

T-4236

LEARNING ALGORITHMS FOR SEQUENTIAL
DECISION MAKING PROBLEMS

ARTHUR LAKES LIBRARY
COLORADO SCHOOL OF MINES
GOLDEN, CO 80401

by

Douglas I. Hart

ProQuest Number: 10796522

All rights reserved

INFORMATION TO ALL USERS

The quality of this reproduction is dependent upon the quality of the copy submitted.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if material had to be removed, a note will indicate the deletion.



ProQuest 10796522

Published by ProQuest LLC (2019). Copyright of the Dissertation is held by the Author.

All rights reserved.

This work is protected against unauthorized copying under Title 17, United States Code
Microform Edition © ProQuest LLC.

ProQuest LLC.
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 – 1346

T-4236

A thesis submitted to the Faculty and the Board of Trustees of the Colorado School of Mines in partial fulfillment of the requirements for the degree of Doctor of Philosophy (Mathematics).

Golden, Colorado

Date Nov. 11, 1992

Signed: Douglas I. Hart
Douglas I. Hart

Approved: Boleslaw Tolwinski
Dr. Boleslaw Tolwinski
Thesis Advisor

Golden, Colorado

Date Nov. 12, 1992

Ardel Boes
Dr. Ardel Boes
Professor and Head,
Department of Mathematical and
Computer Sciences

ABSTRACT

Dynamic programming provides both a theoretical and computational framework for the study of optimizing sequential decision making problems. The dynamic programming algorithm is a computationally effective means for solving these problems. However, it has the disadvantage of requiring a complete enumeration of all potential states in the system. For many problems of interest this is often impractical and sometimes impossible to implement.

Here we develop algorithms that require holding only a fraction of the total number of states in computer memory. These algorithms “learn” the optimal sequence of decisions by repeatedly choosing decision sequences and examining the consequences of each decision sequence, an iterative generate-and-test type strategy.

An iteration of a generate-and-test type algorithm begins by using a stochastic mechanism to generate a decision sequence. A testing process then follows, which makes measurements of the performance of the dynamic system. First, we employ a Monte Carlo search mechanism to discuss how different test or evaluative information structures can increase the probability of discovering an optimal decision sequence. Analysis of the evaluative information structures reveals that the search mechanism must visit every state on an optimal path. The first evaluative information structures we examine require that the search visit the states on the optimal path in certain orders. Then an evaluative information structure that relaxes the need to visit the states on the optimal path in any order is put forth.

Having analyzed the “test” part of the generate-and-test algorithm, we next examine a value-based search mechanism that increases the probability of discover-

T-4236

ing an optimal decision sequence. Finally, computational experiments confirm that different evaluative information structures are capable of improving the chance of discovering an optimal decision sequence and show that the value-based search generally outperforms the Monte Carlo search process.

TABLE OF CONTENTS

ABSTRACT	iii
LIST OF FIGURES	x
LIST OF TABLES	xv
ACKNOWLEDGEMENTS	xvi
DEDICATION	xvii
1 INTRODUCTION	1
1.1 Dynamic Programming	1
1.2 Artificial Intelligence and Machine Learning	2
1.3 Examples of Sequential Decision Making Problems	5
1.4 Discussion	5
2 SEQUENTIAL DECISION MAKING PROBLEMS	7
2.1 Ideas from Optimal Control Theory	7
2.1.1 Optimal Control Terminology	8
2.1.2 Sequential Decision Making and Other Problems	11
2.1.3 Solution of Optimal Control Problems	12
2.1.4 Computational Considerations	13
2.2 Deterministic Problems	13
2.2.1 The Longest Path in a Directed Acyclic Graph Problem	14
2.2.2 Dynamic Programming Solution	14
2.3 Stochastic Problems	15
2.3.1 The Markov Decision Process Problem	15

2.3.2	Dynamic Programming Solution	15
2.4	Discussion	16
3	EVALUATIVE INFORMATION PROPAGATION	18
3.1	Information Propagation and Search Procedures	18
3.1.1	Convergence of the Algorithms	19
3.1.2	Stopping Criteria for the Algorithms	21
3.1.3	An Example	21
3.2	Monte Carlo Search Algorithm	22
3.2.1	Description of the Algorithm	22
3.2.2	Solution of the Example Graph	24
3.2.3	Convergence Analysis	25
3.3	Monte Carlo Search Algorithm with Estimated Value Informa- tion Propagation	27
3.3.1	Description of the Algorithm	28
3.3.2	Properties of Estimated Values	28
3.3.3	Solution of the Example Graph	32
3.3.4	Information Propagation	34
3.3.5	Convergence Analysis	35
3.4	Monte Carlo Search with Estimated and Target Value Informa- tion Propagation	39
3.4.1	Description of the Algorithm	40
3.4.2	Properties of Target Values	41
3.4.3	Solution of the Example Graph	47
3.4.4	Information Propagation	49

3.4.5	Convergence Analysis	50
3.5	Monte Carlo Search with Complete Estimated Value Information Propagation	54
3.5.1	Description of the Algorithm	56
3.5.2	Solution of the Example Graph	58
3.5.3	Convergence Analysis	61
3.6	Discussion	66
4	THE VALUE-BASED SEARCH PROCEDURE	69
4.1	Description of the Value-Based Search Procedure	69
4.2	Proof that the Value-Based Search is Correct	73
4.3	Discussion	75
5	COMPUTATIONAL EXPERIMENTS	77
5.1	The Graph Examples	78
5.2	The Algorithms	78
5.3	The Types of Computational Experiments	79
5.4	Computational Experiments on the Example Graphs	79
5.4.1	The Small Example Graph	79
5.4.2	The Tall Graph (T1) Example	82
5.4.3	The Tall Graph (T2) Example	84
5.4.4	The Tall Graph (T3) Example	84
5.4.5	The Wide Graph (W1) Example	84
5.4.6	The Wide Graph (W2) Example	85
5.4.7	The Wide Graph (W3) Example	86
5.5	The Value-Based Search Parameters, p_x and p_y	86

5.6	Discussion	87
6	CONCLUSIONS	89
6.1	Using Information	89
6.2	Directions for Future Research	90
6.2.1	Forgetting Information	90
6.2.2	Stochastic Estimation for the Markov Decision Problem	90
6.2.3	Increasing the Probability of Finding Optimal States	92
A	SMALL GRAPH EXAMPLE	95
A.1	The Small Graph's Topology	95
A.2	Paths in the Example Graph	95
B	SMALL GRAPH COMPUTATIONAL TESTS	98
C	TALL GRAPH (T1) EXAMPLE	103
C.1	Graph T1's Topology	103
C.2	Paths in Graph T1	103
D	TALL GRAPH (T1) COMPUTATIONAL EXPERIMENTS	107
E	TALL GRAPH (T2) EXAMPLE	114
E.1	Graph T2's Topology	114
E.2	Paths in Graph T2	114
F	TALL GRAPH (T2) COMPUTATIONAL EXPERIMENTS	118

G TALL GRAPH (T3) EXAMPLE	125
G.1 Graph T3's Topology	125
G.2 Paths in Graph T3	125
H TALL GRAPH (T3) COMPUTATIONAL EXPERIMENTS	129
I WIDE GRAPH (W1) EXAMPLE	132
I.1 Graph W1's Topology	132
I.2 Paths in Graph W1	132
J WIDE GRAPH (W1) COMPUTATIONAL EXPERIMENTS	135
K WIDE GRAPH (W2) EXAMPLE	142
K.1 Graph W2's Topology	142
K.2 Paths in Graph W2	142
L WIDE GRAPH (W2) COMPUTATIONAL EXPERIMENTS	145
M WIDE GRAPH (W3) EXAMPLE	152
M.1 Graph W3's Topology	152
M.2 Paths in Graph W3	152
N WIDE GRAPH (W3) COMPUTATIONAL EXPERIMENTS	156

LIST OF FIGURES

3.1	Example Graph	22
3.2	Internodal Graph Connections and Estimated Value Computations .	31
3.3	Example Graph for the MC-EV Algorithm	33
3.4	Internodal Graph Connections and Target Value Computations . . .	47
3.5	Example Graph for the MC-TEV Algorithm	48
3.6	Complete Estimated Values for the Example Graph	55
3.7	Example Graph for the MC-CEV Algorithm	59

LIST OF TABLES

1.1	Applications of Sequential Decision Making Problems	6
3.1	Dynamic Programming Solution of the Example Graph	23
3.2	Decision Probabilities for the Monte Carlo Solution of the Example Graph	24
3.3	Information Propagation via Estimated Values for the Example Graph	34
3.4	Convergent Sequences of Events for the MC-EV Algorithm in the Example Graph	37
3.5	Information Propagation via Estimated and Target Values for the Example Graph	49
3.6	Estimated Values for the Example Algorithm Iterations in the Example Graph	55
3.7	Actual, Estimated, and Target Values for the Example Algorithm Iterations in the Example Graph	60
3.8	Complete Estimated Values for the Example Algorithm Iterations in the Example Graph	62
4.1	Data used in the Value-Based Search Procedure	70
5.1	Graph Examples	78
5.2	Algorithm Acronyms	79
5.3	Statistics Generated from the Computational Experiments	80
5.4	Statistical Data for the Graph Examples	80

A.1	Topological Structure of the Small Example Graph	96
A.2	The Myopic Path through the Small Example Graph	97
A.3	The Optimal Path through the Small Example Graph	97
B.1	Monte Carlo Search Tests	98
B.2	Value-Based Search Tests (MN-IC)	99
B.3	Value-Based Search Tests (SD-IC)	100
B.4	Value-Based Search Tests (MN-NNV)	101
B.5	Value-Based Search Tests (SD-NNV)	102
C.1	Graph T1's Topological Structure	104
C.2	The Myopic Path through Graph T1	105
C.3	The Optimal Path through Graph T1	106
D.1	Monte Carlo Search Tests	107
D.2	Value-Based Search Tests (MN-IC)	108
D.3	Value-Based Search Tests (SD-IC)	109
D.4	Value-Based Search Tests (MN-NNV)	110
D.5	Value-Based Search Tests (SD-NNV)	111
D.6	Value-Based Search Tests (MN-EV)	112
D.7	Value-Based Search Tests (SD-EV)	112
D.8	Value-Based Search Tests (MN-ITR)	113
D.9	Value-Based Search Tests (SD-ITR)	113
E.1	Graph T2's Topological Structure	115
E.2	The Myopic Path through Graph T2	116
E.3	The Optimal Graph T2	117

F.1	Monte Carlo Search Tests	118
F.2	Value-Based Search Tests (MN-IC)	119
F.3	Value-Based Search Tests (SD-IC)	120
F.4	Value-Based Search Tests (MN-NNV)	121
F.5	Value-Based Search Tests (SD-NNV)	122
F.6	Value-Based Search Tests (MN-EV)	123
F.7	Value-Based Search Tests (SD-EV)	123
F.8	Value-Based Search Tests (MN-ITR)	124
F.9	Value-Based Search Tests (SD-ITR)	124
G.1	Graph T3's Topological Structure	126
G.2	The Myopic Path through Graph T3	127
G.3	The Optimal Path through Graph T3	128
H.1	Monte Carlo Search Tests	129
H.2	Value-Based Search Tests (MN-EV)	130
H.3	Value-Based Search Tests (SD-EV)	130
H.4	Value-Based Search Tests (MN-ITR)	131
H.5	Value-Based Search Tests (SD-ITR)	131
I.1	Graph W1's Topological Structure	133
I.2	The Myopic Path through Graph W1	134
I.3	The Optimal Path through Graph W1	134
J.1	Monte Carlo Search Tests	135
J.2	Value-Based Search Tests (MN-IC)	136
J.3	Value-Based Search Tests (SD-IC)	137

J.4	Value-Based Search Tests (MN-NNV)	138
J.5	Value-Based Search Tests (SD-NNV)	139
J.6	Value-Based Search Tests (MN-EV)	140
J.7	Value-Based Search Tests (SD-EV)	140
J.8	Value-Based Search Tests (MN-ITR)	141
J.9	Value-Based Search Tests (SD-ITR)	141
K.1	Graph W2's Topological Structure	143
K.2	The Myopic Path through Graph W2	144
K.3	The Optimal Path through Graph W2	144
L.1	Monte Carlo Search Tests	145
L.2	Value-Based Search Tests (MN-IC)	146
L.3	Value-Based Search Tests (SD-IC)	147
L.4	Value-Based Search Tests (MN-NNV)	148
L.5	Value-Based Search Tests (SD-NNV)	149
L.6	Value-Based Search Tests (MN-EV)	150
L.7	Value-Based Search Tests (SD-EV)	150
L.8	Value-Based Search Tests (MN-ITR)	151
L.9	Value-Based Search Tests (SD-ITR)	151
M.1	Graph W3's Topological Structure	153
M.2	The Myopic Path through Graph W3	154
M.3	The Optimal Path through Graph W3	155
N.1	Monte Carlo Search Tests	156
N.2	Value-Based Search Tests (MN-EV)	157

N.3 Value-Based Search Tests (SD-EV)	157
N.4 Value-Based Search Tests (MN-ITR)	158
N.5 Value-Based Search Tests (SD-ITR)	158

ACKNOWLEDGEMENTS

The academic and research support provided by my advisor, Dr. Boleslaw Tolwinski is invaluable. I have found Bolek's insight into mathematics and the nature of mathematical research to be a source of inspiration. Countless discussions with Dr. Robert Underwood, who is on my thesis committee and is one of Bolek's colleagues, I deeply appreciate.

