	0	0	0	0	0	0	0	0	0	0	0
A0.9	176,304.38	1	1	0	0	0	0	0	0	0	0	0	0	0	0
A1.0	176,714.20	1	1	0	0	0	0	0	0	0	0	0	0	0	0
Period 1-60															
H2	382,300.52	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
H6	301,404.98	1	0	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
H12	270,101.32	1	0	0	0	0	0	0	0	0	0	0	0	0	0
S	256,682.32	1	1	0	0	0	0	0	0	0	0	0	0	0	0
A0.1	258,922.96	1	1	0	0	0	0	0	0	0	0	0	0	0	0
A0.2	256,399.06	1	1	0	0	0	0	0	0	0	0	0	0	0	0
A0.3	257,945.22	1	1	0	0	0	0	0	0	0	0	0	0	0	0
A0.4	258,247.02	1	1	0	0	0	0	0	0	0	0	0	0	0	0
A0.5	257,897.36	1	1	0	0	0	0	0	0	0	0	0	0	0	0
A0.6	257,297.44	1	1	0	0	0	0	0	0	0	0	0	0	0	0
A0.7	258,958.24	1	1	0	0	0	0	0	0	0	0	0	0	0	0
A0.8	253,894.80	1	1	0	0	0	0	0	0	0	0	0	0	0	0
A0.9	255,770.60	1	1	0	0	0	0	0	0	0	0	0	0	0	0
A1.0	259,527.94	1	1	0	0	0	0	0	0	0	0	0	0	0	0

Figure 5.3 displays BCa confidence intervals and means of aggregate costs obtained with the 2-period Look-Ahead method, 6-period Look-Ahead method, 12-period Look-Ahead method, Sarsa with the learning rate of 0.5 and Sarsa without GARCH(1,1) state variables using learning rates from 0.1 to 1.0. The Sarsa method without GARCH(1,1) model state variables is labeled "A#" where the suffix number indicates the learning rate. Table 5.6 shows a cross significance test summary of the Look-Ahead method, Sarsa, and Sarsa without latent state variables.

In addition to Sarsa, Rollout is used with 1, 5, 10 and 50 simulations with simulation horizons of 1, 3, 6 and 12 periods. Table 5.7 shows a summary of the cross significance tests of aggregate costs

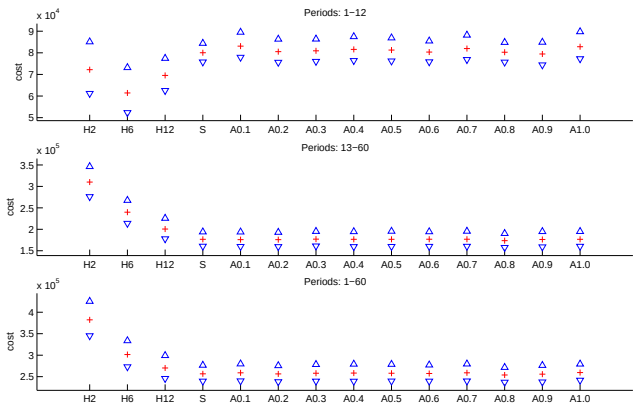

Figure 5.3: Average aggregate costs from Sarsa and Sarsa w/o z & σ^2; GARCH(1,1)

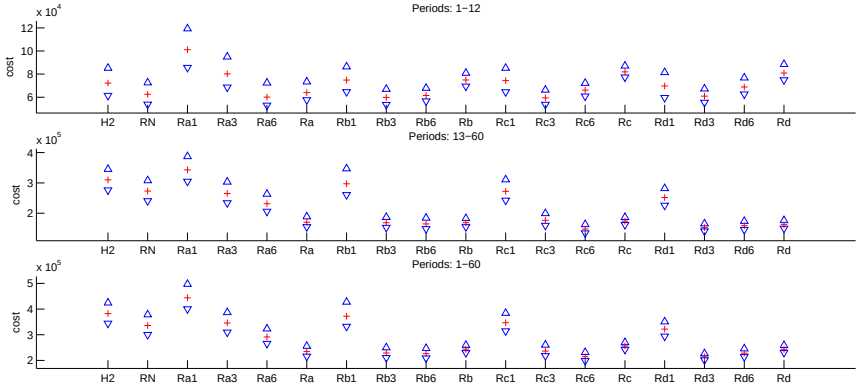

Figure 5.4: Average aggregate costs from Rollout; GARCH(1,1)

obtained with Look-Ahead and Rollout. Figure 5.4 shows BCa confidence intervals and means of aggregate costs obtained with 2-period Look-Ahead and Rollout. Rollout is labeled "Rx#" where the uncapitalized letter denotes the number of simulations: 'a' for $N = 1$, 'b' for $N = 5$, 'c' for $N = 10$, and 'd' for $N = 50$. The suffix number denotes the simulation horizon. The absence of this number indicates 12-period horizon. For example, the label Rb3 means Rollout with 5 simulations and a 3-period simulation horizon. The label Rd means Rollout with 50 simulations and 12-period simulation horizon.

It should be noted that the label RN, shown in Table 5.7 and Figure 5.4, indicates a Rollout method using an average projection. This Rollout method uses $N = 1$, $T = 1$, and an (s,S) policy that, instead of using a regular simulation, uses an average projection to evaluate cost. It is run only to be compared to a regular Rollout using $N = 1$ and $T = 1$ (labeled Ra1).

Table 5.7: Cross significance tests: Look-Ahead and Rollout; GARCH(1,1)

			Look-Ahead				Rollout															
		sample					$N=1$				$N=5$				$N=10$				$N=50$			
		mean	H2	H6	H12	RN	1	3	6	12	1	3	6	12	1	3	6	12	1	3	6	12
Period 1-12																						
H2		72,219.00	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1
H6		61,453.86	0	0	1	0	1	1	0	0	1	0	0	1	1	0	1	1	0	0	1	1
H12		69,539.66	0	-1	0	0	1	0	-1	0	0	-1	0	0	0	-1	0	1	0	0	0	1
RN		62,495.86	0	0	0	0	1	1	0	0	1	0	0	1	0	0	0	1	0	0	0	1
Rollout																						
N	T																					
1	1	101,167.74	-1	-1	-1	-1	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	-1	-1	-1	0
1	3	80,196.92	0	-1	0	-1	1	0	-1	0	0	-1	0	0	0	-1	0	0	0	0	0	0
1	6	60,048.68	0	0	1	0	1	1	0	0	1	0	0	1	1	0	1	1	0	0	1	1
1	12	63,978.64	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1
5	1	74,833.02	0	-1	0	-1	1	0	-1	0	0	-1	0	0	0	-1	0	0	0	0	0	0
5	3	59,826.46	0	0	1	0	1	1	0	0	1	0	0	1	1	0	0	1	0	0	1	1
5	6	61,492.16	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1
5	12	74,891.28	0	-1	0	-1	1	0	-1	-1	0	-1	-1	0	0	-1	-1	1	-1	-1	-1	0
10	1	74,359.86	0	-1	0	0	1	0	-1	0	0	-1	0	0	0	-1	0	0	0	0	0	0
10	3	59,364.14	0	0	1	0	1	1	0	0	1	0	0	1	1	0	1	1	0	0	1	1
10	6	66,114.00	0	-1	0	0	1	0	-1	0	0	0	0	1	0	-1	0	1	0	0	0	1
10	12	81,970.96	-1	-1	-1	-1	0	0	-1	-1	0	-1	-1	-1	0	-1	-1	0	-1	-1	-1	0
50	1	69,701.82	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1
50	3	60,928.78	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1
50	6	68,874.68	0	-1	0	0	1	0	-1	0	0	-1	0	1	0	-1	0	1	0	0	0	1
50	12	80,876.42	-1	-1	-1	-1	0	0	-1	-1	0	-1	-1	0	0	-1	-1	0	-1	-1	-1	0
Period 13-60																						
H2		310,081.52	0	-1	-1	0	0	-1	-1	-1	0	-1	-1	-1	0	-1	-1	-1	-1	-1	-1	-1
H6		239,951.12	1	0	-1	0	1	0	0	-1	1	-1	-1	-1	0	-1	-1	-1	0	-1	-1	-1
H12		200,561.66	1	1	0	1	1	1	0	-1	1	-1	-1	-1	1	0	-1	0	1	-1	-1	-1
RN		273,276.80	0	0	-1	0	1	0	-1	-1	0	-1	-1	-1	0	-1	-1	-1	0	-1	-1	-1
Rollout																						
N	T																					
1	1	342,712.88	0	-1	-1	-1	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
1	3	265,272.58	1	0	-1	0	1	0	0	-1	0	-1	-1	-1	0	-1	-1	-1	0	-1	-1	-1
1	6	231,378.82	1	0	0	1	1	0	0	-1	1	-1	-1	-1	0	-1	-1	-1	0	-1	-1	-1
1	12	170,396.46	1	1	1	1	1	1	1	0	1	0	0	0	1	0	-1	0	1	0	0	0
5	1	297,242.04	0	-1	-1	0	1	0	-1	-1	0	-1	-1	-1	0	-1	-1	-1	-1	-1	-1	-1
5	3	168,878.42	1	1	1	1	1	1	1	0	1	0	0	0	1	0	-1	0	1	0	0	0
5	6	164,868.72	1	1	1	1	1	1	1	0	1	0	0	0	1	0	0	0	1	0	0	0
5	12	168,837.32	1	1	1	1	1	1	1	0	1	0	0	0	1	0	-1	0	1	-1	0	0
10	1	272,222.48	0	0	-1	0	1	0	0	-1	0	-1	-1	-1	0	-1	-1	-1	0	-1	-1	-1
10	3	176,742.46	1	1	0	1	1	1	1	0	1	0	0	0	1	0	-1	0	1	-1	0	0
10	6	148,003.02	1	1	1	1	1	1	1	1	1	1	0	1	1	1	0	1	1	0	0	0
10	12	174,005.32	1	1	0	1	1	1	1	0	1	0	0	0	1	0	-1	0	1	-1	-1	0
50	1	251,982.76	1	0	-1	0	1	0	0	-1	1	-1	-1	-1	0	-1	-1	-1	0	-1	-1	-1
50	3	152,749.60	1	1	1	1	1	1	1	0	1	0	0	1	1	1	0	1	1	0	0	0
50	6	158,996.16	1	1	1	1	1	1	1	0	1	0	0	0	1	0	0	1	1	0	0	0
50	12	162,406.70	1	1	1	1	1	1	1	0	1	0	0	0	1	0	0	0	1	0	0	0
Period 1-60																						
H2		382,300.52	0	-1	-1	0	0	0	-1	-1	0	-1	-1	-1	0	-1	-1	-1	-1	-1	-1	-1
H6		301,404.98	1	0	0	0	1	1	0	-1	1	-1	-1	-1	1	-1	-1	-1	0	-1	-1	-1
H12		270,101.32	1	0	0	1	1	1	0	-1	1	-1	-1	0	1	-1	-1	0	1	-1	-1	-1
RN		335,772.66	0	0	-1	0	1	0	0	-1	0	-1	-1	-1	0	-1	-1	-1	0	-1	-1	-1
Rollout																						
N	T																					
1	1	443,880.62	0	-1	-1	-1	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
1	3	345,469.50	0	-1	-1	0	1	0	-1	-1	0	-1	-1	-1	0	-1	-1	-1	0	-1	-1	-1
1	6	291,427.50	1	0	0	0	1	1	0	-1	1	-1	-1	-1	1	-1	-1	0	1	-1	-1	-1
1	12	234,375.10	1	1	1	1	1	1	1	0	1	0	0	0	1	0	0	1	1	-1	0	0
5	1	372,075.06	0	-1	-1	0	1	0	-1	-1	0	-1	-1	-1	0	-1	-1	-1	0	-1	-1	-1
5	3	228,704.88	1	1	1	1	1	1	1	0	1	0	0	1	1	0	0	1	1	0	0	1
5	6	226,360.88	1	1	1	1	1	1	1	0	1	0	0	1	1	0	0	1	1	0	0	1
5	12	243,728.60	1	1	0	1	1	1	1	0	1	-1	-1	0	1	0	-1	0	1	-1	0	0
10	1	346,582.34	0	-1	-1	0	1	0	-1	-1	0	-1	-1	-1	0	-1	-1	-1	0	-1	-1	-1
10	3	236,106.60	1	1	1	1	1	1	1	0	1	0	0	0	1	1	0	0	1	1	0	0
10	6	214,117.02	1	1	1	1	1	1	1	0	1	0	0	1	1	0	0	1	1	0	0	1
10	12	255,976.28	1	1	0	1	1	1	0	-1	1	-1	-1	0	1	-1	-1	0	1	-1	-1	0
50	1	321,684.58	1	0	-1	0	1	0	-1	-1	0	-1	-1	-1	0	-1	-1	-1	0	-1	-1	-1
50	3	213,678.38	1	1	1	1	1	1	1	1	1	0	0	1	1	0	0	1	1	0	0	1
50	6	227,870.84	1	1	1	1	1	1	1	0	1	0	0	0	1	0	0	1	1	0	0	0
50	12	243,283.12	1	1	1	1	1	1	1	0	1	-1	-1	0	1	0	-1	0	1	-1	0	0