I also would like to thank the other members of my thesis committee, Dr. Steven Pruess, Dr. Chidamber Ganesh, and Dr. Ken Lerner. When needed, they all have been more than willing to help. Many thanks also go to the faculty members of the Department of Mathematical and Computer Sciences for the excellent classes they have taught. The mathematical foundations and rigors of analysis I learned, I hope never to forget.

I also appreciate the financial support from the Department of Mathematical and Computer Sciences, which has paid my tuition for over three years. The Graduate School also deserves special mention for providing additional financial support that has allowed me to work full-time on my research and thesis.

Lastly, I would like to thank my wife Sharon. -Thanks for helping to make it possible for me to return to school and achieve my dream of getting a doctoral degree.

DEDICATION

This thesis is dedicated to the memory of my father,

Meredith Ira Hart (1930 - 1991).

Long ago, before I entered elementary school my father began to encourage me to do well in mathematics. –Dad, I know you would have been proud of me.

Chapter 1

INTRODUCTION

Sequential decision making problems, as the name implies, characterize situations where decision making occurs in stages. The outcome of each decision may or may not be totally predictable depending on whether the system under consideration is deterministic or stochastic. However, we can observe the result of making a decision before making the next decision. The objective, in these problems, is to find the set of decisions or a policy that will minimize (maximize) the costs (rewards) associated with operating the system. A key observation is that we cannot make the decisions in isolation since we must balance our desire for low (high) present costs (rewards) against the possibility of high (low) costs (rewards) in the future.

1.1 Dynamic Programming

Dynamic programming, whose formal development is credited to Richard Bellman's work [1], provides a theoretical and computational framework for optimizing sequential decision making problems. The dynamic programming (DP) algorithm results from a simple idea, the *principle of optimality*. The principle of optimality asserts: An optimal policy has the property that whatever the initial state and the initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision. For the sequential decision making problem this means, make the decision that minimizes (maximizes) the sum of the current stage costs (rewards), and the best cost (reward) that we can expect in

the future.

Although the DP algorithm provides significant computational savings as compared with an exhaustive search through all sequences of decisions, it is still computationally prohibitive on “large” problems. This is because DP requires a complete enumeration of all the states that the system might enter. To deal with these large problems, various heuristic procedures have been developed. In large part, the artificial intelligence and machine learning communities inspire some of the more interesting algorithms.

1.2 Artificial Intelligence and Machine Learning

When the number of states that the dynamic system can enter becomes larger than can be held effectively in computer memory, we must consider alternatives to the DP algorithm for optimizing the performance of these systems. Often it is possible to evaluate the performance of the dynamic system for a given sequence of decisions. For these types of systems we envision a “learning algorithm” that performs the optimization task. The basic structure of this type of algorithm is:

1. Using a “search procedure” generate a sequence of decisions and consequently a sequence of states the algorithm visits.
2. Using a specified objective function, evaluate the performance of the dynamic system that results from using this decision sequence.
3. **If** the value of the objective functions meets a stopping criterion, **stop**; **else** store information regarding the performance of this decision sequence, update the search procedure, and go to step 1.

Two ideas are key for the algorithms we are considering; the propagation of information among states of the dynamic system that allows evaluation of the dynamic systems performance and a search procedure that selects decisions at the states of the dynamic system.

We simulate the evolution of the dynamic system by repeatedly selecting decisions and evaluating the consequences of these decisions. Through these simulations we gain knowledge of the value of the objective function that can be realized from various states. Propagation of evaluative information, among the states visited by the algorithm, provides a mechanism for finding decision sequences leading to better values of the objective function. The advantage of these algorithms over DP is that they only require access to a subset of all possible states that the dynamic system might enter.

A search procedure guides the selection of decisions for the simulations of the dynamic system. To insure convergence, the search procedure must guarantee that the probability of selecting any decision at every state is bounded away from zero for an unbounded number of the algorithm's iterations. This is a quite general requirement. We call algorithms that satisfy this condition, "correct." Although all correct algorithms converge, the rate of convergence can be very slow. Information regarding the underlying application that can be used to direct the search procedure into areas where we suspect optimality can significantly increase the rate of convergence.

If our search procedures and evaluative information propagation structures lead to knowledge of decision sequences of increasing value then we can say that algorithm is learning. The artificial intelligence and machine learning communities call this type of learning, *reinforcement learning* [2, 3, 4, 5].

Generally, reinforcement learning is the process of learning with a critic. We choose an input, the critic evaluates its performance. Using such a performance measure, other inputs are selected that might yield higher levels of performance. The critic, however, provides no information about where these inputs might be found.

The nature of the dynamic system constrains the selection of decisions at each state during the simulation of the dynamic system. Consequently, the decision making algorithm cannot select any sequence of decisions. Furthermore, the decision making algorithm can only examine the current state and the potential decisions there, to make the selection of a decision. The decision making algorithm has only local knowledge of the system dynamics. To accommodate these constraints, the decision making algorithm has the following behavior.

1. Given some state of the dynamic system the algorithm selects a decision to execute.
2. Using this decision and the system dynamics, the system evolves to a new state.

This process begins at a given initial state and repeats until the process reaches a terminal state. Upon termination we evaluate the overall performance of the simulation.

Fundamental to this process is the ability to make an admissible decision at each state of the dynamic system. We can identify the decision making process each state uses as a *learning automata* [6]. These learning automata direct the decision making process at each state. In a sense the individual learning automata participate in a cooperative game whose objective is to find the optimal sequence of actions.

1.3 Examples of Sequential Decision Making Problems

Sequential decision making problems arise in many engineering and economic applications. We classify these problems by first noting whether the system under consideration is strictly deterministic or is affected by noise and then has a stochastic nature. When the state and decision spaces are finite a Markov Decision Process¹ results. If these spaces are infinite, the problem is usually referred to as an optimal control problem. In the latter case, for computational reasons it is often necessary to discretize the state and/or decision spaces. The algorithms under consideration apply to Markov Decision Processes with finite state and decision spaces that may also arise from approximations of more general control problems.

Table (1.1) contains a partial list of applications of sequential decision making problems.

1.4 Discussion

In this study we develop a theory of evaluative information propagation. We also discuss necessary conditions that the search procedures must satisfy to insure convergence. To exemplify the process, the information propagation structures are combined with search procedures in order to construct specific algorithms for computational experimentation.

¹For the deterministic problem, a Markov Decision Process is equivalent to the problem of finding an optimal path in an acyclic graph.

Table 1.1: Applications of Sequential Decision Making Problems

Deterministic, finite state and decision spaces
<ol style="list-style-type: none"> 1. Production scheduling 2. The traveling salesman problem 3. Setting optimal binary search trees (computer science) 4. Critical path analysis
Deterministic Optimal Control (infinite state spaces)
<ol style="list-style-type: none"> 1. Aerospace - optimal orbital dynamics 2. Economics 3. Control of chemical plants 4. Engineering 5. Defense
Stochastic, finite state and decision spaces
<ol style="list-style-type: none"> 1. Inventory control 2. Gambling 3. Control of queues 4. Cash management (investments) 5. Capacity expansion 6. Clinical trials 7. Project selection 8. Catalog mailings 9. Forest management 10. Fishery management 11. Scheduling 12. Water resource management 13. Hotel and airline reservations 14. Location analysis 15. Asset selling
Stochastic Optimal Control (infinite state spaces)
<ol style="list-style-type: none"> 1. Aerospace 2. Economics 3. Engineering 4. Defense 5. Preventive maintenance

Chapter 2

SEQUENTIAL DECISION MAKING PROBLEMS

Sequential decision making problems form a subset of the problems generally considered in optimal control theory. Optimal control theory concerns itself with the problem of finding a “control input” that will optimize the performance of a dynamic system. If the dynamic system operates in continuous time, the control input is a function defined on some interval of time. A dynamic system that operates on discrete time steps requires a sequence of control inputs or decisions.

2.1 Ideas from Optimal Control Theory

Several ideas from optimal control theory are useful for describing sequential decision making problems. Central among these is the state equation description of dynamic systems. State equations describe the temporal evolution of a dynamic system. These equations yield the next state of the dynamic system given the current state and a selected control. Also important, is the idea of an objective or reward¹ functional that is additive over time. This functional is a mapping that takes as input the state and control sequences and yields a real number that measures the performance of the dynamic system. The optimal control or sequential decision making problem is that of finding a sequence of controls that will maximize this functional.

¹The theory discussed here can deal with both rewards and costs. To be more precise, our theory addresses the problem of finding a maximum of a reward functional, a minimum of a cost functional, or generally the problem of finding an extreme value for an objective functional. However, the consistent effort required to avoid the words reward, cost, minimize, and maximize in favor of words like optimize and extremize, in the name of generality, is furthestmost from our objectives. So we choose to be specific at this point and discuss the problem of maximizing rewards.

2.1.1 Optimal Control Terminology

- We index discrete **time** with the variable k .
- N is the **horizon** of a problem or its temporal duration, $k \in \{0, 1, \dots, N - 1\}$.
- x_k are the **state variables** for the dynamic system. Each x_k summarizes past information that is relevant for future optimization. The x_k 's also describe the evolution of the dynamic system over time. We write $x_k \in S_k$ to indicate that the state variable x_k can take values in the set S_k . These sets form the **state space** for the dynamic system.
- The variables u_k are the **controls** that are input or applied to the dynamic system. **Decisions** or **actions** are other words often used for controls. The decisions take values from some nonempty set that may depend upon the current state, $u_k \in U_k(x_k) \subset C_k$.
- w_k are **random disturbances** characterized by some probability measure $P_k(\cdot | x_k, u_k)$ that may explicitly depend upon the current state and selected control. However w_k does not depend upon the values of prior disturbances w_{k-1}, \dots, w_0 . To denote the set of disturbances we write $w_k \in D_k$.
- Difference equations describe the evolution of the dynamic system over time. We call these equations, **state equations**. For deterministic dynamic systems the state equations take the form

$$x_{k+1} = f_k(x_k, u_k) \tag{2.1}$$

The dynamics of stochastic systems explicitly depend on a realization of a random variable from the disturbance space. Consequently, the state equations are of the form

$$x_{k+1} = f_k(x_k, u_k, w_k) \quad (2.2)$$

- Associated with the operation of the dynamic system is a set of **rewards**. At each time step k the dynamic system generates a reward. For a deterministic dynamic system the reward may depend on the time step, the state, and the selected control. We denote this reward as $g_k(x_k, u_k)$. Rewards for stochastic dynamic systems might also depend on the random disturbances. The reward here is $g_k(x_k, u_k, w_k)$. Also, at the terminal time N the dynamic system produces a terminal reward that may depend on the terminal state, $g_N(x_N)$. The total reward generated by operating the deterministic dynamic system (2.1) using the control sequence $u = (u_0, u_1, \dots, u_{N-1})$ is the sum

$$J(x_0, u) = g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, u_k) \quad (2.3)$$

Operation of the stochastic dynamic system (2.2) using the control sequence $u = (u_0, u_1, \dots, u_{N-1})$ yields the sum

$$g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, u_k, w_k) \quad (2.4)$$

Because equation (2.4) depends on the random disturbances w_k it is generally a random variable and therefore cannot be meaningfully optimized. However

the expected value of (2.4) can be optimized. So we consider

$$J(x_0, u) = E\left[g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, u_k, w_k)\right] \quad (2.5)$$

where the expected value is taken with respect to the joint distribution of all the random variables involved, $\{w_k : k = 0, 1, \dots, N - 1\}$.

- The primary **objective of optimal control theory** is to find a control sequence $\mathbf{u} = (u_0, u_1, \dots, u_{N-1})$ that maximizes the reward functional. That is, we seek \mathbf{u}^* that maximizes the functionals (2.3) or (2.5) over the set of admissible controls subject to the constraints of the system dynamics (2.1) or (2.2).
- Control sequences fall into two different categories; **open-loop** and **feedback** controls. Open-loop controls specify the action to select at each time step. Feedback controls, on the other hand, depend on both time and the current state. For deterministic problems, (2.1) and (2.3), one can predict the future states given the initial state and the sequence of controls. Here, both the optimal open-loop and optimal feedback controls lead to the same reward. For stochastic problems, (2.2) and (2.5), feedback controls are essential because it is no longer possible to predict future states exactly. Therefore the control to be executed at each time needs to depend on the state we find the dynamic system in currently. In these cases we seek a control law or policy that dictates the action to perform from any state the system might enter. We then specify a policy as

$$\pi = (\mu_0, \mu_1, \dots, \mu_{N-1})$$

where μ_k maps states x_k into controls, $u_k = \mu_k(x_k)$.

2.1.2 Sequential Decision Making and Other Problems

The problems described above are finite horizon deterministic and stochastic sequential decision making problems. The solution of these problems is a finite sequence of actions, which may depend on the current state, to perform as the dynamic system evolves in time. As already mentioned, these problems form only a subset of the problems optimal control theory addresses. Below we briefly contrast aspects of different types of types of optimal control problems. Fleming and Rishel [7] and Bertsekas [8] discuss the theoretical apparatus necessary to describe these problems and their solutions in detail.

Deterministic and Stochastic Problems

The behavior of a deterministic dynamic system follows immediately by specifying an initial state and a sequence of decisions. To determine its evolution one simply solves the state equations for the system using the given initial state and a sequence of decisions. Optimizing a stochastic system also requires the specification of an initial state. However we cannot determine the evolution of the system through the state space because of the random disturbances. In this case the sequence of decisions is a sequence of functions, which depend on the states of the dynamic system, and specify the optimal action to perform at each state of the dynamic system.

Finite and Infinite Horizon Problems

The dynamics of a finite horizon problem terminate at some finite time. Infinite horizon problems never terminate. Analysis of infinite horizon problems is generally more sophisticated than finite horizon problems. One reason for this is that infinite horizon problems introduce all the mathematical problems associated with limits and

infinite sums.

Discrete and Continuous Time Problems

Discrete time optimal control problems have state equations that are difference equations and cost functionals that are sums. The solution of the discrete time problem is in the form of a sequence of decisions. These problems form the class of problems known as sequential decision making problems. A relatively simple generalization to continuous time results in state equations that are differential equations and an integral reward functional. The continuous time optimal control problem solution is a function of time that specifies the decision to execute at every instant in an interval of time.

2.1.3 Solution of Optimal Control Problems

Two theories provide means for solving of optimal control problems, Pontryagin's maximum principle and Bellman's dynamic programming.

The Maximum Principle

Pontryagin's maximum principle offers an analytic method for dealing with finite horizon optimal control problems. It is developed using the Calculus of Variations and is analogous to solving a constrained optimization problem using Lagrange multipliers.

Dynamic Programming

Dynamic programming provides the main analytic and algorithmic impetus for solving finite state space optimal control problems i.e., Markov Decision Processes. The principle of optimality which states, no matter what state the dynamic system is

presently in, make the decision which maximizes the sum of the immediate reward and the reward that can be expected from the state resulting from this decision, shapes the overall structure of the DP algorithm. We call the result of the maximization, computed at the states of a dynamic system, the value of the state. The argument of the maximization is the decision to execute, should the dynamic system evolve into this state. This maximization naturally proceeds backwards in time starting at terminal states. To generate a solution, we need to perform this computation at every state the dynamic system can enter.

2.1.4 Computational Considerations

Before going into the details of the dynamic programming equations we note that these equations are most often unsolvable analytically. Numerical solution is therefore the only viable means for finding solutions to many practical problems. To effect a numerical solution we may need to discretize the state space, decision spaces, and time in some manner. This being the case, the number of elements in each of these spaces will be finite. So from now on we will concentrate on problems that have a finite state space, a finite number of admissible decisions at each state, and a discrete time structure with a finite horizon.

2.2 Deterministic Problems

If the state space is finite, we can label the states of the dynamic system with integers. Similarly, we can label the decisions at each state with integers if the number of decisions is finite. This problem can now be cast into graph theoretic terms. First, we identify each state as a node in a graph and then the decisions as arcs connecting the various graph nodes. The graph arcs have lengths that correspond to the reward

received for making a given decision at a given state. The sequential decision making problem now corresponds to the problem of finding the longest path in the graph starting at the initial node and ending at a terminal node.

2.2.1 The Longest Path in a Directed Acyclic Graph Problem

Let S be a finite set of nodes including an initial state s_0 and a subset of terminal nodes ∂S . Let r_{ij} denote the reward for going from node $i \in S$ to node $j \in S$ (the length of the arc joining nodes i and j), with the understanding that $r_{ij} = -\infty$ if there is no direct connection between i and j . Finally, let r_t be the reward associated with reaching terminal node $t \in \partial S$. The problem is to find a path from s_0 to a terminal node $s_t \in \partial S$ that maximizes the total associated reward.

2.2.2 Dynamic Programming Solution

The value function for the problem at hand satisfies the following equations:

$$\begin{aligned}
 J(i) &= r_i \text{ for } i \in \partial S \\
 J(i) &= \max_{j \in S} \{r_{ij} + J(j)\} \text{ for } i \in \{S \setminus \partial S\}
 \end{aligned}
 \tag{2.6}$$

It is easy to see that $J(i)$ is the optimal reward that accumulates along a path originating at node i and, in particular, $J(s_0)$ is the reward associated with an optimal solution of the problem. Also, knowledge of the value function provides a means, through equation (2.6), to compute an optimal path from any nonterminal node i .

2.3 Stochastic Problems

Similar to the deterministic problem, we consider the case of finite state and decision spaces. Again we label the states and decisions with integers. This problem differs from the deterministic problem only in the fact that, given a decision to execute at some state, we can only make probabilistic statements about the state into which the dynamic system evolves.

2.3.1 The Markov Decision Process Problem

Again let S be a finite set of states including an initial state s_0 and a subset of terminal states ∂S . Let $g(i, u)$ denote the expected reward that results from making decision u when the dynamic system is in any state $i \in S \setminus \partial S$. And finally, let $g_t(i)$ be the terminal reward associated with states $i \in \partial S$.

To formalize the probabilistic nature of the system dynamics we define state transition probabilities $p_{ij}(u)$, which specify the probability of transition from state i to j when decision u is selected. These probabilities must satisfy $0 \leq p_{ij}(u) \leq 1$ for all decisions u , and $\sum_{j \in S} p_{ij}(u) = 1$ for all states i and decisions u . The problem is to find a policy that will maximize the expected reward of operating the stochastic dynamic system starting at the initial state s_0 and terminating in some state $i \in \partial S$.

2.3.2 Dynamic Programming Solution

The value function for this stochastic dynamic system satisfies the following equations:

$$J(i) = g_t(i) \text{ for } i \in \partial S$$

$$J(i) = \max_u \{g(i, u) + \sum_{j \in S} p_{ij}(u)J(j)\} \text{ for } i \in \{S \setminus \partial S\} \quad (2.7)$$

Here $J(i)$ has the interpretation of the optimal expected reward when the dynamic system originates in state i . $J(s_0)$ is the optimal expected reward associated with a solution of this problem. Similar to the deterministic problem, equation (2.7) provides a means to compute the optimal policy from any nonterminal state i .

2.4 Discussion

The concept of the value at a state is central to the dynamic programming solution of the sequential decision making problem. It is a mapping from the state space of the problem into the real numbers. The numerical value it takes is the optimal reward we can realize (deterministic problems) or expect (stochastic problems) for the problem originating at the state under consideration.

Dynamic programming quite naturally provides a feedback solution to the sequential decision making problem. The argument of the value computation at each state results in the optimal decision to execute at that state. Otherwise stated, the computed optimal decision depends on or is a function of the current state.

The drawback of the dynamic programming algorithm is the necessity to compute a value for every state in the problems state space. For moderately sized problems this is not an issue. However, when the number of states that the dynamic system might evolve into becomes large, it can become impractical and sometimes impossible to enumerate and compute values for all possible states. To deal with these large problems we envision algorithms that need visit only a fraction of the dynamic system states.

Important computational techniques, which avoid enumeration of every state,

exist for finding an optimal decision sequence e.g., forward search, Dijkstra's algorithm, branch and bound, and limited lookahead strategies [8]. These techniques apply to problems with nonnegative rewards or to problems where information about the value function is available. The algorithms we will develop in the next chapter, which use a stochastic search mechanism, apply to more general problems.

Chapter 3

EVALUATIVE INFORMATION PROPAGATION

Dynamic programming provides an effective means of solving sequential decision making problems when the dynamic systems states can be held in computer memory. However, for larger problems it is often impractical and sometimes impossible to enumerate all the states into which the dynamic system might possibly enter. While it may not be possible to examine all possible states and decisions, it is quite often possible to examine the consequences of individual sequences of decisions. This leads to the generate-and-test idea where a sequence of actions is chosen and we then evaluate the performance of the dynamic system.

The algorithms we will examine all use this generate-and-test strategy. The algorithms repeatedly select decision sequences and evaluate the performance of the dynamic system. By remembering information concerning these sequences we hope to influence the long term behavior of the decision sequences to yield enough information about the dynamic system to compute the optimal sequence of decisions.

3.1 Information Propagation and Search Procedures

The iterative algorithms we will develop all have the following structure.

- While (Not Converged)
 - (1) Run a decision making simulation starting at the given initial state and ending at a terminal state.

- (a) If a state encountered in the course of simulation has not been visited on a previous iteration of the algorithm, initialize the search procedure for this state and select a decision.
 - (b) If the encountered state has been visited previously, select a decision using the search procedure for this state.
- (2) Evaluate the performance of the decision making episode on this iteration.
- (3) If (Converged) STOP, else go to (1).

Using this structure we will examine how the search procedure selects decisions and consequently directs a search through the state space of the dynamic system. Central to our discussion will be how states can communicate with other states and thereby propagate information about the value function.

To implement these algorithms we need to know what it means for the algorithm to converge. In the process of developing the idea of convergence we will also develop a stopping criterion for these algorithms.

3.1.1 Convergence of the Algorithms

On every iteration of an algorithm, the search procedure produces a sequence of states visited beginning with the initial state and terminating at a terminal state. Associated with this sequence of states there is a real number, a computed value that measures the current best performance of the dynamic system. We say that our algorithm converges when the value associated with these sequences approaches the optimal value of the problem.

Suppose

$$S_k = \{s_{0,k}, s_{1,k}, \dots, s_{N,k}\}$$

is the sequence of states on the k th iteration of the algorithm. ($s_{i,k}$ indicates the selection of state s_i on iteration k .) Associated with each visit to state s_i on iteration k is an estimated value, $\tilde{J}(s_{i,k})$. Generally, these values will depend on estimated values of other states computed previously.

Let

$$\cup S_n = \bigcup_{k=1}^n S_k$$

be the set of all states visited by the algorithm through iteration n . Then the estimated value for state $s_{i,k}$ can be defined as

$$\tilde{J}(s_{i,k}) = \max_{s \in \cup S_n} \{r_{s_{i,k}s} + \tilde{J}(s)\} \quad (3.1)$$

We shall say that the algorithm converges if

$$\lim_{k \rightarrow \infty} \tilde{J}(s_{0,k}) = J(s_0)$$

where $J(s_0)$ denotes the optimal value at s_0 (see Section 2.2.2.) Notice also, that for the deterministic problem and the update scheme (3.1)

$$J(s_0) \geq \tilde{J}(s_{0,k+1}) \geq \tilde{J}(s_{0,k}) \text{ for all } k$$

If $\cup S_n = S$, then the estimated value of the initial state is its value. Clearly then, the value of the initial state is a least upper bound for the sequence of estimated values of the initial state. That the sequence of estimated values is nondecreasing, follows from the definition of the estimated value and the fact that states are never removed from $\cup S_n$.

3.1.2 Stopping Criteria for the Algorithms

When we know the value function at s_0 i.e., $J(s_0)$, a stopping criterion is relatively simple to devise. We continue to iterate until the computed value of the dynamic system equals the known value of the problem. If the value of the problem is unknown it can be difficult to decide when to stop the algorithm. Most often we can only make a statement about the probability that the algorithm has converged. Depending on the search procedure it is possible to assert two types of probabilistic statements about convergence of the algorithm.

1. The probability that the algorithm has converged, i.e., $\tilde{J}(s_{0,k}) = J(s_0)$, after a fixed number of iterations, is greater than some given number.
2. The probability that the algorithm has converged, given that there has been no improvement in performance of the dynamic system in a fixed number of iterations, is greater than some given number.

3.1.3 An Example

Figure (3.1) presents an example of the type of graphs we consider for the deterministic sequential decision making problem. Although not noted on the figure, we will label the decisions at each node¹ sequentially with positive integers starting with the decision with largest immediate reward and ending with smallest immediate reward. For example, at node 9 the decision with immediate reward 4 is labeled 1. The decisions with immediate rewards 2 and 1 are labeled 2 and 3, respectively. For reference, Table (3.1) contains the DP solution for the longest path in graph problem.

¹Since the nodes in a graph represent the states of a dynamic system, for the deterministic sequential decision making problem, the terms “state” and “node” are used interchangeably.

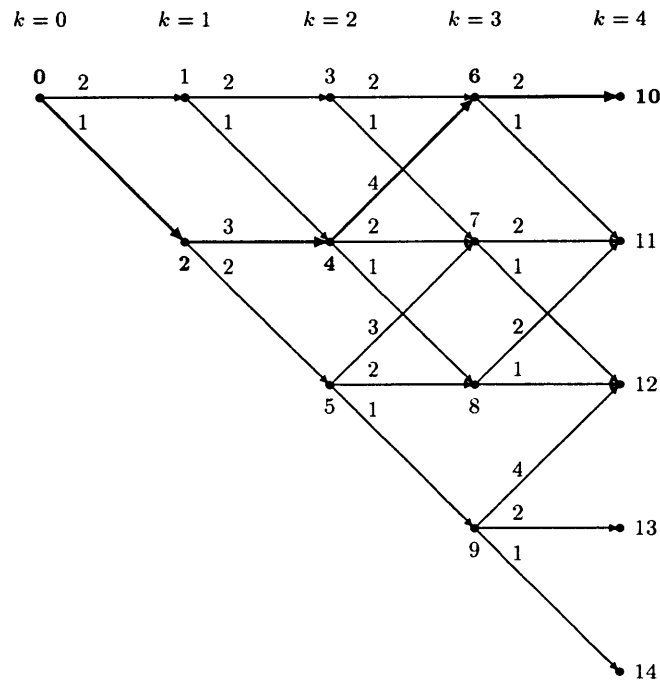


Figure 3.1: Example Graph

Nodes on the optimal path are in bold.