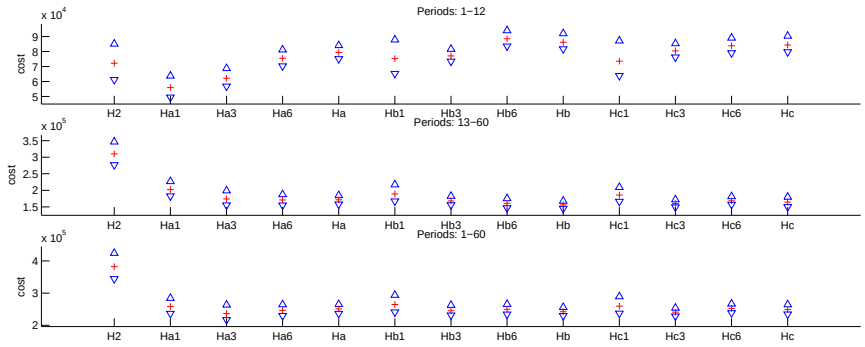

Figure 5.5: Average aggregate costs from HO; GARCH(1,1)

Figure 5.5 shows BCa confidence intervals and means of aggregate costs obtained with 2-period Look-Ahead and HO. The HO method is labeled "Hx#", where the uncaptalized letter indicates the number of simulations and the suffix number indicates the duration of the simulation horizon. Due to the high computation requirement, HO is used with $N = \{1, 5, 10\}$ (labeled Ha, Hb, Hc) and $T = \{1, 3, 6, 12\}$ (labeled 1, 3, 6 and the absence of number respectively). Table 5.8 shows a summary of the cross significance tests of aggregate costs obtained with Look-Ahead and HO.

Using HO with different parameters does not significantly affect inventory cost. For simplicity of presentation HO with $N = 1$ and $T = 3$ is selected to represent HO compared to other methods as shown in Figure 5.6 and Table 5.9.

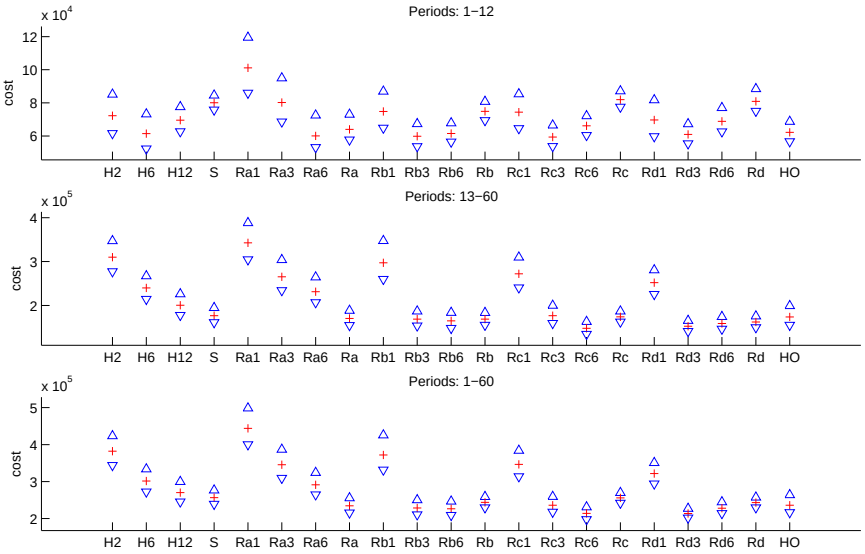

Figure 5.6: Average aggregate costs from different methods; GARCH(1,1)

Table 5.8: Cross significance tests: Look-Ahead and HO; GARCH(1,1)

			Look-Ahead			Hindsight Optimization											
		sample				$N=1$				$N=5$				$N=10$			
treatment		mean	H2	H6	H12	1	3	6	12	1	3	6	12	1	3	6	12
Period 1-12																	
H2		72,219.00	0	0	0	0	0	0	1	0	0	1	1	0	1	1	1
H6		61,453.86	0	0	1	0	0	1	1	1	1	1	1	1	1	1	1
H12		69,539.66	0	-1	0	-1	0	0	1	0	1	1	1	0	1	1	1
HO																	
N	T																
1	1	55,917.04	0	0	1	0	1	1	1	1	1	1	1	1	1	1	1
1	3	62,163.54	0	0	0	-1	0	1	1	0	1	1	1	0	1	1	1
1	6	75,479.78	0	-1	0	-1	-1	0	0	-1	0	1	1	0	0	1	1
1	12	79,341.64	-1	-1	-1	-1	-1	0	0	-1	0	1	1	-1	0	0	0
5	1	75,307.76	0	-1	0	-1	0	1	1	0	1	1	1	0	1	1	1
5	3	77,061.54	0	-1	-1	-1	-1	0	0	-1	0	1	1	-1	0	1	1
5	6	88,551.20	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	0	-1	-1	0	0
5	12	86,208.66	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	0	-1	-1	0	0
10	1	73,541.80	0	-1	0	-1	0	0	1	0	1	1	1	0	1	1	1
10	3	80,412.50	-1	-1	-1	-1	-1	0	0	-1	0	1	1	-1	0	0	0
10	6	83,870.92	-1	-1	-1	-1	-1	-1	0	-1	-1	0	0	-1	0	0	0
10	12	84,325.66	-1	-1	-1	-1	-1	-1	0	-1	-1	0	0	-1	0	0	0
Period 13-60																	
H2		310,081.52	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
H6		239,951.12	1	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
H12		200,561.66	1	1	0	0	0	-1	-1	0	-1	-1	-1	0	-1	-1	-1
HO																	
N	T																
1	1	202,294.48	1	1	0	0	-1	-1	-1	0	-1	-1	-1	0	-1	-1	-1
1	3	173,871.12	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0
1	6	170,902.52	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
1	12	171,478.82	1	1	1	1	0	0	0	0	0	0	-1	0	0	0	0
5	1	189,142.30	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
5	3	168,023.52	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
5	6	160,976.50	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
5	12	155,991.80	1	1	1	1	0	0	1	0	0	0	0	0	0	0	0
10	1	185,780.54	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
10	3	160,253.60	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
10	6	168,517.98	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
10	12	164,548.96	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
Period 1-60																	
H2		382,300.52	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
H6		301,404.98	1	0	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
H12		270,101.32	1	0	0	0	-1	0	0	0	0	0	-1	0	-1	0	0
HO																	
N	T																
1	1	258,211.52	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
1	3	236,034.66	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
1	6	246,382.30	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
1	12	250,820.46	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
5	1	264,450.06	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
5	3	245,085.06	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
5	6	249,527.70	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
5	12	242,200.46	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
10	1	259,322.34	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
10	3	240,666.10	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
10	6	252,388.90	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
10	12	248,874.62	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 5.9: Cross significance tests: Look-Ahead, Sarsa, Rollout and HO; GARCH(1,1)