3.2 Monte Carlo Search Algorithm

Perhaps the simplest iterative algorithm for finding an optimal decision sequence is a Monte Carlo search.

3.2.1 Description of the Algorithm

For the Monte Carlo search algorithm (MC algorithm), the process by which a decision is made at any state is uniformly random. For example, at node 0 in the graph in Figure (3.1) the algorithm selects each decision with probability $1/2$. Also,

Table 3.1: Dynamic Programming Solution of the Example Graph

Node i	Value $J(i)$	Decision	Value Computation
0	10	2	$\max\{2 + J(1), 1 + J(2)\}$
1	7	2	$\max\{2 + J(3), 1 + J(4)\}$
2	9	1	$\max\{3 + J(4), 2 + J(5)\}$
3	4	1	$\max\{2 + J(6), 1 + J(7)\}$
4	6	1	$\max\{4 + J(6), 2 + J(7), 1 + J(8)\}$
5	5	1 or 3	$\max\{3 + J(7), 2 + J(8), 1 + J(9)\}$
6	2	1	$\max\{2 + J(10), 1 + J(11)\}$
7	2	1	$\max\{2 + J(11), 1 + J(12)\}$
8	2	1	$\max\{2 + J(11), 1 + J(12)\}$
9	4	1	$\max\{4 + J(12), 2 + J(13), 1 + J(14)\}$
10	0	-	0
11	0	-	0
12	0	-	0
13	0	-	0
14	0	-	0

since there are 3 potential decisions at node 4, the algorithm selects between each of these decisions with probability $1/3$. The algorithm repeatedly chooses random sequences of decisions and evaluates their performance.

The MC algorithm operates using the following scheme:

- (1) Choose a random decision sequence.
- (2) Evaluate the performance of the decision sequence i.e., the value of the reward function.
- (3) If the performance of the current decision sequence exceeds the performance of previous decision sequences, save it as the best decision sequence found so far.
- (4) If (Converged) STOP, else go to (1).

Table 3.2: Decision Probabilities for the Monte Carlo Solution of the Example Graph

Node i	Value $J(i)$	Decision	Probability of making the optimal decision
0	10	2	1/2
2	9	1	1/2
4	6	1	1/3
6	2	1	1/2
10	0	-	-

Clearly, for the MC algorithm to converge it must randomly select an optimal sequence of decisions on some iteration. Suppose there are n_i admissible decisions at state s_i^* on the optimal path. The probability of making a correct decision here is at least $1/n_i$. (There may be more than one optimal decision.) Since the selection of each decision is independent of the other decisions and there are only a finite number of decisions

$$\Pr\{\text{Selecting an optimal decision sequence}\} \geq \frac{1}{n_0} \cdot \frac{1}{n_1} \cdot \dots \cdot \frac{1}{n_{N-1}} > 0$$

Consequently, when the state and decision spaces are finite, the probability of selecting an optimal decision sequence is strictly greater than zero.

3.2.2 Solution of the Example Graph

The longest path in the graph depicted in Figure (3.1) sequentially visits the nodes $\{0, 2, 4, 6, 10\}$. Table (3.2) shows the probabilities of making the optimal decision at each of these nodes. The probability of finding the optimal sequence of decisions is the product of these quantities i.e., it is equal to $1/24$.

3.2.3 Convergence Analysis

Convergence of the MC algorithm follows from one of the Borel-Cantelli lemmas, which Feller [9] proves.

Lemma (Borel-Cantelli): Let A_i be an infinite sequence of mutually independent events. Further let $a_i = \Pr\{A_i\}$. If $\sum_i a_i$ diverges, then with probability one infinitely many A_i occur. ■

If there is more than one optimal sequence of decisions, the probability of finding one of them is greater than the case where only one optimal decision sequence exists. Since the probability of encountering an optimal decision sequence is smallest when the sequence is unique this analysis concerns it, without loss of generality. Let A_1, A_2, \dots be an infinite sequence of trials of a Bernoulli random variable where success is defined as selection of the optimal decision sequence and failure as the selection of any other decision sequence. Also, let

$$p = \Pr\{A_i \text{ is a success}\}$$

Since each selection of a decision sequence is independent of the previous sequence and $p > 0$, which implies $\sum_i p$ diverges, the Borel-Cantelli lemma applies. Therefore, with probability one, infinitely many of the A_i result in success. This means that, in the limit, the MC algorithm will select the optimal decision sequence infinitely often.

If the optimal value of the problem is known and we are only searching for the optimal path, we clearly know when to stop iterating, upon finding a decision sequence with the known optimal value. If the optimal value is unknown, however, all we can do is determine the probability of convergence in any given number of

iterations.

Again, let $p > 0$ be the probability of selecting the optimal sequence of decisions. Let A_1, A_2, \dots be an infinite sequence of trials of a Bernoulli random variable with probability of success p and define $A_i = 1$ for success and $A_i = 0$ for failure. Then

$$\Pr\{A_1 = 1\} = p$$

$$\begin{aligned} \Pr\{A_1 = 1 \cup A_2 = 1\} &= 1 - \Pr\{A_1 = 0 \cap A_2 = 0\} \\ &= 1 - (1 - p)^2 \end{aligned}$$

$$\begin{aligned} \Pr\{A_1 = 1 \cup A_2 = 1 \cup \dots \cup A_n = 1\} &= 1 - \Pr\{A_1 = 0 \cap A_2 = 0 \cap \dots \cap A_n = 0\} \\ &= 1 - (1 - p)^n \end{aligned}$$

These results can be generalized into the following theorem.

Theorem 3.1: Given $\epsilon > 0$ there exists N such that for all $n > N$

$$\Pr\left\{\bigcup_{i=1}^n A_i = 1\right\} > 1 - \epsilon$$

Proof: Let N be the smallest integer greater than $\log_{(1-p)} \epsilon$. Then for any $n > N$,

$$\Pr\left\{\bigcup_{i=1}^n A_i = 1\right\} > \Pr\left\{\bigcup_{i=1}^N A_i = 1\right\} = 1 - (1 - p)^N > 1 - \epsilon$$

■

The theorem above can be used to compute the number of iterations necessary

to find the optimal sequence of decisions with any given probability.

3.3 Monte Carlo Search Algorithm with Estimated Value Information Propagation

The Monte Carlo search algorithm discussed in the previous section provides a simple means for finding an optimal decision sequence. However, when the size of the problem increases, the probability of making all the correct choices on any particular iteration can become very small.

To improve the odds of finding the optimal solution we will next consider an algorithm that chooses decisions randomly but saves information about states visited on previous iterations. To implement this procedure we envision a list that will contain the states that have been visited on past iterations of the algorithm. (Obviously, this will be a list that increases in length as the number of iterations of the algorithm increases.) Associated with each state in the list will be a number, the current estimated value of the state.

Definition 3.2: Let $S_k \subset S$ be the subset of states visited on the k th iteration of the algorithm. Denote the terminal state encountered on this iteration as $s_{N,k}$. Then we define the *estimated value* of state $i \in S_k$ as

$$\begin{aligned}\bar{J}(i) &= r_i \text{ for } i = s_{N,k} \\ \bar{J}(i) &= \max_{j \in \cup S_k} \{r_{ij} + \bar{J}(j)\} \text{ for } i \in S_k \setminus \{s_{N,k}\}\end{aligned}$$

■

Note that the computation of an estimated value depends on the estimated value of descendent states. Naturally, this computation proceeds from a terminal

state back towards the initial state similar to the dynamic programming algorithm.

3.3.1 Description of the Algorithm

The Monte Carlo search algorithm with estimated value information propagation (MC-EV algorithm) works according to the following scheme:

- Initialize a NULL list of states.
 - (1) Choose a random decision sequence and generate the corresponding trajectory of states.
 - (2) Add any state on the trajectory not yet on the list of states to that list.
 - (3) Compute the estimated value of each state on the current trajectory i.e., compute $\bar{J}(i)$ for each $i \in S_k$ according to Definition 3.2.
 - (4) If (Converged) STOP, else go to (1).

The next sections presents the analysis of the MC-EV algorithm.

3.3.2 Properties of Estimated Values

For the states visited on the current iteration, estimated values have the interpretation of the optimal reward that can be received by making previously explored decisions. This is in contrast to the actual value, which is the optimal reward that can be received when considering all decision sequences.

The following lemma states the relationship of estimated values and actual values at all states.

Lemma 3.3: For all states $i \in S$

$$\bar{J}(i) \leq J(i)$$

Proof: Suppose it was not so. Then $\bar{J}(i) > J(i)$ for some $i \in S$. This would imply that there exists a path from state i to a terminal state that resulted in a reward larger than the value of state, $J(i)$. But this contradicts the definition of the value of a state. ■

Proposition 3.4: For any state $i \in S \setminus \partial S$ suppose that the decision to go to state m is the best decision found so far, that is

$$m = \arg \max_j \{r_{ij} + \bar{J}(j)\}$$

Then if the estimated value and actual value are equal at state i i.e., $\bar{J}(i) = J(i)$, the estimated value and actual value are equal at state m , $\bar{J}(m) = J(m)$.

Proof: Since m is the argument of the estimated value computation we have

$$\bar{J}(i) = r_{im} + \bar{J}(m)$$

and then

$$J(i) = r_{im} + \bar{J}(m)$$

From lemma 3.3 we have $\bar{J}(m) \leq J(m)$. Therefore

$$J(i) \leq r_{im} + J(m)$$

Using the definition of the value of state i

$$\max_j \{r_{ij} + J(j)\} \leq r_{im} + J(m)$$

This maximization is achieved by selecting $j = m$ and cannot be larger for any other $j \in S$. Therefore the decision to go to state m at state i is optimal. And then

$$J(i) = r_{im} + J(m)$$

Therefore $\bar{J}(m) = J(m)$. ■

Proposition 3.5: Suppose that at some state $m \in (S - s_0)$ the estimated value and actual value are equal i.e., $\bar{J}(m) = J(m)$, and the optimal decision at some state $i \in S$ is to go to state m ,

$$m = \arg \max_j \{r_{ij} + J(j)\}$$

Then $\bar{J}(i) = J(i)$.

Proof: Since the optimal decision at state i is to go to state m we have

$$J(i) = r_{im} + J(m)$$

and then

$$J(i) = r_{im} + \bar{J}(m)$$

From lemma 3.3 we have $\bar{J}(i) \leq J(i)$. Therefore using the definition of estimated value we have

$$\max_j \{r_{ij} + \bar{J}(j)\} \leq r_{im} + \bar{J}(m)$$

This maximization is achieved by selecting $j = m$ and cannot be larger for any other $j \in S$. So

$$\bar{J}(i) = r_{im} + \bar{J}(m)$$

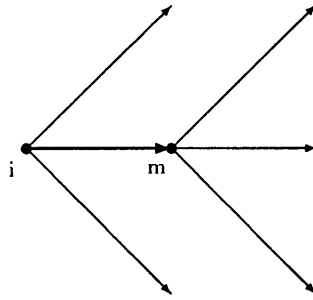


Figure 3.2: Internodal Graph Connections and Estimated Value Computations

And consequently $\bar{J}(i) = J(i)$. ■

Propositions 3.4 and 3.5 state the necessary conditions for propagation of value information among the states. Consider states i and m depicted in Figure (3.2). Proposition 3.4 states that if the estimated value and actual value are equal at i and if the decision to go to m is the best decision found so far, then the estimated value and value at m are equal. This implies the argument of the estimated value computation is the optimal decision. The repeated use of this proposition at states descendent from i yields the optimal decision sequence from i .

The necessary conditions for the estimated value of a state to be equal to its value are given in Proposition 3.5. Again consider Figure (3.2). Proposition 3.5 implies that if the estimated value of m equals its value and the optimal decision at i is to go to m , then the estimated value of i equals its value. This proposition shows that the estimated value at a state will equal its value only when the estimated value of the destination state on the optimal path equals its value.

To compute the value of state i , using dynamic programming, the values of all descendent states must be known. Proposition 3.5 relaxes this condition by saying

that in order to know the value of i , we need only know the value of the state that results by making the optimal decision.

3.3.3 Solution of the Example Graph

To demonstrate the features of the MC-EV algorithm, consider the graph in Figure (3.1) copied here in Figure (3.3). Note that by using the optimal sequence of decisions, the system visits the nodes $\{0, 2, 4, 6, 10\}$. Now suppose that three iterations of the algorithm resulted in the following trajectories,

1. $\{0, 1, 3, 6, 10\}$
2. $\{0, 1, 4, 6, 11\}$
3. $\{0, 2, 4, 8, 11\}$

Table (3.3) shows the computation of the estimated values, $\bar{J}(i)$, at the nodes visited on each iteration. The actual value of the node, $J(i)$, is also listed for reference.