		Look-Ahead			S	Rollout																HO
	sample					$N=1$				$N=5$				$N=10$				$N=50$				
	mean	H2	H6	H12		1	3	6	12	1	3	6	12	1	3	6	12	1	3	6	12	
Period 1-12																						
H2	72,219.00	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0
H6	61,453.86	0	0	1	1	1	1	0	0	1	0	0	1	1	0	1	1	0	0	1	1	0
H12	69,539.66	0	-1	0	1	1	0	-1	0	0	-1	0	0	0	-1	0	1	0	0	0	1	0
S	80,044.56	-1	-1	-1	0	0	0	-1	-1	0	-1	-1	0	0	-1	-1	0	-1	-1	-1	0	-1
Ra1	101,167.74	-1	-1	-1	0	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	-1	-1	-1	0	-1
Ra3	80,196.92	0	-1	0	0	1	0	-1	0	0	-1	0	0	0	-1	0	0	0	0	0	0	0
Ra6	60,048.68	0	0	1	1	1	1	0	0	1	0	0	1	1	0	1	1	0	0	1	1	0
Ra	63,978.64	0	0	0	1	1	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0
Rb1	74,833.02	0	-1	0	0	1	0	-1	0	0	-1	0	0	0	-1	0	0	0	0	0	0	0
Rb3	59,826.46	0	0	1	1	1	1	0	0	1	0	0	1	1	0	0	1	0	0	1	1	0
Rb6	61,492.16	0	0	0	1	1	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0
Rb	74,891.28	0	-1	0	0	1	0	-1	-1	0	-1	-1	0	0	-1	-1	1	-1	-1	-1	0	-1
Rc1	74,359.86	0	-1	0	0	1	0	-1	0	0	-1	0	0	0	-1	0	0	0	0	0	0	0
Rc3	59,364.14	0	0	1	1	1	1	0	0	1	0	0	1	1	0	1	1	0	0	1	1	0
Rc6	66,114.00	0	-1	0	1	1	0	-1	0	0	0	0	1	0	-1	0	1	0	0	0	1	0
Rc	81,970.96	-1	-1	-1	0	0	0	-1	-1	0	-1	-1	-1	0	-1	-1	0	-1	-1	-1	0	-1
Rd1	69,701.82	0	0	0	1	1	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0
Rd3	60,928.78	0	0	0	1	1	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0
Rd6	68,874.68	0	-1	0	1	1	0	-1	0	0	-1	0	1	0	-1	0	1	0	0	0	1	0
Rd	80,876.42	-1	-1	-1	0	0	0	-1	-1	0	-1	-1	0	0	-1	-1	0	-1	-1	-1	0	-1
HO	62,163.54	0	0	0	1	1	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0
Period 13-60																						
H2	310,081.52	0	-1	-1	-1	0	-1	-1	-1	0	-1	-1	-1	0	-1	-1	-1	-1	-1	-1	-1	-1
H6	239,951.12	1	0	-1	-1	1	0	0	-1	1	-1	-1	-1	0	-1	-1	-1	0	-1	-1	-1	-1
H12	200,561.66	1	1	0	0	1	1	0	-1	1	-1	-1	-1	1	0	-1	0	1	-1	-1	-1	0
S	176,637.76	1	1	0	0	1	1	1	0	1	0	0	0	1	0	-1	0	1	-1	0	0	0
Ra1	342,712.88	0	-1	-1	-1	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
Ra3	265,272.58	1	0	-1	-1	1	0	0	-1	0	-1	-1	-1	0	-1	-1	-1	0	-1	-1	-1	-1
Ra6	231,378.82	1	0	0	-1	1	0	0	-1	1	-1	-1	-1	0	-1	-1	-1	0	-1	-1	-1	-1
Ra	170,396.46	1	1	1	0	1	1	1	0	1	0	0	0	1	0	-1	0	1	0	0	0	0
Rb1	297,242.04	0	-1	-1	-1	1	0	-1	-1	0	-1	-1	-1	0	-1	-1	-1	-1	-1	-1	-1	-1
Rb3	168,878.42	1	1	1	0	1	1	1	0	1	0	0	0	1	0	-1	0	1	0	0	0	0
Rb6	164,868.72	1	1	1	0	1	1	1	0	1	0	0	0	1	0	0	0	1	0	0	0	0
Rb	168,837.32	1	1	1	0	1	1	1	0	1	0	0	0	1	0	-1	0	1	-1	0	0	0
Rc1	272,222.48	0	0	-1	-1	1	0	0	-1	0	-1	-1	-1	0	-1	-1	-1	0	-1	-1	-1	-1
Rc3	176,742.46	1	1	0	0	1	1	1	0	1	0	0	0	1	0	-1	0	1	-1	0	0	0
Rc6	148,003.02	1	1	1	1	1	1	1	1	1	1	0	1	1	1	0	1	1	0	0	0	0
Rc	174,005.32	1	1	0	0	1	1	1	0	1	0	0	0	1	0	-1	0	1	-1	-1	0	0
Rd1	251,982.76	1	0	-1	-1	1	0	0	-1	1	-1	-1	-1	0	-1	-1	-1	0	-1	-1	-1	-1
Rd3	152,749.60	1	1	1	1	1	1	1	0	1	0	0	1	1	1	0	1	1	0	0	0	0
Rd6	158,996.16	1	1	1	0	1	1	1	0	1	0	0	0	1	0	0	1	1	0	0	0	0
Rd	162,406.70	1	1	1	0	1	1	1	0	1	0	0	0	1	0	0	0	1	0	0	0	0
HO	173,871.12	1	1	0	0	1	1	1	0	1	0	0	0	1	0	0	0	1	0	0	0	0
Period 1-60																						
H2	382,300.52	0	-1	-1	-1	0	0	-1	-1	0	-1	-1	-1	0	-1	-1	-1	-1	-1	-1	-1	-1
H6	301,404.98	1	0	0	-1	1	1	0	-1	1	-1	-1	-1	1	-1	-1	-1	0	-1	-1	-1	-1
H12	270,101.32	1	0	0	0	1	1	0	-1	1	-1	-1	0	1	-1	-1	0	1	-1	-1	-1	-1
Sarsa	256,682.32	1	1	0	0	1	1	0	0	1	-1	-1	0	1	-1	-1	0	1	-1	-1	0	0
Ra1	443,880.62	0	-1	-1	-1	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
Ra3	345,469.50	0	-1	-1	-1	1	0	-1	-1	0	-1	-1	-1	0	-1	-1	-1	0	-1	-1	-1	-1
Ra6	291,427.50	1	0	0	0	1	1	0	-1	1	-1	-1	-1	1	-1	-1	0	1	-1	-1	-1	-1
Ra	234,375.10	1	1	1	0	1	1	1	0	1	0	0	0	1	0	0	1	1	-1	0	0	0
Rb1	372,075.06	0	-1	-1	-1	1	0	-1	-1	0	-1	-1	-1	0	-1	-1	-1	0	-1	-1	-1	-1
Rb3	228,704.88	1	1	1	1	1	1	1	0	1	0	0	1	1	0	0	1	1	0	0	1	0
Rb6	226,360.88	1	1	1	1	1	1	1	0	1	0	0	1	1	0	0	1	1	0	0	1	0
Rb	243,728.60	1	1	0	0	1	1	1	0	1	-1	-1	0	1	0	-1	0	1	-1	0	0	0
Rc1	346,582.34	0	-1	-1	-1	1	0	-1	-1	0	-1	-1	-1	0	-1	-1	-1	0	-1	-1	-1	-1
Rc3	236,106.60	1	1	1	1	1	1	1	0	1	0	0	0	1	0	0	1	1	0	0	0	0
Rc6	214,117.02	1	1	1	1	1	1	1	0	1	0	0	1	1	0	0	1	1	0	0	1	0
Rc	255,976.28	1	1	0	0	1	1	0	-1	1	-1	-1	0	1	-1	-1	0	1	-1	-1	0	0
Rd1	321,684.58	1	0	-1	-1	1	0	-1	-1	0	-1	-1	-1	0	-1	-1	-1	0	-1	-1	-1	-1
Rd3	213,678.38	1	1	1	1	1	1	1	1	1	0	0	1	1	0	0	1	1	0	0	1	0
Rd6	227,870.84	1	1	1	1	1	1	1	0	1	0	0	0	1	0	0	1	1	0	0	0	0
Rd	243,283.12	1	1	1	0	1	1	1	0	1	-1	-1	0	1	0	-1	0	1	-1	0	0	0
HO	236,034.66	1	1	1	0	1	1	1	0	1	0	0	0	1	0	0	0	1	0	0	0	0

Figure 5.6 shows the BCa confidence intervals and means of aggregate costs obtained with Look-Ahead, Sarsa (labeled S), Rollout (labeled Rx#), and HO (labeled HO). Table 5.9 shows a summary of cross significance tests of aggregate costs obtained with Look-Ahead, Sarsa, Rollout and HO.

Regarding computation time expended, the experiments of Sarsa without the GARCH(1,1) state variables required only around 1 to 10 percent of the the time expended for Sarsa with all state variables included. HO experiments required approximately 4 to 60 times the computation time required for Rollout with equivalent settings of N and T. While computation time required when using Sarsa depends mainly on the size of the RBF, time required for Rollout and HO depend on the number of simulations and simulation horizons. With an (s,S) policy as its base policy, Rollout with $N = 50$, $T = 12$, requires the most computation time compared to other settings, roughly the same amount of time as Sarsa with all state variables. Rollout with other settings required much less time than Sarsa. It should be noted that the timing information mentioned here is only a rough estimate, since experiments were not run in a controlled environment, rather in a multi-server/multi-user computing system with variable computational loads.

5.4 Discussions and Conclusions

Both with and without GARCH(1,1) state variables, Sarsa performs effectively when compared to Look-Ahead methods. Using learning rate values within the tested range ($\beta = \{0.1, 0.2, ..., 1.0\}$) does not significantly change aggregate costs. They all work equally well. Inventory cost performance using Sarsa without GARCH(1,1) state variables does not differ significantly from using Sarsa with all state variables included. This finding mitigates a concern about the effect of the latent variables introduced by the GARCH(1,1) model when applying a learning-based ADP. The explanation may be that these latent state variables do not have a direct effect on a transition cost. They only give extra information about uncertainty in the variance in the demand forecast. The robust nature of Sarsa with some information missing shown in our experiments confirms the conclusions made by Csáji and Monostori [34] that temporal difference learning can, to some extent, tolerate inaccurate information.

While the inventory cost performance is not significantly different, Sarsa without GARCH(1,1) state variables requires less computation time than Sarsa with all state variables. Sarsa, with all state variables, is used with RBF having 175,000 centers for a 6-dimension state-action space. Sarsa without GARCH(1,1) state variables is used with RBF having only 700 centers, for a 4-dimension state-action space. Using Sarsa with 700 RBF centers requires less than 1 percent of the

computation time to update weight[2] required by Sarsa with 175,000 RBF centers. The exploitation of reduced state representation is worth further investigation. Sutton and Barto [114] and Bertsekas and Tsitsiklis [15] discuss using feature extraction, which is a general concept of preparing ADP state variables by preprocessing the system raw state variables to extract some of the more important aspects of a state. Sutton and Barto [114] commented that feature extraction allows utilization of domain knowledge in a learning-based ADP.

Regarding to the performance of Rollout, Rollout yields significantly lower aggregate costs than the Look-Ahead method when Rollout is used with an adequate number of simulations with a long enough simulation horizon. Rollout with one simulation ($N = 1$) shows a monotonic trend in relation to average total cost and the simulation horizon. This monotonic trend is not observed when using Rollout with $N > 1$. Figure 5.7 displays plots of average on-site inventory levels (mean x), average period costs (mean cost) and the maximum period costs (max cost) for the Rollout simulation.

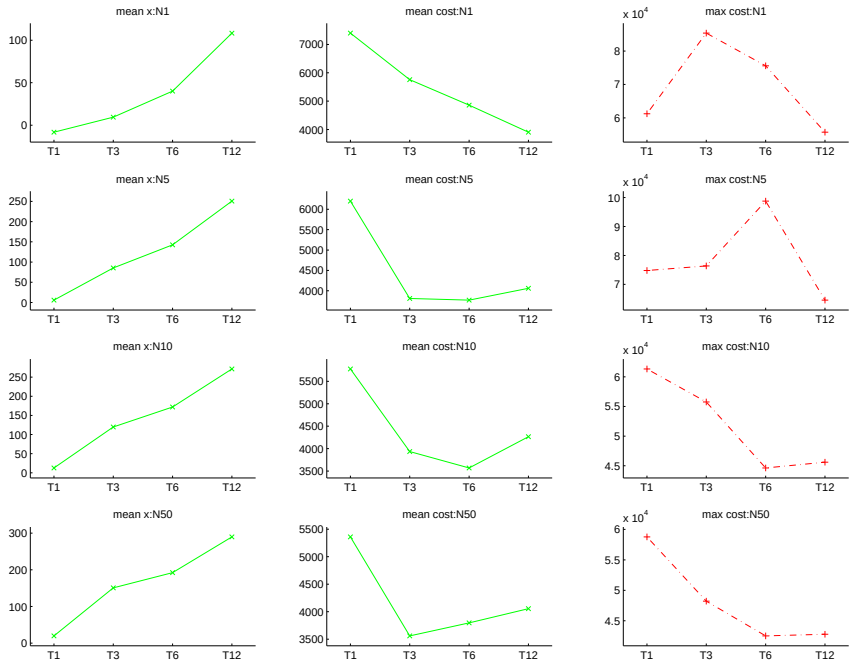

Figure 5.7: Average inventory and average and maximum costs from Rollout

[2] The actual total time expended when using Sarsa without GARCH(1,1) state variables is around 1 to 10 percent of the amount of time expended when using Sarsa with all state variables. As mentioned earlier, time expended for each experiment is only a rough estimate of computation required by each method. There were many uncontrollable factors including different server computing capabilities and loads. In addition, this total time includes all activities in experiments including accessing disk storage to save data.

For every N, the plots show an upward trend of on-site inventory levels with increasing T. The downward trend is more apparent in average costs when using the lower N and in the maximum cost when using the higher N. The explanation for this may lie in the role of parameters N and T in approximating state-action cost. With a few simulations (small N) Rollout may not have seen many rare demand surges, so a longer horizon (large T) helps promote a more representative averaging effect, which results in a better decisions for the average case. With more simulations and a longer horizon (large N and T) Rollout may have a better chance to see rare demand surge, which, even though rare, can substantially effect the total cost. Such a surge may drive Rollout to choose a more conservative action, such as stocking a higher inventory level, to safeguard against such a negative effect. Therefore, it results in higher average cost, but lower maximum cost. Figure 5.8 displays empirical CDF of on-site inventory and period cost.