On the first iteration the terminal node visited is 10. Using definition 3.2 we know its estimated value is the terminal reward associated with this node, 0. Node 10 and its estimated value, 0, are now added to the list of states. We next advance to the immediately previous node, 6. Here the estimated value is $\max\{2 + \bar{J}(10)\}$. Since node 10 is in the list of states, we retrieve its estimated value. The maximization does not include node 11 since it has not been visited. Therefore, the estimated value of node 6 is 2. This computational process continues until the initial node is encountered.

Proposition 3.5 stated sufficient conditions for the value at a prior state to equal the actual value of the state. The consequences of this proposition can be seen in Table (3.3). On the first iteration we know that the estimated value of node 10

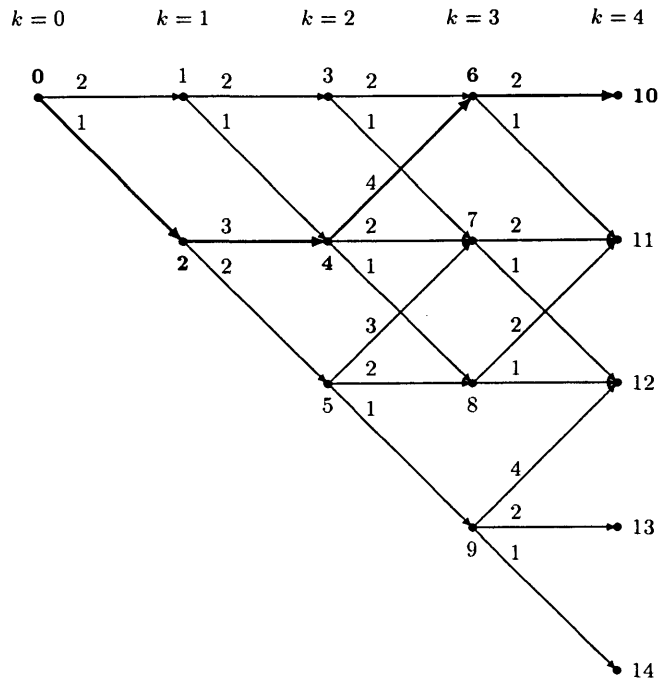


Figure 3.3: Example Graph for the MC-EV Algorithm

is equal to its value. Since the optimal decision at node 6 is to go to node 10, the estimated value at node 6 is equal to its value. Similarly, on the second iteration since the estimated value and value at node 6 are equal and the optimal decision at node 4 is to go to node 6, the estimated value at node 4 is equal to its value. Finally, on the third iteration, the estimated values at nodes 0 and 2 are equal to their respective values because of Proposition 3.5.

If the value of the problem is known and the estimated value of the initial state equals this value, Proposition 3.4 then applies and we can find the optimal decision sequence. For example, since the value of the longest path in the example graph is 10, after the third iteration we have enough information to find the optimal decision

Table 3.3: Information Propagation via Estimated Values for the Example Graph

Iteration	Node i	$J(i)$	$\bar{J}(i)$	$\bar{J}(i)$ Computation
1	10	0	0	0
	6	2	2	$\max\{2 + \bar{J}(10)\}$
	3	4	4	$\max\{2 + \bar{J}(6)\}$
	1	7	6	$\max\{2 + \bar{J}(3)\}$
	0	10	8	$\max\{2 + \bar{J}(1)\}$
2	11	0	0	0
	6	2	2	$\max\{2 + \bar{J}(10), 1 + \bar{J}(11)\}$
	4	6	6	$\max\{4 + \bar{J}(6)\}$
	1	7	7	$\max\{2 + \bar{J}(3), 1 + \bar{J}(4)\}$
	0	10	9	$\max\{2 + \bar{J}(1)\}$
3	11	0	0	0
	8	2	2	$\max\{2 + \bar{J}(11)\}$
	4	6	6	$\max\{4 + \bar{J}(6), 1 + \bar{J}(8)\}$
	2	9	9	$\max\{3 + \bar{J}(4)\}$
	0	10	10	$\max\{2 + \bar{J}(1), 1 + \bar{J}(2)\}$

sequence. The decision sequence is the argument of the estimated value computation at each node starting at the initial node.

In contrast to the dynamic programming solution of the example graph, it is not necessary to enumerate all possible states in the graph and compute their values. Note also that the optimal sequence of states was never selected on any iteration, as was necessary using the Monte Carlo search algorithm. Information computed previously leads to convergence of the algorithm on the third iteration.

3.3.4 Information Propagation

This algorithm remembers information from previous iterations by using the list of states. For instance, on iteration 2 at node 1 the maximization needs estimated

values from both nodes 3 and 4. Node 4 was visited on this iteration and consequently is in the list of states. Node 3 was not visited on this iteration. However, it was visited on the first iteration so it has an estimated value in the list of states.

The estimated value computation also propagates information between states. This occurs because the estimated value computation at any state directly depends on its descendent states estimated values. Estimated value information propagates from the terminal states back toward the initial state.

3.3.5 Convergence Analysis

If the optimal value of the problem is known, this algorithm stops on the iteration when the estimated value at the initial state equals the known value. As a consequence of Proposition 3.5, the algorithm has propagated enough information from the terminal states to the initial state so that the estimated value and value of the initial state are equal. Proposition 3.4 then provides a procedure for finding the optimal decision sequence.

Similar to the MC algorithm, if the optimal value is unknown then we can only make probabilistic statements about whether the algorithm has found an optimal sequence of decisions. Without knowing the underlying graph topology, imposed by the system dynamics, it is difficult to compute the probability of having found an optimal decision sequence. The MC algorithm requires that we know the number of decisions at each state on the optimal path to compute the probability of selecting an optimal decision sequence. The MC-EV requires more knowledge for the computation of this probability.

Consider the eight possible sequences of events that would lead to convergence of the MC-EV algorithm for the problem of Figure (3.3) shown in Table (3.4). The

dashes in the table mean select any admissible state at that point on the state trajectory. The first event in the table, selection of the optimal sequence of decisions on some iteration i of the algorithm, is the only case listed where the MC algorithm successfully finds the optimal sequence of decisions. However, the MC-EV algorithm succeeds in computing the correct value of the initial state upon the occurrence of any of the eight events. Clearly then, the number of ways for finding an optimal decision sequence is larger using the MC-EV algorithm. Since the probability of any event of the type 1-8 is greater than zero, it is more likely that the MC-EV algorithm will have found an optimal decision sequence in a fixed number of iterations than the MC algorithm.

Again, without loss of generality, suppose the optimal sequence of decisions is unique. Let $\{s_0^*, s_1^*, \dots, s_N^*\}$ be the optimal sequence of states. Define $s_{k,i}$ to be the event of selecting state s_k on iteration i . Now consider

$$\Pr\{s_{0,i_0}^*, s_{1,i_1}^*, \dots, s_{N,i_N}^*\}$$

If $i_0 = i_1 = \dots = i_N$, then this is the probability of selecting the optimal sequence of decisions at some iteration of the algorithm. If this occurs, both the MC algorithm and the MC-EV algorithm converge. If $i_0 \geq i_1 \geq \dots \geq i_N$ then, as a consequence of Proposition 3.5, the estimated value of the initial state will equal the value of the initial state.

Theorem 3.6: Suppose there exists a set of $n + 1$ partially optimal sequences of states of the following type where $n \geq 1$:

$$(1) \quad \{s_0^*, s_1, \dots, s_{k_n}, s_{k_n+1}^*, \dots, s_N^*\}$$

Table 3.4: Convergent Sequences of Events for the MC-EV Algorithm in the Example Graph

1.	$\{0, 2, 4, 6, 10\}$ on any iteration, say i
2.	$\{0, -, 4, 6, 10\}$ on iteration i_1 and $\{0, 2, -, -, -\}$ on iteration i_2 where $i_1 < i_2$
3.	$\{0, -, -, 6, 10\}$ on iteration i_1 and $\{0, 2, 4, -, -\}$ on iteration i_2 where $i_1 < i_2$
4.	$\{0, -, -, -, 10\}$ on iteration i_1 and $\{0, 2, 4, 6, -\}$ on iteration i_2 where $i_1 < i_2$
5.	$\{0, -, -, 6, 10\}$ on iteration i_1 and $\{0, -, 4, -, -\}$ on iteration i_2 and $\{0, 2, -, -, -\}$ on iteration i_3 where $i_1 < i_2 < i_3$
6.	$\{0, -, -, -, 10\}$ on iteration i_1 and $\{0, -, 4, 6, -\}$ on iteration i_2 and $\{0, 2, -, -, -\}$ on iteration i_3 where $i_1 < i_2 < i_3$
7.	$\{0, -, -, -, 10\}$ on iteration i_1 and $\{0, -, -, 6, -\}$ on iteration i_2 and $\{0, 2, 4, -, -\}$ on iteration i_3 where $i_1 < i_2 < i_3$
8.	$\{0, -, -, -, 10\}$ on iteration i_1 and $\{0, -, -, 6, -\}$ on iteration i_2 and $\{0, -, 4, -, -\}$ on iteration i_3 and $\{0, 2, -, -, -\}$ on iteration i_4 where $i_1 < i_2 < i_3 < i_4$

$$(2) \quad \{s_0^*, s_1, \dots, s_{k_{n-1}}, s_{k_{n-1}+1}^*, \dots, s_{k_n}^*, s_{k_n+1}, \dots, s_N\}$$

(:)

$$(n) \quad \{s_0^*, s_1, \dots, s_{k_1}, s_{k_1+1}^*, \dots, s_{k_2}^*, s_{k_2+1}, \dots, s_N\}$$

$$(n+1) \quad \{s_0^*, \dots, s_{k_1}^*, s_{k_1+1}, \dots, s_N\}$$

where $0 \leq k_1 < k_2 < \dots < k_n < N$. That is to say, the first sequence is optimal from state $s_{k_n+1}^*$ to state s_N^* , the second sequence is optimal from $s_{k_{n-1}+1}^*$ to $s_{k_n}^*$, and so on

until the $n + 1$ sequence, which is optimal from s_0^* to $s_{k_1}^*$ (the initial state, s_0^* is always on the optimal path.) Then the probability of finding the optimal decision sequence in a fixed number of iterations of the MC-EV algorithm exceeds the probability of finding the optimal decision sequence using the MC algorithm for the same number of iterations.²

Proof: The MC algorithm only converges when the optimal sequence of states $\{s_0^*, s_1^*, \dots, s_N^*\}$ is selected on some iteration. The repeated application of Proposition 3.5 in this case implies that the estimated value of the initial state s_0 is equal to its value and then Proposition 3.4 can be used to find the optimal path. Therefore the MC-EV algorithm also converges for this sequence.

Now we need only show that there exists some other sequence of state sequences, which can occur with positive probability, that then leads to convergence of the MC-EV algorithm. The existence of these sequences is asserted in the hypothesis of the theorem. Since there are only a finite number of states, each of these sequences has a probability of being selected that is bounded away from zero. If on some iteration a sequence of type 1 in the list occurs, then application of Proposition 3.5 implies that the estimated value at state $s_{k_n+1}^*$ is equal to its value. If on a subsequent iteration a sequence of type 2 occurs, then again application of Proposition 3.5 implies that the estimated value at the state $s_{k_{n-1}+1}^*$ is equal to its value. Continuing this process we see that on the last sequence the estimated value of state s_0 is equal to its value and the algorithm converges.

Since this sequence of events occurs with positive probability, for a fixed number

²The structure of the sequences, hypothesized to exist by this theorem, imply that the graph is not a tree. If the graph were a tree, then in order for the search process to visit the optimal terminal state it must traverse all other states on the optimal trajectory. To achieve a strict inequality for the probabilities, at least two paths from a state on the optimal trajectory to another state on the optimal trajectory must exist, one optimal the other not.

of iterations of each algorithm the probability of convergence of the MC-EV algorithm exceeds the convergence probability of the MC algorithm. ■

3.4 Monte Carlo Search with Estimated and Target Value Information Propagation

The reason for the slow convergence for the MC algorithm is the low probability associated with the restrictive event of choosing an optimal sequence. The MC-EV algorithm did not increase the probability of choosing the optimal sequence. The information propagation structure enables us to enlarge the set of decision sequences that lead to discovery of an optimal sequence.

In this section, we consider another algorithm that allows us to further enlarge the set of events that leads to discovery of an optimal sequence. This is accomplished by associating another piece of information with each state on the list of states. This information will be referred to as the target value of a state.

Definition 3.7: Again, let $S_k \subset S$ be the subset of states visited on the k th iteration of the algorithm. Then, given $J(s_0)$, the value of the initial state. The *target value* for any state $i \in S_k$ is

$$\hat{J}(i) = \min_{j \in \cup S_n} \{-r_{ji} + \hat{J}(j)\} \text{ for } i \in S_k \setminus \{s_0\}$$

$$\hat{J}(i) = J(s_0) \text{ for } i = s_0$$

Target values depend on the value of the initial state. In contrast to estimated values, the target value computation proceeds forward in time to the terminal state ■

encountered on the current iteration. Since this computation depends on knowing the value of the initial state, this algorithm will only be applicable to the problem of finding an optimal path when we know the value of the problem and are only searching for an optimal decision sequence.