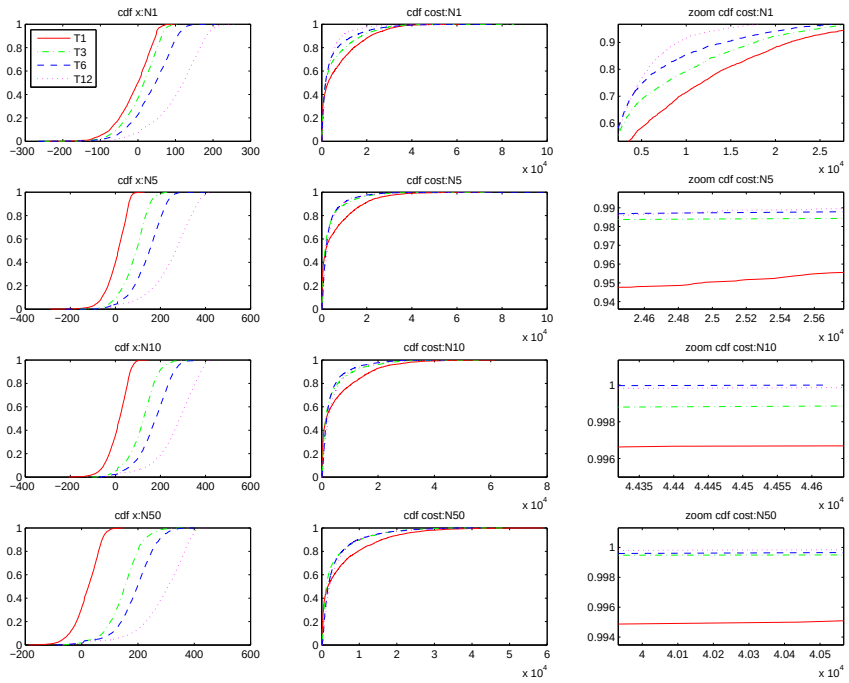

Figure 5.8: CDF Plot of inventory and single-period cost for each Rollout setting.

It shows that for most cases a longer simulation horizon the CDF approaches 1.0 more quickly. This scenario means that the likelihood of having high costs is lower, when using a longer simulation horizon. This supports the explanation that using a larger T make Rollout choose actions more conservatively.

Regarding the number of simulations used in Rollout, Table 5.10 displays the average, standard deviation, minimum total costs and maximum total costs in the columns labeled 'mean', 'std', 'min' and 'max', respectively. A summary of cross significance tests for Rollout is also provided in Table 5.10. Note that the table has a rearranged order of the presentation of simulation horizon and number of simulation(s).

Table 5.10: Rollout numbers of simulations and total costs

Rollout						Rollout															
						T = 1				T = 3				T = 6				T = 12			
N	T	mean	std	min	max	1	5	10	50	1	5	10	50	1	5	10	50	1	5	10	50
1	1	443,881	175,337	176,611	889,314	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
5	1	372,075	171,653	172,034	930,075	1	0	0	0	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
10	1	346,582	127,031	137,375	727,967	1	0	0	0	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
50	1	321,685	102,817	118,110	553,761	1	0	0	0	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
1	3	345,470	138,251	99,055	734,338	1	0	0	0	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
5	3	228,705	71,590	117,153	409,686	1	1	1	1	1	0	0	0	1	0	0	0	0	1	1	1
10	3	236,107	76,082	113,408	538,262	1	1	1	1	1	0	0	0	1	0	0	0	0	0	1	0
50	3	213,678	45,829	133,999	343,796	1	1	1	1	1	0	0	0	1	0	0	0	1	1	1	1
1	6	291,428	106,443	78,198	624,453	1	1	1	1	1	-1	-1	-1	0	-1	-1	-1	-1	-1	0	-1
5	6	226,361	69,516	132,268	402,963	1	1	1	1	1	0	0	0	1	0	0	0	0	1	1	1
10	6	214,117	59,941	119,637	362,721	1	1	1	1	1	0	0	0	1	0	0	0	0	1	1	1
50	6	227,871	56,781	143,600	387,366	1	1	1	1	1	0	0	0	1	0	0	0	0	0	1	0
1	12	234,375	73,380	113,876	456,552	1	1	1	1	1	0	0	-1	1	0	0	0	0	0	1	0
5	12	243,729	54,109	157,341	373,204	1	1	1	1	1	-1	0	-1	1	-1	-1	0	0	0	0	0
10	12	255,976	50,269	157,244	362,195	1	1	1	1	1	-1	-1	-1	0	-1	-1	-1	-1	0	0	0
50	12	243,283	51,551	157,651	348,886	1	1	1	1	1	-1	0	-1	1	-1	-1	0	0	0	0	0

Using Rollout with 5, 10 or 50 simulations does not significantly alter total costs over all values of the simulation horizon. When horizons of 1, 3 and 6 are used, Rollout with 5, 10 and 50 simulations results in significantly lower total costs than Rollout with one simulation. However, Rollout with one simulation has a lower total cost than Rollout with 5, 10 and 50 simulations when using a 12-period simulation horizon. The explanation for this may lie in the failure to capture variation in cost performance. Although Rollout with one simulation has the lowest average cost, the variation in its cost performance shows up in a high standard deviation (36% higher than the second largest value), the lowest minimum cost and the highest maximum cost. In addition, Rollout with one simulation has the greatest standard deviations for all lengths of the simulation horizon ($T = 1$, 3, 6 and 12). The greater number of simulations provides an improved consistency of cost performance, as indicated in the values of standard deviation that decrease with more simulations (with a few exceptions that may be caused by variations of a stochastic nature of the problem). It should be noted that the non-monotonically downward trend in maximum costs when using a higher number of simulations is evident.

An alternative to Rollout, HO is another simulation-based ADP. HO performs significantly better than the Look-Ahead method. The two parameters, N and T, of HO are not as critical as in Rollout. In addition, HO does not require extra decisions in its set up. The non-criticality of its parameter

values makes HO an attractive alternative to providing a good measure of how a simulation-based ADP method performs without a choice of a base policy or criticality of choices of its parameter values.

When Sarsa, Rollout and HO methods are compared, Rollout (with $N = 5$ to 50 and $T = 3$ to 6) performs significantly better than the 12-period Look-Ahead method and Sarsa over a total of 60 periods. Rollout (with $N = 10$, $T = 6$ and $N = 50$, $T = 3$) can outperform Sarsa significantly in the last 48 periods. Rollout and HO performance appear to be equivalent. Although not confirmed with significance tests, Rollout (with $N = 10$, $T = 6$ and $N = 50$, $T = 3$) has a noticeable lower average total cost than HO. Sarsa, Rollout and HO can be used for an inventory problem with AR1/GARCH(1,1) demand and can outperform a simple controller such as the Look-Ahead method. Sarsa can be employed when a model of the problem is not available. Using Sarsa without latent state variables introduced by the GARCH(1,1) model is seen to not deteriorate inventory cost performance. On the other hand, leaving out the latent state variables is beneficial because the computation times are reduced. When there is a model of the problem, Rollout is recommended for better performance. HO can deliver similar performance to Rollout. Using HO does not involve determining critical parameter values, but HO requires substantially longer computation times.

CHAPTER 6

CONCLUSIONS

6.1 Summary of Research Issues

Inventory management is one of the major functions of business. A well-managed inventory can help business stay competitive by keeping its cash flow at a controllable level. Since inventory management is an essential operation in many different businesses, inventory problems appear in various forms and the forms often change over time. Recently, Zhang [130] found evidence of temporal demand heteroscedasticity, GARCH(1,1), in inventory data and showed that there are significant costs when GARCH(1,1) has not been accounted for. He proposed an analytical solution to the problem. However, his analytical approach is too problem specific. A slightly change in the structure of the problem requires rigorous reanalysis of the problem and redevelopment of the solution. However, Approximate Dynamic Programming (ADP) has been shown to have the generality and flexibility to overcome the shortcomings of this and many other analytical approaches. Accordingly, ADP has gained much attention in inventory management research.

The main research objective here is to design a more general ADP solution approach for practical inventory problems. This objective can be broken down into the following four research investigations corresponding to major issues of ADP applications.

6.1.1 Investigation of Function Approximation

One of the reasons ADP is gaining interest is its model-free property[1] associated with learning-based ADP. A learning-based ADP method uses updatable function approximation along with a learning strategy to determine how to control a process without needing assumptions about a demand distribution. A Radial Basis Function (RBF) and other approximation functions belonging to a linear family are recommended by Sutton and Barto [114] for learning-based ADP. According to Barreto and Anderson [9], RBF is one of the most widely-used choices for function approximation for most applications, but, as the time of this writing, it has not yet been applied to an inventory problem. RBF has three sets of parameters: centers, scales and weights. In order to operate RBF in a linear mode, RBF centers and scales are predefined. RBF weights will be updated online during the learning process. Billings et al. [17] commented that RBF is normally set up with a single scale and then Chen et al. [26]'s Orthogonal Least Square Learning Algorithm (OLS) is used to determine its centers. This conventional setup is developed for supervised learning applications where data is available before RBF is designed. Therefore, when the conventional RBF setup is used with ADP, initial data is required. However, RBF performance may deteriorate, when data acquired online is different from the initial data. For inventory problems, a restricted range of the state-action space

[1] A model-free property means that an inventory solution technique can work well without requires the knowledge of a demand distribution.

can be reasonably estimated. Therefore, it was possible to specify RBF centers based on this domain knowledge. How RBF should be set up, either with initial data or with prior knowledge, had not been explored previously.

6.1.2 Investigation of Learning Strategies

The second essential issue of learning-based ADP is a learning strategy. There are many learning strategies. (See Gosavi [49] for recent review of learning-based ADP.) Sarsa is a well-known learning-based ADP method. It uses a one-step temporal difference learning (TD0) technique. Sarsa(λ) is Sarsa bootstrapped by Eligibility Trace. Eligibility Trace is intended to speed up the learning process in TD(0). An Eligibility Trace technique has been applied successfully in many other applications, including studies of Tesauro [116] and Gelly and Silver [43], but it has never been investigated for an inventory problem before. The use of Residual Gradient guarantees convergence of the algorithm, but associated slow learning speed have been reported by Baird [7]. To improve Residual Gradient learning speed, Direct Credit Back was developed here using a bootstrapping technique similar to Eligibility Trace. An explicit evaluation of these learning ADP methods for different kinds of inventory problems had not been developed previously.

6.1.3 Investigation of the Effect of GARCH Variables

Recently, the GARCH(1,1) model has been found in inventory data. It has been shown to be significant in inventory control performance by Zhang [130]. The GARCH(1,1) model introduces latent state variables. These latent state variables will be inadvertently left out if the GARCH(1,1) model is not accounted for. This has posed a challenge to a model-free property of a learning-based ADP method. How a learning-based ADP method performs under the absence of these latent state variables had not been studied previously.

6.1.4 Investigation of Simulation-based Methods

In addition to a learning-based ADP method, there is a simulation-based ADP method. A simulation-based ADP method is an approach intermediate between an analytical approach and a learning-based ADP method. An analytical approach requires a model of a problem and a rigorous development of an analytical solution. A learning-based ADP method requires minimum knowledge of the problem. A simulation-based ADP method requires a model of a problem and, rather than using a rigorous analysis, it uses a simulation to provide information for assisting action selection. Rollout, one of the simulation-based ADP methods investigated here, uses the model and a pre-specified base policy in its Monte Carlo simulator to evaluate the consequence of actions. Hindsight

Optimization (HO) uses the hindsight simulator to provide approximate costs. Though it has been used in other applications, Rollout had been investigated for inventory problems in only one study. HO had not been studied for an inventory problem previously. Also, how these simulation-based ADP methods perform compared to a learning-based ADP method had not been investigated previously.

6.2 Summary of Research Approach

To answer the research questions above, simulated-based experiments are conducted. Each inventory controller is run for 50 replications of each of 60 time-indeterminate periods. An aggregate cost is used as a performance indicator. Conclusions are drawn mainly from statistical significance tests, either the T test or the Wilcoxon Rank Sum test, depending on degree of normality of the data.

6.2.1 Function Approximation

A learning-based ADP method has two major components: a learning strategy and a function approximation. In Chapter 3, a Radial Basis Function is investigated for a function approximation. First, to emphasize the benefit of RBF as an approximation function for a cost-to-go value, one-period Look-Ahead (H1) with an approximate cost-to-go is compared to H1 without a cost-to-go. Second, different RBF scales are investigated. The midpoint strategies (strategies for setting RBF scales) are experimentally investigated for midpoint parameters from 0.1 to 0.9. Third, different numbers of RBF centers are investigated. Evenly distributed structure RBFs are experimentally investigated for center gap sizes of 5, 10 and 15. Experiments for RBF scales and centers use a Look-Ahead method with approximate cost-to-go as an inventory controller. The Look-Ahead's cost-to-go is approximated with RBF and updated with TD(0). Use of this controller allows us to study the effect of a cost-to-go approximation more effectively when compared with regular Look-Ahead, which does not have an approximate cost-to-go. The inventory problem investigated in this Chapter is a zero leadtime problem with AR1 demand. This problem has a two-dimensional state-action space, which allows simple visualization.

6.2.2 Learning Strategies

In Chapter 4, learning strategies are investigated. A new development of Direct Credit Back, based on the Residual Gradient method and the Eligibility Trace technique, is discussed. The well-known Sarsa method, its bootstrapping extension Sarsa(λ), the guaranteed convergence Residual Gradient method and the method developed in this research, Direct Credit Back, are investigated in both zero and one-period leadtime problems. Running experiments with both zero and one-period

leadtime problems allows us to study the performance of each learning scheme in different leadtime settings. In a nonzero leadtime inventory problem, there is a partially delayed return, because the delivery leadtime affects only a part of its single-period cost. A holding or shortage cost is delayed according to a leadtime, but a replenishment cost is still immediately affected by an action.

6.2.3 The Effect of GARCH Variables and Simulation-based Methods

In Chapter 5, both learning- and simulation-based ADP methods are investigated. All methods are investigated using a one-period leadtime problem with AR1/GARCH(1,1) demand. This problem allows investigation of the ADP performances under the presence of the GARCH(1,1) model. Based on conclusions from Chapter 3 and 4, Sarsa is chosen as a learning-based ADP method to compare with simulation-based ADP methods, Rollout and HO. First, Sarsa, with all state variables, is compared to Sarsa without GARCH(1,1)'s latent state variables to study an effect of the missing variables. Then Sarsa, Rollout and HO are compared.

6.2.4 Research Results

The experimental results have shown the effectiveness of the evenly distributed structure of RBF centers for cost-to-go approximation. The density of RBF centers can be determined with Akaike Information Criteria (AIC) when sample data is available. RBF scales can be efficiently assigned with the half-midpoint strategy.

Comparing to Sarsa(λ) and Residual Gradient, our experiments show Sarsa to be the most suitable learning strategy for both zero- and one-period inventory problems. Eligibility Trace, represented by Sarsa(λ), did not show any potential to improve Sarsa performance in either problem. Although Direct Credit Back achieved lower average costs than Residual Gradient, the improvement is not confirmed by significance tests at a 0.05 significance level.

For a problem with the AR1/GARCH(1,1) demand, Sarsa performed indifferently either with or without the latent state variables. These results support the robustness of learning-based ADP and the adequacy of ADP's model-free property. Sarsa's learning rate value between 0.1 to 1.0 did not show a significantly different performance.

Simulation-based ADP Rollout showed significantly better performance than learning-based ADP Sarsa. Use of different Rollout parameters (the number of simulations N at a horizon T) produced significantly different performance. With $N > 1$, Rollout delivers a better average performance, but, with a very high N and a long T, Rollout shows conservative behavior. That is a higher inventory level with its accompanying slightly higher average cost was produced, but with a lower maximum cost. HO performance is very close to Rollout, but choices of its parameters are not as critical.

6.3 Discussion of Research Results

6.3.1 Function Approximation

To address the issue of function approximation the evenly distributed structure and the midpoint strategy were used and were effective in setting up RBF. Setting up RBF with these strategies is a more general systematic approach as compared to other choices made in previous studies of ADP application to inventory problems. A Look-up table, used by Jiang and Sheng [64], Kim et al. [71], Kwon et al. [75] and Kim et al. [70], is the most widely-used implementation for an approximate cost-to-go in ADP inventory control. A Look-up table is simple to implement, but it can not be scaled up to handle large state-action space. An aggregation, used by Giannoccaro and Pontrandolfo [47] and Chaharsooghi et al. [25], is an extension of a Look-up table. It is simple to implement, but its parameter aggregation size is domain specific. Setting this size too large causes a too coarse discretization and low approximation quality. Setting this size too small causes a slow learning process. A smaller aggregation size causes a requirement of more entries to cover the same state-action space. Since the major learning scheme TD(0) is very localized, the more entries an aggregation has, the more well-distributed samples are required to allow the learning process to converge. RBF can be conceived as a way to interpolate coarse size aggregations, where RBF centers are entry points and RBF scales are interpolation parameters. In addition, RBF's ability to deliver continuous approximation makes it more suitable for a wider range of applications, including real-time feedback control. (See Balakrishnan et al. [8] for a review of ADP applications to feedback controllers.) A linear combination of features, used by Van Roy et al. [118], is a good alternative for an approximate cost-to-go. Its drawback would only be that it is very problem specific and there is as yet no systematic approach to it. A Multilayer Perceptron Neural Network (MLP), used by Van Roy et al. [118] and Shervais et al. [108], requires high technical and domain expertise to tune it for learning-based ADP. RBF, with the setting strategies developed in our research, have filled in a gap in the current spectrum of approximate cost-to-go choices. RBF provides a smoother function approximation than an aggregation and our setting strategies provide a more systematic setup than either a linear combination of features or MLP. Although conclusions here are drawn from experiments with ADP methods for inventory problems, RBF and the set up strategies, the evenly distributed structure and the midpoint strategy developed here, are not limited to ADP or to inventory problems. They can be used for other applications as well.

6.3.2 Learning Strategies

For the issue of a learning strategy, Sarsa is found to be more suitable for an inventory problem than Sarsa(λ), Residual Gradient and Direct Credit Back. Sarsa(λ) is expected to be an improve-

ment over Sarsa for a delayed-return problem. Even though a leadtime may be nonzero, an inventory problem is not a delayed-return problem. Although a realization of a holding or penalty cost is delayed according to a leadtime, a replenishment cost and an in-transit inventory are observed immediately after an action is taken. Therefore Sarsa(λ) could not provide an improvement over the average performance of Sarsa. Sarsa(λ) is an implementation of Eligibility Trace and Sarsa is an implementation of Temporal Difference learning, TD(0). Eligibility Trace is a bootstrap learning process in TD(0) with a state-action trajectory. As illustrated in Sutton and Barto [114], its parameter λ controls how much bootstrapping there will be. When λ is closer to one, the behavior of the learning process will be closer to a Monte Carlo method, which fully bootstraps state-action cost approximation with the whole trajectory. When λ is closer to zero, the behavior of the learning process will be closer to TD(0), which corrects the state-action cost approximation only based on the most recent state-action pair.

Although a bootstrapping technique in Sarsa(λ) did not show an improvement on average performance, it was determined that it may help safeguard against a worst-case scenario. The investigation in Chapter 4 shows the relations between a high value of bootstrapping parameter λ and conservative actions. These relations are similar to high values of N and T in the Rollout method as discussed in Chapter 5. Both high value of λ in Sarsa(λ) and high values of N and T in Rollout cause a high degree of bootstrapping in a corresponding method. The relations between a high degree of bootstrapping and conservative behavior may reflect the fact that an approximate state-action cost is closer to an expected value. Since a consequence of an inventory shortage is usually substantially worse than a cost of an overstock, this may lead ADP to associate a riskier action to a higher expected cost and eventually make ADP behave conservatively. In addition to decision and control applications, temporal difference learning has been utilized to model and study brain function, as discussed by Schultz et al. [105], Seymour et al. [106] and Dayan [37]. The found insights into the stronger relationship between bootstrapping experience and conservative behavior may help neuroscientists understand how we make decisions in relating experience to risk taking preference.

Motivated by the bootstrapping approach, we developed Direct Credit Back to improve performance of the Residual Gradient method. Direct Credit Back uses bootstrapping to speed up the learning process in the Residual Gradient method. The experimental results indicated that Direct Credit Back had lower average aggregate costs than the Residual Gradient method. This suggests a potential for improvement, but the limited range of problems investigated here may not fully have identified bootstrapping potential. Within the scope of delayed-return problems, Direct Credit Back may have a better chance to show an advantage over the Residual Gradient method.

6.3.3 The Effect of GARCH variables

Relative to the issue of GARCH(1,1)'s latent state variables, Sarsa's ability to work well with some information missing is in agreement with Csáji and Monostori [34] in that TD(0) can tolerate some degree of inaccurate information. The explanation may be that these latent state variables do not have a direct effect on the cost function. They only give extra information about the uncertainty of the variance in the estimate of the demand. The results shown here reaffirm that Sarsa is a model-free solution and assuage Zhang [130]'s concern about the significance of the GARCH(1,1) model.

6.3.4 Simulation-based Methods

Despite receiving less attention, Rollout was shown to be the best option when a model of the problem is available. Rollout has been investigated in Choi et al. [30]. Choi et al. [30] used heuristic search over sets of pre-specified (s,S) policies as Rollout's base policy. The heuristic search is computationally time consuming compared to the EOQ parameterized (s,S) policy used in our study. Rollout parameters, N and T, were shown to be critical to the overall performance, but it is safe to set them to high values, e.g., $N = 10$; $T = 6$, to produce an averaging effect. If N and T happen to be set too high, Rollout just takes more computation and behaves more conservatively.

Flexibility of a learning-based ADP method may be a factor that would make it more appealing than a simulation-based ADP method. However, most inventory problems can be easily formulated given known uncertainty in demand, leadtime and production capacity. In addition, there are many available forecasting techniques that provide a fair estimate of uncertainty. A forecasting technique and an update scheme can be incorporated into a simulation-based method to provide estimates of the unknown values and their variance for a simulation. The estimates can be updated periodically, rather than preset as in conventional simulation-based ADP. Then, given the estimates, the simulation can use a model of known system dynamics to simulate system transition and consequences. Having the notion of uncertainty separated from the known dynamics allows flexibility of implementation and increases adaptability of a simulation-based ADP solution. For example, when a cost structure changes, only a model of a known system dynamics will be updated. The demand forecast does not need to be revised.

Lastly, what we found in this study provides the bases for improved strategies to design an appropriate ADP method for inventory problems. The results help assuage the concern about learning-based ADP's performance in an environment with latent state variables. Furthermore it provides insights into relations among simulation parameters, behaviors and the performance of both Rollout and Hindsight Optimization. These findings can become building blocks for larger and more

complex applications. The greater understanding hopefully will help promote efficient inventory management and aid in the transfer of inventory research into practice.

6.4 Limitations of the Research

Powell [97] identified three issues in ADP. They are (1) step sizes, (2) balancing exploration and exploitation and (3) the evaluation of ADP solutions. Our experimental results show insignificant difference among step sizes (in some literature learning rates), between 0.1 and 1.0. However, it is possible that we may have passed a specific learning rate that would make Sarsa perform at its best and this could change the conclusions. Rollout and HO are simulation-based ADP, so they are not subjected to an issue of step sizes nor to a balance between exploration and exploitation.

Exploration of Sarsa and Sarsa(λ) in Chapter 4 is performed in the context of zero initialization of the cost-to-go values in a minimization problem and a heuristic search using simulated annealing. In addition to zero initialization, exploration of Sarsa in Chapter 5 is done more explicitly with the addition of noise to the optimal action which was found by an exhaustive search. Both approaches were tested to assure that they resulted in reliable and comparable inventory controllers. In addition, there are some differences in code implementation over the period of the research. The differences include restructuring the code to be compliant with the then new version `7.7.0` of Matlab and to be more computationally efficient, as more complicated problems were investigated. These implementation differences were tested to ensure consistency in the results.