3.4.1 Description of the Algorithm

The Monte Carlo search algorithm with target and estimated value information propagation (MC-TEV algorithm) is:

- Initialize a NULL list of states.
 - (1) Choose a random decision sequence.
 - (2) If a state visited using this decision sequence is not on the list of states, add it to the list.
 - (3) Compute the estimated value of each state on the current trajectory i.e., compute $\bar{J}(i)$ for each $i \in S_k$ according to Definition 3.2.
 - (4) Compute the target value of each state visited on the current trajectory i.e., compute $\hat{J}(i)$ for each $i \in S_k$ according to Definition 3.7.
 - (5) If (Converged) STOP, else go to (1).

It is easy to see that the target value of a state is always greater than or equal to its actual value (Lemma 3.8.) Further, the target value of a state can only equal its value if the state is on the optimal trajectory. A proposition, developed in the next section, will give the necessary conditions for the target value at a state to equal its actual value. Since the target value is greater than or equal to the value at a state and the estimated value at a state is less than or equal to this value, we will

know the actual value of this state when the target value equals the estimated value. (Proposition 3.4, discussed previously, provides a means to find the optimal decision sequence from this state onward. A similar proposition (3.12) provides a method to find the optimal decision sequence from the state with known value backward to the initial state.) This algorithm converges when, on some iteration, the target value and estimated value are equal at a state visited on the current simulation.

3.4.2 Properties of Target Values

The target value of the initial state equals its value. Its interpretation at other states is, that it equals the reward needed from this state onward if this state is on the optimal path. If all paths from the initial state to a given state have been traversed, then the target value for this state is the initial value minus the length of the longest path to this state. This is the value that needs to be realized starting at this state, if this state is on the optimal path.

The relationship of target values and actual values is captured in the following lemma.

Lemma 3.8: For all states $i \in S$

$$\hat{J}(i) \geq J(i)$$

Proof: Suppose it is not so. Then there exists a state s_i such that $\hat{J}(s_i) < J(s_i)$.

Hence, there is a path from s_0 to s_i for which

$$-\sum_{j=1}^i r_{s_{j-1}, s_j} + J(s_0) < J(s_i)$$

and as a result

$$J(s_0) < J(s_i) + \sum_{j=1}^i r_{s_{j-1}, s_j}$$

This implies, however, the existence of a decision sequence of larger value than the value of the initial state which contradicts the definition of $J(s_0)$. ■

Proposition 3.9: For any state $s_i \in S$,

$$J(s_0) - \hat{J}(s_i)$$

is the length of the longest path found, from the initial state s_0 to the state s_i .

Proof: Let S_i be the set of states for which r_{s_{i-1}, s_i} is finite, that is, the set of states that are accessible from $s_{i-1} \in S_{i-1}$. Consider

$$\begin{aligned} J(s_0) - \hat{J}(s_1) &= J(s_0) - \min_{s_0 \in S_0} \{-r_{s_0 s_1} + \hat{J}(s_0)\} \\ &= r_{s_0 s_1} \end{aligned}$$

This is obviously the maximum length between state s_0 and any state $s_1 \in S_1$. Now suppose that $J(s_0) - \hat{J}(s_{i-1})$ is the longest path between state s_0 and any state $s_{i-1} \in S_{i-1}$. Consider

$$\begin{aligned} J(s_0) - \hat{J}(s_i) &= J(s_0) - \min_{s_{i-1} \in S_{i-1}} \{-r_{s_{i-1} s_i} + \hat{J}(s_{i-1})\} \\ &= J(s_0) + \max_{s_{i-1} \in S_{i-1}} \{r_{s_{i-1} s_i} - \hat{J}(s_{i-1})\} \\ &= \max_{s_{i-1} \in S_{i-1}} \{r_{s_{i-1} s_i} + J(s_0) - \hat{J}(s_{i-1})\} \end{aligned}$$

This is just the dynamic programming equation for the longest path from state s_i to state s_0 . And now the result of the proposition holds by induction. ■

Proposition 3.10: For any state $s_i \in S$,

$$\hat{J}(s_i) = J(s_i)$$

if and only if $s_i \in S$ is on the optimal path.

Proof: Suppose s_i is on the longest path. As a consequence of the principle of optimality, the longest path from s_0 to a terminal state s_N is the sum of the longest path from s_i to s_N and the longest path from s_0 to s_i . Using proposition 3.9 and the definition of value at a state we have

$$J(s_0) = J(s_i) + J(s_0) - \hat{J}(s_i)$$

and then

$$\hat{J}(s_i) = J(s_i)$$

Now suppose that $\hat{J}(s_i) = J(s_i)$. Then

$$J(s_0) = J(s_i) + J(s_0) - \hat{J}(s_i)$$

and the longest path from s_0 to s_N is composed of the longest paths from s_i to s_N and from s_0 to s_i . Thus, by principle of optimality, s_i must be on the optimal path. ■

The next proposition defines a more general stopping criterion for the MC-TEV algorithm.

Proposition 3.11: For any state $i \in S$, if

$$\hat{J}(i) = \bar{J}(i)$$

then

$$J(i) = \hat{J}(i) = \bar{J}(i)$$

Proof: The result immediately follows from lemmas 3.3 and 3.8. ■

The procedure for moving from the state where convergence of the algorithm occurred backward along the optimal path arises from the next proposition.

Proposition 3.12: For any state $m \in (S - s_0)$ and $i \in S$, if $\hat{J}(m) = \bar{J}(m)$ and

$$i = \arg \min_j \{-r_{jm} + \hat{J}(j)\}$$

then $\hat{J}(i) = J(i)$.

Proof: Since i is the argument of the target value computation we have

$$\hat{J}(m) = -r_{im} + \hat{J}(i)$$

then, using proposition 3.11,

$$J(m) = -r_{im} + \hat{J}(i)$$

and rearranging terms

$$\hat{J}(i) = r_{im} + J(m)$$

From lemma 3.8 we have $J(i) \leq \hat{J}(i)$. Therefore

$$J(i) \leq r_{im} + J(m)$$

Using the definition of the value of state i

$$\max_j \{r_{ij} + J(j)\} \leq r_{im} + J(m)$$

This maximization is achieved by selecting $j = m$ and cannot be larger for any other $j \in S$. Then

$$J(i) = r_{im} + J(m)$$

And consequently $\hat{J}(i) = J(i)$. ■

The final proposition describes the necessary conditions for propagation of target value information along the optimal path.

Proposition 3.13: For any state $i \in S \setminus \partial S$, suppose that the target value and actual value are equal i.e., $\hat{J}(i) = J(i)$, and the optimal decision at state i is to go to state $m \in S$

$$m = \arg \max \{r_{ij} + J(j)\}$$

then $\hat{J}(m) = J(m)$.

Proof: Since the optimal decision at state i is to go to state m

$$J(i) = r_{im} + J(m)$$

then

$$\hat{J}(i) = r_{im} + J(m)$$

and rearranging terms results in

$$J(m) = -r_{im} + \hat{J}(m)$$

From lemma 3.8 we have $J(m) \leq \hat{J}(m)$. Therefore using the definition of target value we have

$$\min_j \{-r_{jm} + \hat{J}(j)\} \geq -r_{im} + \bar{J}(i)$$

This minimization is achieved by selecting $j = m$ and cannot be smaller for any other $j \in S$. Then

$$\hat{J}(m) = -r_{im} + \hat{J}(i)$$

And consequently $\hat{J}(m) = J(m)$. ■

Propositions 3.12 and 3.13 are analogous to Propositions 3.4 and 3.5. Proposition 3.4 stated the necessary conditions for finding the optimal decision at a given state in terms of estimated values. Likewise, Proposition 3.12 provides a means for finding the optimal decision leading to a state when target values are considered. Consider states i and m in Figure (3.4). Proposition 3.12 states that if the target value and estimated value of state m are equal and if i is the argument of the target value computation, then the target value at i equals its value. This implies the decision to go from i to m is optimal. Similar to Proposition 3.4, which provides a procedure from moving from m to a terminal state on an optimal path, Proposition 3.12 supplies a procedure for moving from m along an optimal path to the initial

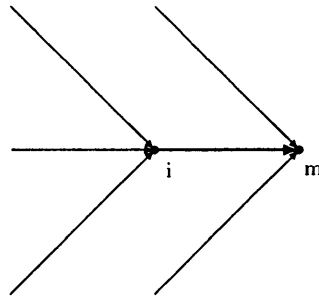


Figure 3.4: Internodal Graph Connections and Target Value Computations

state.

Proposition 3.13 states the conditions which must occur for the target value to equal its value. If, at i in the graph depicted in Figure (3.4), the target value equals the value of this state and if the optimal decision at i is to go to m , then the target value of m equals its value. This implies that the target value at a descendent state can only equal the value of that state if the target value of the ancestral state equals its value.

3.4.3 Solution of the Example Graph

In this section we apply the MC-TEV algorithm to the graph shown in Figure (3.1) copied in Figure (3.5) for ease of reference. Consider two potential iterations of the algorithm that result in the following trajectories,

1. $\{0, 2, 4, 8, 12\}$
2. $\{0, 1, 4, 6, 10\}$

Table (3.5) gives the computations leading to target values for the nodes visited on each iteration. The actual value and estimated value are also listed for reference.

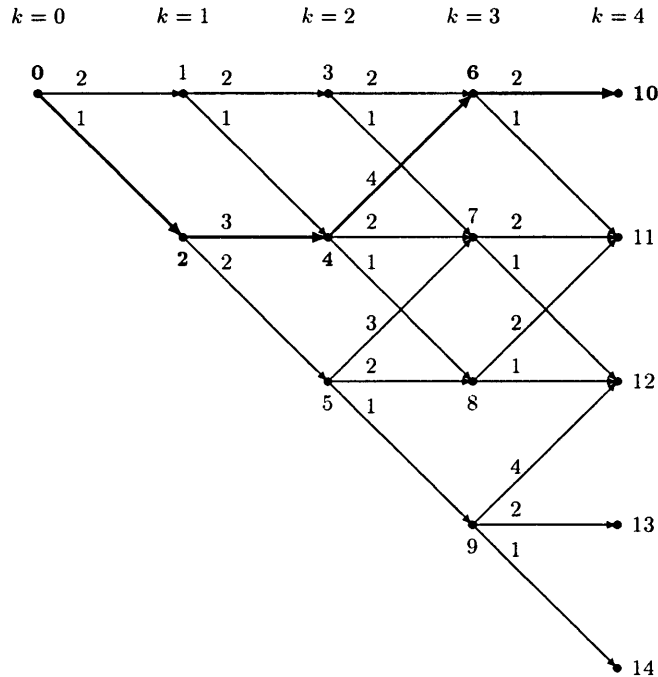


Figure 3.5: Example Graph for the MC-TEV Algorithm

Note that these computations proceed from the known value of the initial node and terminate when a terminal node is encountered.

Consider node 4 on iteration 1. The target value for this node is $\min\{-3 + \hat{J}(2)\}$. Since node 1 has not been encountered previously, it is not considered. Node 2 has been visited and has a target value of 9. Therefore the result of the minimization is 6.

Now consider node 4 on iteration 2. Now both node 1 and 2 have been previously visited. They have target values of 8 and 9, respectively. Nonetheless the result of the minimization remains the same, 6. However here the target value equals the estimated value. This is sufficient to imply convergence of our algorithm. The target

Table 3.5: Information Propagation via Estimated and Target Values for the Example Graph

Iteration	Node i	$J(i)$	$\bar{J}(i)$	$\hat{J}(i)$	Computation
1	0	10	6	10	10
	2	9	5	9	$\min\{-1 + \hat{J}(0)\}$
	4	6	2	6	$\min\{-3 + \hat{J}(2)\}$
	8	2	1	5	$\min\{-1 + \hat{J}(4)\}$
	12	0	0	4	$\min\{-1 + \hat{J}(8)\}$
2	0	10	9	10	10
	1	7	7	8	$\min\{-2 + \hat{J}(0)\}$
	4	6	6	6	$\min\{-1 + \hat{J}(1), -3 + \hat{J}(2)\}$
	6	2	2	2	$\min\{-4 + \hat{J}(4)\}$
	10	0	0	0	$\min\{-2 + \hat{J}(6)\}$

values have propagated information on the value of intermediate nodes on the optimal path.

3.4.4 Information Propagation

Similar to the MC-EV algorithm, the MC-TEV algorithm remembers information from previous iterations of the algorithm. The target value at node 4 on iteration 2 requires information on the target value at nodes 1 and 2. Node 1 was visited on the second iteration so its target value is accessible. Node 2 was visited previously, therefore it has an accessible target value stored in the list of states.

The dependency of the target value computation on ancestral states, starting with the initial state, shows that the target value computation is propagating information forward in time. Target values and estimated values are two computational structures that simultaneously allow for information to propagate both forward and

backward. Whereas, using the MC-EV algorithm it was necessary for the estimated value information to propagate backward to the initial state, the target value computation allows for information to be propagated forward and compared with estimated values to determine if enough information has been computed to solve the problem.

3.4.5 Convergence Analysis

The MC-TEV algorithm is applicable to only those problems where the value of the optimal path is known, the actual path is unknown. This algorithm converges when the target value and estimated value of a state are equal. This is because of Proposition 3.11, which implies these values must equal the actual value of that state. This state must then be on optimal path as a consequence of Proposition 3.10. The repeated use of Propositions 3.4 and 3.12 provide the mechanisms for finding the optimal path from the state where equality of the target value and estimated value occurred.