It should be noted that the optimal action search, $a^* = \arg\min_{a \in A(s)} f(s,a)$ where $f(s,a)$ is a cost function, is done over a feasible set of actions, $A(s)$. A feasible set is obtained from a set of only the actions that are projected to an operating condition. Much too low replenishment that could most certainly lead to inventory shortage and much too high replenishment that could most certainly lead to excess inventory capacity are excluded from the feasible set. This mechanism of obtaining $A(s)$, used for all methods, helps make fair comparisons among different methods. It also helps stabilize learning-based ADP control in its early stages and allows faster convergence for learning-based ADP methods. However, since $A(s)$ is obtained from an average projection based on an inventory model, this makes Sarsa not operate under a completely model-free setting. This issue should not affect the conclusions on robustness of Sarsa over GARCH(1,1)'s latent state variables, because two GARCH(1,1)'s variables, z and σ^2, have only an effect on variance of demand and this variance is discarded in an average projection. Therefore, awareness or knowledge of GARCH(1,1)'s variables does not affect feasible sets used in our experiments. Both affirmation of Sarsa's robustness against GARCH(1,1) latent state variables and its contradiction to Zhang [130]'s significance of GARCH(1,1) in inventory management are still valid.

Our conclusions are drawn from experimental results comparing different ADP methods to a benchmark Look-Ahead method as well as to each other. To evaluate inventory control, Simchi-Levi et al. [111] identified three approaches. They are empirical comparison, worst-case analysis and average-case analysis, but Simchi-Levi et al. [111] observed that worst-case or average-case analysis may be technically very difficult. From an ADP research perspective, Powell [97] identified the comparison to a benchmark as a significant approach to evaluate ADP methods. Therefore, even though this may not be the best way to evaluate the method theoretically, empirical comparison with a benchmark may currently be the best practical way to evaluate ADP, as well as other inventory management solution approaches.

In order to draw conclusions from a variety of results, we used statistical significance tests as a main analysis tool. Although this approach helps confirm the differences and rule out confusion brought on by the results's inherent variability, it may overlook small differences when a null hypothesis cannot be rejected. There are many results in our experiments indicating consistent trends in average values, but significance tests did not confirm the differences. The use of significance tests may have caused us to miss some important relations.

Lastly, RBF, our choice for a cost-to-go approximation, was not specifically compared to an aggregation approach. Sarsa with an aggregation approach requires less computation. We did, however, include an aggregation approach in our pre-experiments. They show that the aggregation approach performs well as an alternative to a cost-to-go approximation. It is possible that an aggregation approach may be more suitable than RBF for this problem setting (a single echelon inventory problem with discrete-value state/action variables). However, we investigated RBF for its potential for scalability and applicability beyond single-echelon inventory management.

6.5 Ideas for Future Research

The most direct extension to this study is to investigate how learning-based and simulation-based ADP methods perform in a multi-echelon inventory problem. For a single-echelon problem, we were able to directly use a system state as an ADP state and a system action, which is a scalar, as an ADP action. A multi-echelon problem deals with more inventory units. Its system state and action will have more dimensions. Using a system state and a system action as an ADP state and an ADP action poses substantial computational difficulties. In addition to general ADP techniques for scalability, domain knowledge should be exploited.

On the issue of an objective function, some previous multi-echelon inventory studies used a target service level as an objective rather than operating costs. Relying on a target service level alone may

not be a good idea, because a deviation from a target level may have an asymmetrical consequence. For example, a service level that is slightly over a target level may be preferable to a service level that is slightly under a target level. In addition, distribution of costs among inventory units, which may represent different business entities, may be unfair, as is discussed in Zhao and Xie [131]. The issue of cost/profit distribution should be taken into account in order to scale inventory control to multi-echelon cases.

Regarding evaluation, independent Rollout setups for each inventory unit can serve as benchmarks in multi-echelon problems. Also, our findings in single-echelon inventory problems serve as building blocks to the attainment of a practical solution for multi-echelon problems, with some additional challenges, including coordinating among each inventory unit, composing an appropriate objective function, establishing problem criteria and handling of larger state and action space.

In addition to its potential extension to a multi-echelon system, this study has laid the foundation for investigating many issues warranting further investigation. Serendipitously, this study found a link between high bootstrapping parameters and ADP conservative behavior. This finding is new and needs further study to further confirm its nature as well as to better formulate the new relationship. Understanding this relationship may contribute to a new design of ADP, which may explicitly incorporate a factor to control conservative behavior, or to create a systematic strategy to determine ADP parameters to balance out average and worst case performance.

In addition to the investigation of and extension of existing methods, Direct Credit Back is newly developed here. Though significance tests did not confirm, its average costs indicated that Direct Credit Back has a potential improvement over the Residual Gradient method, but with extra requirements of memory, computation and implementation effort. The development of Direct Credit Back may be said to be still in an early stage of development. A better credit back mechanism should be established to make it simpler to implement and more efficient to employ. Though the benefit of Direct Credit Back was not be shown to be significant here, possibly because of a nature of an inventory problem as a fast-return problem, the potential of Direct Credit Back is worth investigation in a delayed-return problem. In addition, using linear approximation RBF allows Sarsa to perform stably. Investigation of a nonlinear approximation function, e.g. MLP, may allow Residual Gradient and Direct Credit Back to show greater viability. Since the Residual Gradient method is guaranteed to converge, Direct Credit Back, as the Residual Gradient extension, should be investigated to see if its convergence is guaranteed.

On the issue of ADP robustness against the GARCH(1,1) model, our results support ADP model-free robustness and agree with Csáji and Monostori [34]'s work. This contradicts Zhang [130]'s claim

of the significance of GARCH(1,1) in inventory problem, at least when an inventory is managed by a learning-based ADP method. Our experiments of simulation-based ADP methods assume a known correct model of the problem. The performance of a simulation-based ADP method when its model does not include GARCH(1,1) is unexplored. The investigation of simulation-based ADP robustness against the GARCH(1,1) model may agree with Zhang [130] or eliminate concern about GARCH(1,1) completely, when ADP is applied.

The current study compares learning-based and simulation-based ADP methods to each other. As mentioned earlier, each method has its own strength. Learning-based ADP is more adaptive. Simulation-based ADP is more effective at reducing cost and does so more reliably. A combination of learning-based and simulation-based schemes may provide a good balance between adaptibility and reliability. A forecasting technique can be used to estimate uncertainty and variance, or a distribution, for a simulation. Then, with known system dynamics, simulation can be used to approximate a state-action value. A forecasting technique provides adaptibility, while a simulation provides reliability as it reduces variation in control quality. A hybrid system may provide a new adaptive and reliable solution for an inventory problem as well as for other decision/control applications. Kim et al. [71] has started in this direction. They used known system dynamics to simulate consequences of actions not taken, after demand is observed. Then, not only is a value of a taken action updated, but values of all other actions are also updated. Therefore, their approach provides a fast-adapting inventory control system. However, Kim et al.'s work is limited to a stateless inventory problem. The extension to a state-based case will produce a wider range of applications.

Another issue worth mentioning is that of a large action search space. In ADP, the decided action is either a found as an optimal action, $a_t = \min_{a \in A(s)} \hat{C}(s_t, a)$, or its explorative modification, e.g. noise addition, softmax and ϵ-greedy. When an action is multi-dimensional, the action search space can be quite large. A heuristic search is generally a way to alleviate the problem of searching over a large space. Another option is a policy gradient method, which has been investigated in Baxter and Bartlett [11], Baxter et al. [12], Sutton et al. [115] and Cao [23]. Rather than relying on an approximate state-action cost, their policy gradient method uses an approximate policy, $\pi(s_t, a_t; \theta) = Pr[a = a_t | s = s_t, \theta]$, to determine an action directly. A temporal difference learning technique corrects approximate state-action costs and the decision is made based on these approximate costs. A policy gradient method does not keep track of state-action costs. It corrects policy parameters θ according to a new observation and the decision is made based on the approximate policy $\pi(\cdot, \cdot; \theta)$. This approach is still an active research area. When it yields a stable controller, policy gradient is expected to be more efficient, especially when the state-action space is large. Therefore, the policy gradient method is worth investigation for a multi-echelon problem.

Finally, as mentioned in Section 6.4, we did not spend much effort in fine tuning Sarsa's learning rate. Given a constant step size strategy, fine tuning a learning rate seems to be a daunting task, especially when each experiment is computationally expensive to conduct. Powell [97] provides a recent review of advanced step size strategies. Some of advanced step size schemes, e.g. BAKF, can adjust themselves according to observed regression errors. However, determining a learning rate for ADP is a different problem from determining a step size for a regression application, this being an issue of exploration/exploitation balance. It may be more suitable for ADP to have a learning rate strategy adjusting a learning rate according to a degree of exploration, instead of regression errors. Nevertheless, an advanced learning rate strategy may help mitigate an issue of fine tuning a learning rate and this is worth investigating.

BIBLIOGRAPHY

[1] H. Akaike. A new look at the statistical model identification. *IEEE Transactions on automatic control*, 19(6):716–723, 1974.

[2] C. W. Anderson. Approximating a policy can be easier than approximating a value function, computer science technical report cs-00-101. Technical report, Colorado State University, 2000.

[3] C. W. Anderson. CS545: Machine Learning (Fall 2006), Class Lecture, 2006.

[4] K. J. Arrow. The genesis of "optimal inventory policy". *Operations Research*, 50(1):1–2, 2002.

[5] K. J. Arrow, T. Harris, and J. Marschak. Optimal inventory policy. *Econometrica*, 19(3): 250–272, 1951.

[6] S. Axsäter. Using the deterministic eoq formula in stochastic inventory control. *Management Science*, 42(6):830–834, 1996.

[7] L. Baird. Residual algorithms: Reinforcement learning with function approximation. In *Proceedings of the 12th International Conference on Machine Learning*, pages 30–37. Morgan Kaufmann, 1995.

[8] S. N. Balakrishnan, J. Ding, and F. L. Lewis. Issues on stability of adp feedback controllers for dynamic systems. *IEEE Transactions on Systems, Man, and Cybernatics-Part B*, (4):913–917, 2008.

[9] A. M. S. Barreto and C. W. Anderson. Restricted gradient-descent algorithm for value-function approximation in reinforcement learning. *Artificial Intelligence*, 172:454–482, 2008.

[10] A. R. Barron. Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Transactions on Information Theory*, 39(3), 1993.

[11] J. Baxter and P. Bartlett. Direct gradient-based reinforcement learning: I. gradient estimation algorithms. Technical report, Computer Sciences Laboratory, Australian National University, 1999.

[12] J. Baxter, L. Weaver, and P. Bartlett. Direct gradient-based reinforcement learning: Ii. gradient ascent algorithms and experiments. Technical report, Computer Sciences Laboratory, Australian National University, 1999.

[13] R. E. Bellman. *Dynamic Programming*. Princeton University Press, Princeton, NJ, 1957.

[14] D. P. Bertsekas. *Dynamic Programming: Deterministic and Stochastic Models*. Prentice-Hall, NJ, 1987.

[15] D. P. Bertsekas and J. N. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, Belmont, MA, USA, 1996.

[16] D. Bertsimas and A. Thiele. A robust optimization approach to inventory theory. *Operations Research*, 54(1):150–168, 2006.

[17] S. A. Billings, H. L. Wei, and M. A. Balikhin. Generalized multiscale radial basis function networks. *Neural Networks*, 20:1081–1094, 2007.

[18] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[19] R. G. Brown. Decision rules for inventory management. 1967.

[20] R. G. Brown. *Production and Inventory Control Handbook, James H. Greene editor*, chapter Inventory Control Theory. McGraw-Hill, 3rd edition, 1997.

[21] K. P. Burnham and D. Anderson. *Model Selection and Multi-Model Inference*. Springer, 3rd edition, 2003.

[22] J. A. Buzacott. Economic order quantities with inflation 1975. *Operational Research Quarterly*, 26(3):553–558, 1975.

[23] X.-R. Cao. A basic formula for online policy gradient algorithms. *IEEE Transactions on Automatic Control*, 50(5):696–699, 2005.

[24] A. S. Caplin. The variability of aggregate demand with (s,s) inventory policies. *Economica*, 53(6):1395–1409, 1985.

[25] S. K. Chaharsooghi, J. Heydari, and S. H. Zegordi. A reinforcement learning model for supply chain ordering management: An application to the beer game. *Decision Support System*, 45: 949–959, 2008.

[26] S. Chen, C. F. N. Cowan, and P. M. Grant. Orthogonal least square learning algorithm for radial basis function networks. *IEEE Transactions on Neural Networks*, 2(2):302–309, Mar 1991.

[27] X. Chen and D. Simchi-Levi. Coordinating inventory control and pricing strategies with random demand and fixed ordering cost: The finite horizon case. *Operations Research*, 52(6): 887–896, 2004.

[28] X. Chen and D. Simchi-Levi. Coordinating inventory control and pricing strategies with random demand and fixed ordering cost: The infinite horizon case. *Mathematics of Operations Research*, 29(3):698–723, 2004.

[29] V. L. R. Chinthalapati, N. Yadati, and R. Karumanchi. Learning dynamic prices in multiseller electronic retail markets with price sensitive customers, stochastic demands, and inventory replenishments. *IEEE Transactions on Systems, Man, and Cybernatics Part C*, 36:92–106, 2006.

[30] J. Choi, M. J. Realff, and J. H. Lee. Approximate dynamic programming: Application to process supply chain management. *AIChe Journal*, 52(7), 2006.

[31] E. K. P. Chong, R. L. Givan, and H. S. Chang. A framework for simulation-based network control via hindsight optimization. In *Proceedings of the 39th IEEE Conference on Decision and Control*, 2000.

[32] P. Cichosz. Truncating temporal differences: On the efficient implementation of TD(λ) for reinforcement learning. *Journal of Artificial Intelligence Research*, 2:287–318, 1995.

[33] A. J. Clark and H. E. Scarf. Optimal policies for a multi-echelon inventory problem. *Management Science*, 6(4):475–490, 1960.

[34] B. C. Csáji and L. Monostori. Value function based reinforcement learning in changing markovian environments. *Journal of Machine Learning Research*, 9:1679–1709, 2008.

[35] G. Cybenko. Approximations by superpositions of a sigmoidal functions. *Mathematics of Control, Signals, and Systems*, 2:303–314, 1989.

[36] T. K. Das, A. Gosavi, S. Mahadevan, and N. Marchalleck. Solving semi-markov decision problems using average reward reinforcement learning. *Management Science*, 45(4):560–574, 1999.

[37] P. Dayan. Prospective and retrospective temporal difference learning. *Computation in Neural Systems*, 20(1):32–46, 2009.

[38] J. W. Demmel. *Applied Numerical Linear Algebra*. Society of Industrial and Applied Mathematics, 1997.

[39] E. V. Denardo. *Dynamic Programming: Models and Applications*. Prentice-Hall, NJ, 1982.

[40] D. Erlenkotter. Ford whitman harris and the economic order quantity model. *Operations Research*, 38(6):937–946, Nov-Dec 1990.

[41] T. Falas and A. Stafylopatis. Implementing temporal-difference learning with the scaled gradient algorithm. *Neural Processing Letters*, 22:361–375, 2005.

[42] A. Federgruen and A. Heching. Combined pricing and inventory control under uncertainty. *Operations Research*, 47(3):454–475, 1999.

[43] S. Gelly and D. Silver. Combining online and offline knowledge in UCT. In *ICML '07: Proceedings of the 24th international conference on Machine learning*, pages 273–280, New York, NY, USA, 2007. ACM.

[44] S. Geman, E. Bienenstock, and R. Doursat. Neural networks and the bias/variance dilemma. *Neural Computation*, 4(1):1–58, Jan 1992.

[45] A. George and W. B. Powell. Adaptive stepsizes for recursive estimation with applications in approximate dynamic programming. *Machine Learning*, 65:167–198, 2006.

[46] Y. Gerchak and Y. Wang. Periodic-review inventory models with inventory-level-dependent demand. *Naval Research Logistics*, 41:99–116, 1994.

[47] I. Giannoccaro and P. Pontrandolfo. Inventory management in supply chains: a reinforcement learning approach. *International Journal of Production Economics*, 78:153–161, 2002.

[48] G. A. Godfrey and W. B. Powell. An adaptive, distribution-free algorithm for the newsvendor problem with censored demands, with applications to inventory and distribution. *Management Science*, 47(8):1101–1112, Aug 2001.

[49] A. Gosavi. Reinforcement learning: A tutorial survey and recent advances. *INFORMS Journal on Computing*, 21(2):178–192, 2009.

[50] M. T. Hagan and M. Menhaj. Training feed-forward networks with the marquardt algorithm. *IEEE Transactions on Neural Networks*, 5(6):989–993, Mar 1994.

[51] M. T. Hagan, H. B. Demuth, and M. H. Beale. *Neural Network Design*. Martin Hagan, 2002.

[52] F. W. Harris. How many parts to make at once. *Factory, The Magazine of Management*, 10: 135–136, 152, 1913.

[53] W. H. Hausman. Sequential decision problems: A model to exploit existing forecasts. *Management Science*, 16:B93–B111, 1969.

[54] J. Hawkins. *A Lagrangian Decomposition Approach to Weakly Coupled Dynamic Optimization Problems and its Applications*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, 2003.

[55] A. C. Hax and D. Candea. *Production and Inventory Management*. Prentice-Hall, 1984.

[56] S. Haykin. *Neural Networks: A Comprehensive Foundation*. Macmillan College Publishing Company, Inc., 1994.

[57] D. Heath and P. Jackson. Modeling the evolution of demand forecasts with application to safety stock analysis in production/distribution systems. *IIE Transactions*, 26:17–30, 1994.

[58] D. P. Heyman and M. J. Sobel. *Stochastic Models in Operations Research, Vol. II*. McGraw-Hill, NY, 1984.

[59] F. S. Hillier and G. J. Lieberman. *Introduction to Operations Research*. McGraw-Hill, NY, 6th edition, 2000.

[60] H. Hirano and M. Furuya. *Jit Is Flow*. PCS Press, 2006.

[61] K. Hornik, M. Stinchcombe, and H. White. Multi-layer feedforward networks are universal approximators. *Neural Networks*, 2:359–366, 1989.

[62] T. Iida and P. H. Zipkin. Approximate solutions of a dynamic forecast-inventory model. *Manufacturing and Service Operations Management*, 8(4):407–425, Fall 2006.

[63] T. Jaakkola, S. P. Singh, and M. I. Jordan. On the convergence of stochastic iterative dynamic programming algorithms. *Neural Computation*, 6:1185–1201, 1994.

[64] C. Jiang and Z. Sheng. Case-based reinforcement learning for dynamic inventory control in a multi-agent supply chain system. *Expert Systems with Applications*, 36:6520–6526, 2009.

[65] L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 14:237–285, 1996.

[66] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101:99–134, 1998.

[67] E. L. Kaplan and P. Meier. Nonparametric estimation from incomplete observations. *Journal of the American Statistical Association*, 53(282):457–481, 1958.

[68] M. Karakul. *Combined Pricing and Procurement Decisions in Stochastic Inventory Control Theory*. PhD thesis, University of Toronto, Canada, 2004.

[69] M. Khouja. The single-period (news-vendor) problem: literature review and suggestions for future research. *Omega*, 27:537–553, 1999.

[70] C. O. Kim, J. Jun, J. K. Baek, R. L. Smith, and Y. D. Kim. Adaptive inventory control models for supply chain management. *International Journal of Advanced Manufacturing Technology*, 26:1184–1192, 2005.

[71] C. O. Kim, I. H. Kwon, and J. G. Baek. Asynchronous action-reward learning for nonstationary serial supply chain inventory control. *Applied Intelligence*, 28:1–16, 2008.

[72] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[73] P. Kleinau and W. Thonemann. Deriving inventory-control policies with genetic programming. *OR Spectrum*, 26:521–546, 2004.

[74] T. Kohonen. *Self-Organizing Maps*. Springer, 1995.

[75] I. H. Kwon, C. O. Kim, J. Jun, and J. H. Lee. case-based myopic reinforcement learning for satisfying target service level in supply chain. *Expert Systems with Applications*, 35:389–397, 2008.

[76] D. Lambert, J. R. Stock, and L. M. Ellram. *Fundamentals of Logistics*. McGrawHill/Irwin, 1998.

[77] H. L. Lee and C. Billington. Managing supply chain inventory: Pitfalls and opportunities. *Sloan Management Review*, Spring 1992.

[78] H. L. Lee, V. Padmanabhan, and S. Whang. Information distortion in a supply chain: The bullwhip effect. *Management Science*, 43(4):546–558, 1997.

[79] M. L. Littman. A tutorial on partially observable markov decision processes. *Journal of Mathematical Psychology*, 53:119–125, 2009.

[80] X. Lu, J.-S. Song, and A. Regan. Inventory planning with forecast updates: Approximate solutions and cost error bounds. *Operations Research*, 54(6):1079–1097, November-December 2006.

[81] J. MacQueen. Some methods for classification and analysis of multivariate observation. In *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability (L.M. LeCun and J. Neyman, eds.)*, volume 1, pages 281–291. Berkeley: University of California Press, 1967.

[82] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[83] M. F. Moller. A scaled conjugate gradient algorithm for fast supervised learning. *Neural Networks*, 6:1525–533, 1993.

[84] E. Naddor. *Inventory Systems*. John Wiley & Sons, 1966.

[85] S. Nahmias. Myopic approximations for the perishable inventory problem. *Management Science*, 22:1002–1008, 1976.

[86] S. Nahmias and S. A. Smith. Optimizing inventory levels in a two echelon retailer system with partial lost sales. *Management Science*, 40:582–596, 1994.

[87] M. J. L. Orr. Introduction to radial basis function networks. Technical report, Institute for Adaptive and Neural Computation, Division of Informatics, Edinburgh University, 1996. URL `http://www.anc.ed.ac.uk/~mjo/papers/intro.ps`.

[88] O. Özer and W. Wei. Inventory control with limited capacity and advance demand information. *Operations Research*, 52(6):988–1000, Nov-Dec 2004.

[89] J. Park and W. Sandberg. Universal approximation using radial-basis-function networks. *Neural Computation*, 3:246–257, 1991.

[90] C. D. Paternina-Arboleda and T. K. Das. Intelligence dynamic control policies for serial production lines. *IIE Transactions*, 33:65–77, 2001.

[91] R. Paterson and E. A. Silver. *Decision Systems for Inventory Management and Production Planning*. John Wiley & Sons, 1979.

[92] N. C. Petruzzi and M. Dada. Pricing and the newsvendor problem: A review with extensions. *Operations Research*, 47(2):183–194, 1999.

[93] P. Pontrandolfo, A. Gosavi, and O. G. Okobaa. Global supply chain management: a reinforcement learning approach. *International Journal of Production Research*, 40(6):1299–1317, 2002.

[94] E. L. Porteus. *Foundations of Stochastic Inventory Theory*. Stanford University Press, 2002.

[95] M. J. D. Powell. Radial basis function approximations to polynomials. In *Proceeding 12th Biennial Numerical Analysis Conference (Dundee)*, pages 223–241, 1987.

[96] W. B. Powell. *Approximate Dynamic Programming: Solving the Curses of Dimensionality*. Wiley, 2007.