From the discussion of the proposed solution of the example graph in the previous section, we see that there are more ways for the MC-TEV algorithm to converge than the MC-EV algorithm. The two iterations, sufficient for convergence of the MC-TEV algorithm, are clearly insufficient for convergence of either the MC algorithm or the MC-EV algorithm. The MC-TEV algorithm has again enlarged the set of events that lead to discovery of an optimal decision sequence.

As before, assume that the optimal decision sequence is unique and let $\{s_0^*, s_1^*, \dots, s_N^*\}$ be the optimal state trajectory. Let $s_{k,i}$ be the selection of state s_k on iteration i of the algorithm. Again consider

$$\Pr\{s_{0,i_0}^*, s_{1,i_1}^*, \dots, s_{N,i_N}^*\}$$

As was argued previously, this is the probability of discovery of the optimal decision sequence of the algorithm in question, under specified restrictions on the i_k 's. For the MC-TEV algorithm these restrictions take the form of one of the following conditions:

1. $i_N \leq \dots \leq i_1$
2. $i_N \leq \dots \leq i_k$ and $i_1 \leq \dots \leq i_k$ for $k \in \{2, \dots, N - 1\}$
3. $i_1 \leq \dots \leq i_N$

These relations determine the order for finding states on the optimal trajectory. Case 1 says that discovery of s_N^* must occur before s_{N-1}^* and so on until finding s_1^* . The order of discovery of states on the optimal trajectory for case 3 is the reverse of case 1. Finding state s_1^* must happen before finding s_2^* until the final discovery of s_N^* . For case 2, discovery of states on the optimal trajectory occurs simultaneously starting at both the initial state and the optimal terminal state. Discovery of state s_N^* followed by finding s_{N-1}^* and so on until finding state s_k^* , happens at the same time as the discovery of states s_1^* followed by s_2^* and finally encountering state s_k^* .

For case 1, the algorithm converges because the target value of the initial state (the value of the problem) and its estimated value are equal. In essence the propagation of useful target value information did not occur. Note that this is the condition for convergence of the MC-EV algorithm. Case 3 corresponds to the opposite circumstance. The algorithm propagates the target value to the terminal state on the optimal path. In this case useful estimated value information propagation did not occur. Case 2 is the most likely circumstance. Here both the target and estimated values propagate to some intermediate state on the optimal path.

We will say that the algorithm has visited the optimal states in reverse order if the case 1 inequality is satisfied. Both the MC-TEV and MC-EV algorithms converge

then. If case 3 occurs, the optimal states were visited in forward order. For case 2, the states on partially optimal trajectories were visited in both forward and reverse order. For cases 2 and 3 the MC-TEV algorithm converges, but the MC-EV algorithm fails to converge.

In the next theorem we formalize these results and show that the MC-TEV has a larger probability of convergence in any fixed number of iterations than the MC-EV algorithm. As a consequence of Theorem 3.6 it also is more likely to have converged in a fixed number of iterations than the MC algorithm.

Theorem 3.14: Suppose there exists a set of $n + 1$ partially optimal sequences³ of the following type where $n \geq 1$:

$$(1) \quad \{s_0^*, s_1, \dots, s_{k_n}, s_{k_n+1}^*, \dots, s_N^*\}$$

$$(2) \quad \{s_0^*, s_1, \dots, s_{k_{n-1}}, s_{k_{n-1}+1}^*, \dots, s_{k_n}^*, s_{k_n+1}, \dots, s_N\}$$

$$(:) \quad \cdot$$

$$(n) \quad \{s_0^*, s_1, \dots, s_{k_1}, s_{k_1+1}^*, \dots, s_{k_2}^*, s_{k_2+1}, \dots, s_N\}$$

$$(n + 1) \quad \{s_0^*, \dots, s_{k_1}^*, s_{k_1+1}, \dots, s_N\}$$

where $0 \leq k_1 < k_2 < \dots < k_n < N$. Then the probability of finding the optimal decision sequence in a fixed number of iterations of the MC-TEV algorithm exceeds the probability of finding the optimal decision sequence using the MC-EV algorithm for the same number of iterations.

Proof: As a consequence of Proposition 3.5 the MC-EV algorithm converges only

³See the footnote following Theorem 3.6.

when the states on the optimal trajectory are visited in reverse order. In this case, the MC-TEV algorithm also converges because the estimated value, target value, and actual value of the initial state are all equal.

We need now only show that there exists some other sequence of state sequences that then leads to convergence of the MC-TEV algorithm but does not lead to convergence of the MC-EV algorithm. Again, the existence of these sequences is asserted in the hypothesis of the theorem. If a sequence of type 2 follows a type 1 sequence, as a consequence of proposition 3.5, estimated values equal actual values beginning at some intermediate state, $s_{k_{n-1}+1}^*$, through the optimal terminal state, s_N^* . Concurrently, if a sequence of type n follows the selection of a type $n+1$ sequence, the target values on the optimal path beginning with the initial state through some intermediate state, $s_{k_2}^*$, will equal their actual values as a consequence of Proposition 3.13. Continuing this process we find that at some intermediate state on the optimal trajectory the target value and estimated values are equal. Proposition 3.11 then implies that the estimated value, target value, and actual value are all equal. The MC-TEV algorithm has then converged and Propositions 3.4 and 3.12 provide a means to find the optimal decision sequence. In this case the MC-EV algorithm fails to converge since the estimated value of the initial state does not equal its value (Proposition 3.5 was not satisfied at the initial state.)

Since this sequence of events occurs with positive probability; in a fixed number of iterations, the probability of convergence of the Monte Carlo search algorithm with target and estimated value information propagation exceeds the probability of convergence of the Monte Carlo search algorithm with estimated value information propagation in the same number of iterations.



3.5 Monte Carlo Search with Complete Estimated Value Information Propagation

Convergence of the three algorithms, discussed previously, occurs because the search procedure visits states on the optimal trajectory in certain orders. For the MC algorithm all states on the optimal trajectory must be selected on some iteration of the algorithm. Adding the computation of estimated values to the MC algorithm allows us to relax this condition so that the algorithm need only visit the states on the optimal trajectory in reverse order. When we add target value information to the MC-EV algorithm we are able to further relax the conditions on the order of visiting the states on the optimal trajectory. Here the algorithm can visit the states on the optimal trajectory in both the forward and reverse orders and then have these two partially optimal trajectories meet at some intermediate state on the optimal trajectory.

A natural question to ask at this point is: Can we devise an algorithm that imposes no constraints on the order of visiting the states on the optimal trajectory? The answer to this question is yes. To see what is necessary for such an algorithm again consider the example graph in Figure (3.1). Recall that the optimal sequence of nodes is $\{0, 2, 4, 6, 10\}$. Consider what happens when the MC-EV algorithms search procedure selects the following two trajectories.

1. $\{0, 2, 4, 7, 11\}$
2. $\{0, 1, 4, 6, 10\}$

Figure (3.6) depicts these trajectories over the pertinent part of the graph in Figure (3.1). (The second path is shown using a slightly thicker line than the first path.) Table (3.6) shows the estimated values of the states visited on these two paths.

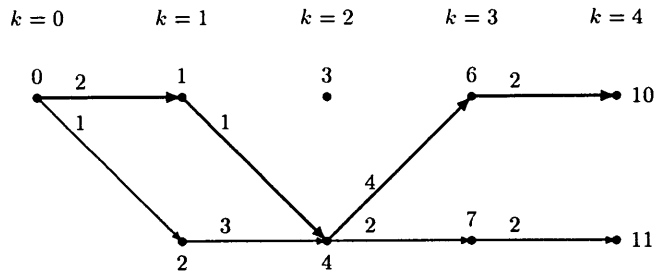


Figure 3.6: Complete Estimated Values for the Example Graph

Table 3.6: Estimated Values for the Example Algorithm Iterations in the Example Graph

Iteration	Node i	$\bar{J}(i)$	Iteration	Node i	$\bar{J}(i)$
1	0	8	2	0	8
	2	7		1	7
	4	4		4	6
	7	2		6	2
	11	0		10	0

The first iteration visits the nodes $\{0, 2, 4\}$ on the optimal path. However, the estimated values of these nodes are not equal to the values of these nodes since the descendent nodes on the optimal path, $\{6, 10\}$, have not yet been visited. On the second iteration, the estimated values of nodes $\{4, 6, 10\}$ now equal their actual values. However, since node $\{2\}$ was not visited on this iteration, the estimated value of both nodes $\{2\}$ and $\{0\}$ do not equal their values. For the MC-EV algorithm to converge, the search procedure must again visit node $\{2\}$. If the MC-EV algorithm had recomputed the estimated value of node $\{2\}$, $\bar{J}(2) = 3 + \bar{J}(4)$, on the second iteration, the estimated value computation of the initial node would have equaled its

value and then the algorithm would have converged.

This suggests an algorithm where we compute the estimated value of every state that has been visited on every iteration of the algorithm. We call this process complete estimated value information propagation and the computed values of the states, complete estimated values.

Definition 3.15: Let $S_k \subset S$ be the subset of states visited on the k th iteration of the algorithm. Further, let $\cup S_k$ be the set of all states visited through iteration k of the algorithm. Denote the set of terminal states encountered through iteration k of the algorithm $\partial \cup S_k$. We then define *complete estimated values* as

$$\mathcal{J}(i) = r_i \text{ for } i \in \partial \cup S_k$$

$$\mathcal{J}(i) = \max_{j \in \cup S_k} \{r_{ij} + \mathcal{J}(j)\} \text{ for } i \in \cup S_k \setminus \partial \cup S_k$$

■

The difference between complete estimated values and estimated values is the set of states for which estimated values are computed. The estimated value computation occurs only for those states visited on the current iteration. Complete estimated values are computed for every state that has been visited on any iteration.

3.5.1 Description of the Algorithm

The Monte Carlo search algorithm with complete estimated value information propagation (MC-CEV algorithm) is:

- Initialize a NULL list of states.
 - (1) Choose a random decision sequence.

- (2) If a state visited using this decision sequence is not in the list of states, add it to the list.
- (3) Compute the complete estimated value of every state in the list of states using Definition 3.15.
- (4) If (Converged) STOP, else go to (1).

Complete estimated values share the same properties as estimated values. Here again, the complete estimated value of a state is always less than or equal to the actual value of the state (Lemma 3.3 applies.) If we know the value of the problem, then we can say the algorithm has converged when the complete estimated value of the initial state equals this value (we then use Proposition 3.4 to find the optimal decision sequence.) Similar to the MC-EV algorithm, if the value of the problem is unknown we can only make a probabilistic statement about convergence of the algorithm. The complete estimated value merely expands the domain of the estimated value computations to include all the states visited on any iteration instead of only the states visited on the current iteration.

The complete estimated value computation, described in Definition 3.15, appears to be quite similar to the DP algorithm's computation of the actual values of states. The primary difference lies in the fact that we compute complete estimated values on only a subset $\cup S_n$, of the entire state space S , where $\cup S_n$ includes states previously visited by the algorithm. In effect, complete estimated values are the actual values for the sequential decision making subproblem defined on $\cup S_n$. In other words, we propose to perform dynamic programming over the part of the state space already held in computer memory. The solution of this subproblem finds the optimal decision sequence among all valid sequences in $\cup S_n$.

3.5.2 Solution of the Example Graph

To see how the MC-CEV algorithm works, again consider the example graph in Figure (3.1) again reproduced in Figure (3.7) for convenient reference. Suppose that three iterations of the search procedure results in the trajectories:

1. {0, 1, 4, 7, 11}
2. {0, 2, 5, 9, 14}
3. {0, 1, 3, 6, 10}

Table (3.7) contains the actual, estimated, and target values of the nodes the algorithm visits using these trajectories (nodes on the optimal trajectory are in bold.) If we examine the sequence of estimated values at the initial node we see that it never equals the actual value of the initial node on any iteration. Therefore the MC-EV algorithm fails to converge after iterating through these three trajectories. Also, the estimated value and target values are never equal at any state on any iteration of the algorithm. So the MC-TEV algorithm does not converge after these iterations. The reason for the failure of both these algorithms is that the nodes on the optimal trajectory have not been visited in an order that would lead to convergence of either algorithm. Note however, all the nodes on the optimal trajectory have been visited on some iteration of the algorithm.