[97] W. B. Powell. What you should know about approximate dynamic programming. *Naval Research Logistics*, 56:239–249, 2009.

[98] M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley and Sons, 1994.

[99] M. Riedmiller and H. Braun. A direct adaptive method for faster backpropagation learning: The rprop algorithm. In *Proceedings of the IEEE International Conference on Neural Networks*, 1993.

[100] S. M. Ross. *Introduction to stochastic Dynamic Programming*. Academic Press, NY, 1983.

[101] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.

[102] S. Russell and P. Norvig. *Artificial Intelligence A Modern Approach*. Prentice Hall, 2nd edition, 2003.

[103] H. E. Scarf. Inventory theory. *Operations Research*, 50(1):186–191, 2002.

[104] J. J. Schneider and S. Kirkpatrick. *Stochastic Optimization*. Springer, 2006.

[105] W. Schultz, P. Dayan, and P. R. Montague. A neural substrate of prediction and reward. *Science*, 275:1593–1599, 1997.

[106] B. Seymour, J. P. O'Doherty, P. Dayan, M. Koltzenburg, A. K. Jones, R. J. Dolan, K. J. Friston, and R. S. Frackowiak. Temporal difference models describe higher-order learning in humans. *Nature*, 429:664–667, 2004.

[107] C. C. Sherbrooke. *Optimal Inventory Modeling of Systems*. Kluwer Academic, 2nd edition, 2004.

[108] S. Shervais, T. T. Shannon, and G. G. Lendaris. Intelligent supply chain management using adaptive critic learning. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 33(2), March 2003.

[109] E. A. Silver. Operations research in inventory management: A review and critique. *Operations Research*, 29(4), Jul-Aug 1981.

[110] E. A. Silver. Inventory modeling, encyclopedia of operations research and management science. 2001.

[111] D. Simchi-Levi, X. Chen, and J. Bramel. *The Logic of Logistics: Theory, Algorithms, and Applications for Logistics and Supply Chain Management*. Springer, 2nd edition, 2005.

[112] S. P. Singh and R. S. Sutton. Reinforcement learning with replacing eligibility traces. *Machine Learning*, 22:123–158, 1996.

[113] R. S. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3:9–44, 1988.

[114] R. S. Sutton and A. G. Barto. *Reinforcement Learning*. MIT Press, 1998.

[115] R. S. Sutton, D. Mcallester, S. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. In *In Advances in Neural Information Processing Systems 12*, pages 1057–1063. MIT Press, 1999.

[116] G. J. Tesauro. TD-gammon, a self-teaching backgammon program, achieves master level play. *Neural Computation*, 6(2):215–219, 1994.

[117] H. Topaloglu and S. Kunnumkal. Approximate dynamic programming methods for an inventory allocation problem under uncertainty. *Naval Research Logistics*, 53, 2006.

[118] B. Van Roy, D. P. Bertsekas, Y. Lee, and J. N. Tsitsiklis. A neuro-dynamic programming approach to retailer inventory management. In *Proceedings of the IEEE Conference on Decision and Control*, 1997.

[119] A. F. Veinott and H. M. Wagner. Computing optimal (s,s) inventory policies. *Management Science*, 11(5):525–552, 1965.

[120] G. K. Venayagamoorthy, R. G. Harley, and D. C. Wunsch. Comparison of heuristic dynamic programming and dual heuristic programming adaptive critics for neurocontrol of a turbogenerator. *IEEE Transactions on Neural Networks*, 13(3), 2002.

[121] H. M. Wagner and T. M. Whitin. Dynamic version of the economic lot size model. *Management Science*, 5(1):89–96, 1958.

[122] C. D. J. Waters. *Logistics: An Introduction to Supply Chain Management*. Palgrave Macmillan, 2003.

[123] C. J. C. H. Watkins. *Learning from Dalayed Rewards*. PhD thesis, Cambridge University, 1989.

[124] P. Werbos. *Handbook Intelligent Control, D. White and D. Sofge Eds.*, chapter Neurocontrol and supervised learning: An overview and evaluation. Van Nostrand Rheinhold, New York, 1992.

[125] P. Werbos. *Handbook of Intelligent Control: Neural, Fuzzy, and Adaptive Approaches, eds. White, D. A. and Sofge, D. A.*, chapter Approximate Dynamic Programming for Real-Time Control and Neural Modeling. Multiscience Press Inc., 1992.

[126] P. Werbos. *ADP: Goals, Opportunities and Principles*. IEEE Press, 2004.

[127] D. Wetterschereck and T. Dietterich. *Improving the performance of radial basis function networks by learning center locations*, pages 1133–1140. Morgan Kaufmann, San Mateo, CA, 1992.

[128] C. C. White III. *Encyclopedia of Operations Research and Management Science, edited by Gass and Harris*, chapter Dynamic Programming. Kluwer, 2001.

[129] P. S. You. A heuristic approach for multiple item and location ordering problem with quantity discount and capacity constraint. *Journal of the Operational Research Society*, 56(3):307–316, 2005.

[130] X. Zhang. Inventory control under temporal demand heteroscedasticity. *European Journal of Operational Research*, 182:127–144, 2007.

[131] X. Zhao and J. Xie. Forecasting errors and the value of information sharing in a supply chain. *International Journal of Production Research*, 40(2):311–335, 2002.

[132] P. H. Zipkin. *Foundations of Inventory Management*. McGraw-Hill/Irwin, 2000.

CHAPTER 7

APPENDICES

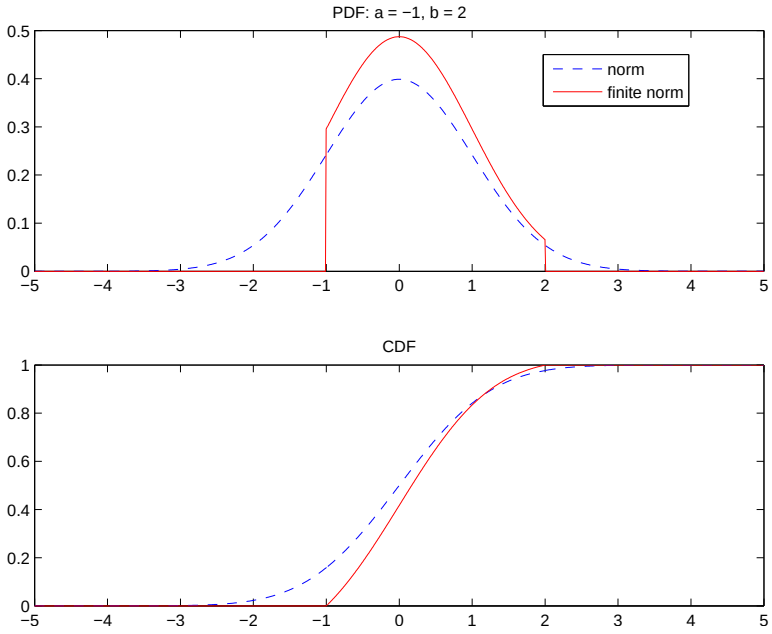

Figure 7.1: PDF and CDF of finite range normal distribution

7.1 Finite range normal function

The finite range Normal function, denoted as $f(x; \mu, \sigma^2, a, b)$ to indicate all parameters or $f(x)$ when omitting parameters, is a proper probability function, i.e., $f(x) \geq 0$ for all x, $\int_{-\infty}^{\infty} f(x)dx = 1$, and $\int_{\alpha}^{\beta} f(x)dx = Pr[\alpha \leq x \leq \beta]$. It behaves similar to a Normal probability function when x falls within allowed range, $x \in [a, b]$, and $f(x) = 0$ otherwise.

Equation 7.3 shows how a uniform random number F_0 is mapped to finite range normal x_{F_0} such that $x_{F_0} \in [a, b]$ and that its value is more likely to be close to μ. Figure 7.1 shows the PDF and CDF of finite Normal distributions of $a = -1$, $b = 2$, $\mu = 0$, and $\sigma = 1$.

The finite Normal distribution is shown in a solid line and the regular Normal distribution is shown in a dashed line. It should be noted that the new distribution may have mean and variance different from μ and σ^2 and it can be very skewed.

It should also be noted that, in implementation, round-off plays an important role and can cause x_{F_0} to be out of range. For example, when $a' = 2$, $\text{erf}(a' = 2) \approx 0.9953$ and when $a' = 3$, with 4 decimal precision, the error function will be rounded to 1 and when it is reinverted, it may not fall back to a'. Figure 7.2 shows this round off error. The plot is prepared with four-decimal-point calculation.

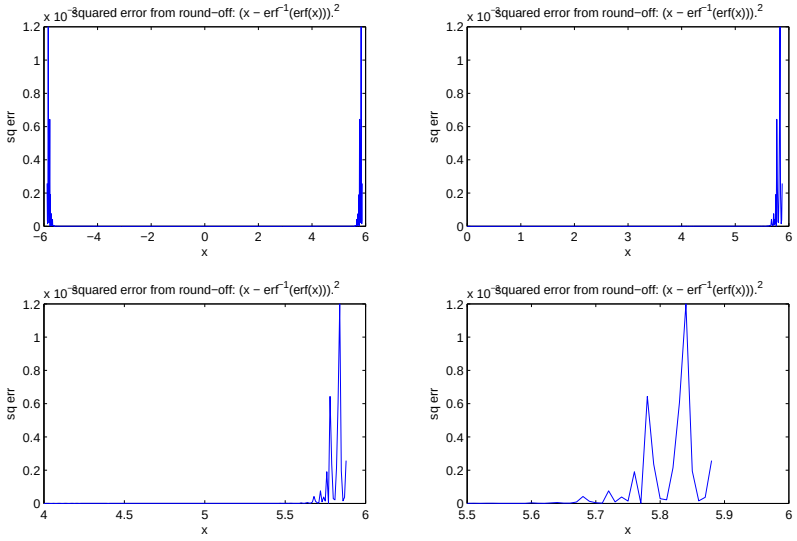

Figure 7.2: Plot of squared error from rounding off in $\text{erf}^{-1}(\text{erf}(x))$

$$f(x) = \begin{cases} \frac{1}{\sigma\sqrt{\pi/2}\cdot(\text{erf}(b')-\text{erf}(a'))} \cdot \exp(\frac{-(x-\mu)^2}{2\sigma^2}) & , x \in [a,b] \\ 0 & , x < a \text{ or } x > b \end{cases} \tag{7.1}$$

$$F(x) = \begin{cases} \frac{1}{\text{erf}(b')-\text{erf}(a')} \cdot \left(\text{erf}(\frac{x-\mu}{\sigma\sqrt{2}}) + \text{erf}(a')\right) & , x \in [a,b] \\ 0 & , x < a \\ 1 & , x > b \end{cases} \tag{7.2}$$

$$x_{F_0} = \mu + \sigma\sqrt{2} \cdot \text{erf}^{-1}\left(F_0 \cdot \{\text{erf}(b') - \text{erf}(a')\} + \text{erf}(a')\right) \ , \ F_0 \in [0,1] \tag{7.3}$$

where $a' = (a-\mu)/(\sigma\sqrt{2})$; $b' = (b-\mu)/(\sigma\sqrt{2})$; the error function[1] $\text{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x \exp(-t^2)dt$; and erf^{-1} is an inverse error function.

As $a \to -\infty$ and $b \to \infty$, $f(x)$ will degenerate to a normal probability function. When $F_0 = 0$, $x_{F_0} = \mu + \sigma\sqrt{2} \cdot \text{erf}^{-1}(\text{erf}(a')) = a$. When $F_0 = 1$, $x_{F_0} = \mu + \sigma\sqrt{2} \cdot \text{erf}^{-1}(\text{erf}(b')) = b$.

[1] Although not defined in closed form, this error function is available in Matlab as `erf`. The inverse error function erf^{-1} is also available in Matlab as `erfinv`.