Table (3.8) shows the computations for the complete estimated values of the nodes encountered from the three given trajectories (nodes on the optimal trajectory are again in bold.) Since we must perform this computation at all nodes visited on the current and previous iterations of the algorithm, the number of computations increases as new nodes are added to the current list of states. In this case, on the first

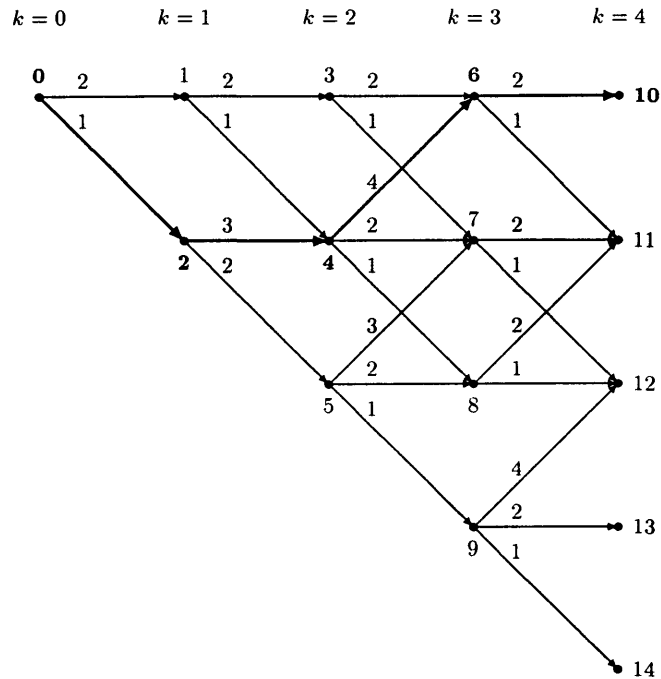


Figure 3.7: Example Graph for the MC-CEV Algorithm

iteration the MC-CEV algorithm computes complete estimated values at the nodes

$$US_1 = \{0, 1, 4, 7, 11\}$$

The second iteration results in complete estimated values for the nodes in the set

$$\begin{aligned} US_2 &= \{0, 1, 4, 7, 11\} \cup \{0, 2, 5, 9, 14\} \\ &= \{0, 1, 2, 4, 5, 7, 9, 11, 14\} \end{aligned}$$

Table 3.7: Actual, Estimated, and Target Values for the Example Algorithm Iterations in the Example Graph

Iteration	Node i	$J(i)$	$\bar{J}(i)$	$\hat{J}(i)$
1	0	10	7	10
	1	7	5	8
	4	6	4	7
	7	2	2	5
	11	0	0	3
2	0	10	8	10
	2	9	7	9
	5	5	5	7
	9	4	1	6
	14	0	0	5
3	0	10	8	10
	1	7	6	8
	3	4	4	6
	6	2	2	4
	10	0	0	2

On the third iteration complete estimated values are computed for all nodes in

$$\begin{aligned} US_3 &= \{0, 1, 4, 7, 11\} \cup \{0, 2, 5, 9, 14\} \cup \{0, 1, 3, 6, 10\} \\ &= \{0, 1, 2, 3, 4, 5, 6, 7, 9, 10, 11, 14\} \end{aligned}$$

Let $S^* = \{0, 2, 4, 6, 10\}$ be the set of nodes on the optimal trajectory, then the following relations hold among these sets

$$S^* \not\subset US_1 \text{ and } S^* \not\subset US_2 \text{ but } S^* \subset US_3$$

When the current subproblem contains the nodes in the optimal trajectory, the MC-CEV algorithm converges.

The benefit of the MC-CEV algorithm is that we no longer have constraints on the order that the algorithm must visit the states on the optimal trajectory. The cost is added computation on each iteration of the algorithm. Generally, however, the additional cost is not prohibitive. These value computations are relatively inexpensive since these states are already in computer memory.

3.5.3 Convergence Analysis

The MC-CEV algorithm removes the necessity of visiting the states on the optimal trajectory in any particular order. The algorithm requires only that each state on the optimal trajectory be visited at least once.

Proposition 3.16: Suppose that

$$\cup S_n = \{s_0^*, s_1^*, \dots, s_N^*\}$$

that is the list of states contains only the states on an optimal trajectory. Then the complete estimated value of the initial state equals the actual value of the initial state,

$$J(s_0) = \mathcal{J}(s_0)$$

Proof: The result follows immediately from the repeated application of Proposition 3.5 to each state beginning with s_{N-1}^* and ending with s_0^* . ■

Table 3.8: Complete Estimated Values for the Example Algorithm Iterations in the Example Graph

Iteration	Node i	$J(i)$	$\mathcal{J}(i)$	$\mathcal{J}(i)$ Computation
1	0	10	7	$\max\{2 + \mathcal{J}(1)\}$
	1	7	5	$\max\{1 + \mathcal{J}(4)\}$
	4	6	4	$\max\{4 + \mathcal{J}(7)\}$
	7	2	2	$\max\{2 + \mathcal{J}(11)\}$
	11	0	0	0
2	0	10	8	$\max\{2 + \mathcal{J}(1), 1 + \mathcal{J}(2)\}$
	1	7	5	$\max\{1 + \mathcal{J}(4)\}$
	2	9	7	$\max\{3 + \mathcal{J}(4), 2 + \mathcal{J}(5)\}$
	4	6	4	$\max\{2 + \mathcal{J}(7)\}$
	5	5	5	$\max\{3 + \mathcal{J}(7), 1 + \mathcal{J}(9)\}$
	7	2	2	$\max\{2 + \mathcal{J}(14)\}$
	9	4	1	$\max\{1 + \mathcal{J}(14)\}$
	11	0	0	0
	14	0	0	0
	3	0	10	10
1		7	7	$\max\{2 + \mathcal{J}(3), 1 + \mathcal{J}(4)\}$
2		9	9	$\max\{3 + \mathcal{J}(4), 2 + \mathcal{J}(5)\}$
3		4	4	$\max\{2 + \mathcal{J}(6), 1 + \mathcal{J}(7)\}$
4		6	6	$\max\{4 + \mathcal{J}(6), 2 + \mathcal{J}(7)\}$
5		5	5	$\max\{3 + \mathcal{J}(7), 1 + \mathcal{J}(9)\}$
6		2	2	$\max\{2 + \mathcal{J}(10), 1 + \mathcal{J}(11)\}$
7		2	2	$\max\{2 + \mathcal{J}(11)\}$
9		4	1	$\max\{1 + \mathcal{J}(14)\}$
10		0	0	0
11		0	0	0
14		0	0	0

Proposition 3.17: Suppose that

$$\cup S_n = \{s_0^*, s_1^*, \dots, s_N^*\} \cup \{s_i : \text{for some } s_i \in S\}$$

that is the list of states contains the states on an optimal trajectory and some other states in the state space. Then

$$J(s_0) = \mathcal{J}(s_0)$$

which is to say, the complete estimated value equals the value of the problem.

Proof: By selecting the optimal states we see that $\mathcal{J}(s_0)$ cannot be less than $J(s_0)$ or

$$\mathcal{J}(s_0) \geq J(s_0)$$

Now if $\mathcal{J}(s_0) > J(s_0)$ there would exist some sequence of states that would lead to a larger value than an optimal sequence. But this cannot be true. Therefore the proposition holds. ■

If all the states on an optimal trajectory are not in the list of states, the complete estimated value will be less than the value of the problem.

Proposition 3.18: Suppose that the optimal sequence of states is unique. Further suppose

$$\cup S_n = \{s_0^*, s_1^*, \dots, s_N^*\} \cup \{s_i : \text{for some } s_i \in S\} \setminus \{s_i^* : \text{for some } s_i^* \in S\}$$

or the set of states does not contain all the states on the optimal trajectory but may

contain states not on the optimal trajectory. Then

$$\mathcal{J}(s_0) < J(s_0)$$

or the complete estimated value of the initial state is less than the value of the problem.

Proof: $\mathcal{J}(s_0)$ cannot be greater than $J(s_0)$ because this would imply the existence of a sequence with value larger than the value of the problem which cannot be true. Also, $\mathcal{J}(s_0)$ cannot equal $J(s_0)$ since this would imply the existence of a sequence with equal value to the value of the problem or an optimal sequence. Therefore $\mathcal{J}(s_0)$ must be strictly less than $J(s_0)$. ■

Using this algorithm we become more concerned with search procedures that are more likely to encounter states on the optimal trajectory. We no longer need be concerned with the order that the algorithm visits the states on the optimal trajectory.

Theorem 3.19: Suppose there exists a set of $n + 1$ partially optimal sequences⁴ of states of the following type where $n \geq 2$:

$$(1) \quad \{s_0^*, s_1, \dots, s_{k_n}, s_{k_n+1}^*, \dots, s_N^*\}$$

$$(2) \quad \{s_0^*, s_1, \dots, s_{k_{n-1}}, s_{k_{n-1}+1}^*, \dots, s_{k_n}^*, s_{k_n+1}, \dots, s_N\}$$

$$(\vdots) \quad \quad \quad \vdots$$

⁴Similar to Theorems 3.6 and 3.14, these sequences imply that the graph in question is not a tree. To insure strict inequality it is now necessary for there to be at least two states on the optimal trajectory from which it is possible to select paths that are not optimal but eventually return to the optimal path.

$$(n) \quad \{s_0^*, s_1, \dots, s_{k_1}, s_{k_1+1}^*, \dots, s_{k_2}^*, s_{k_2+1}, \dots, s_N\}$$

$$(n+1) \quad \{s_0^*, \dots, s_{k_1}^*, s_{k_1+1}, \dots, s_N\}$$

Then the probability of finding the optimal decision sequence in a fixed number of iterations of the MC-CEV algorithm exceeds the probability of finding the optimal decision sequence using the MC-TEV algorithm for the same number of iterations.

Proof: Similar to the proofs of Theorems 3.6 and 3.14 we will show that the MC-CEV algorithm converges whenever the MC-TEV algorithm does. Then we will produce a sequence of state sequences where the MC-TEV algorithm fails to converge but the MC-CEV algorithm does converge.

As was argued before, the MC-TEV algorithm converges whenever the iteration indices for the states on the optimal trajectory, $s_{k,i}^*$, satisfy one of the following conditions.

1. $i_N \leq \dots \leq i_1$
2. $i_N \leq \dots \leq i_k$ and $i_1 \leq \dots \leq i_k$ for $k \in \{2, \dots, N-1\}$
3. $i_1 \leq \dots \leq i_N$

For each of these conditions the MC-CEV algorithm converges as a consequence of Proposition 3.17, all the states in the optimal trajectory are on the list of states.

The existence of a set of sequences where the MC-CEV algorithm converges but the MC-TEV algorithm fails to converge is asserted in the hypothesis of the theorem. Since the probability of selecting these sequences in an order for which the MC-TEV algorithm fails to converge is strictly greater than zero, the probability of finding the

optimal trajectory using the MC-CEV algorithm exceeds the probability of finding the optimal trajectory with the MC-TEV algorithm in any fixed number of iterations. ■

3.6 Discussion

In this chapter we have considered in detail the “test” part of the generate-and-test algorithms for deterministic sequential decision making problems. So that we could carefully consider this part of the algorithm, the “generate” part of the algorithm was quite simple, a Monte Carlo search. We found that by using information structures that propagate value information among the states of the dynamic system we were able to increase the probability of convergence of our algorithm when run for a fixed number of iterations. Quantification of the increase in probability is difficult because of its dependency on the underlying graph topology imposed by the dynamics of the system.

All of the algorithms require the search procedure to visit states on an optimal trajectory. The differing algorithms imposed different constraints on the order in which the states could be visited. However, all the algorithms require visitation of every state on an optimal trajectory. Therefore, in order to increase the probability of convergence of the algorithm in a fixed number of iterations we must next examine search processes that increase the probability of visiting states on the optimal trajectory.

The four algorithms, presented here, require differing computational efforts. Each algorithm uses a Monte Carlo search process to generate a simulation of the dynamic system. The MC algorithm needs only evaluate the length of this path and compare it to the length of the longest path found up to the current iteration of the

algorithm. The MC-EV algorithm requires a list of states visited on all iterations of the algorithm. Then we need to perform maximizations at each state in the current simulation. These maximizations use value information stored in the list of states.

The MC-TEV requires the same list of states and the same maximization process. Additional computation is introduced through the computation of target values. Here a minimization over all states leading to the state under consideration must be performed. Typically, we cannot examine a state and find out what other states might lead to it. (The Markov property of the problems considered implies that knowledge of how we got to a state need not be known. The description of each state is to contain all the information necessary for optimization from this state forward.) To implement the target value computation we examine the list of states and select those states that immediately precede the state under consideration. For these states only, the target value minimization is performed.

The MC-CEV algorithm also requires a list of states. For this algorithm, however, instead of performing the estimated value computation for only the states visited during the current simulation we have to recompute estimated values for every state in the list of states. Therefore, the computational burden of performing the estimated value maximization increases with every iteration of the algorithm (assuming at least one new state is added on every iteration.) However, we do not need to compute target values using this algorithm. Since complete estimated values are computed for each state in the list of states we may envision an algorithm that runs more than one simulation of the dynamic system before computation of the complete estimated values. We might structure this algorithm like this:

- Initialize a NULL list of states.
- Set $k = 1$.

- (1) Choose n_k random decision sequence(s).
- (2) If a state visited using these decision sequences is not in the list of states, add it to the list.
- (3) Compute the complete estimated value of every state in the list of states.
- (4) If (Converged) STOP, else set $k \leftarrow k + 1$ and go to (1).

where the n_k are positive integers dictating how many simulations to run on each iteration of the algorithm.

Chapter 4

THE VALUE-BASED SEARCH PROCEDURE

The Monte Carlo search procedure, discussed in Chapter 3, provides a simple mechanism for selecting decisions at states of the dynamic system. It led to four computational algorithms based on differing methods of information propagation,

1. No information propagation
2. Estimated value information propagation
3. Target and estimated value information propagation
4. Complete estimated value information propagation

Here we will discuss a value-based search algorithm that Tolwinski and Underwood [10] use for estimating the optimal evolution of an open pit mine. The search procedure they use attempts to direct decision making into areas of the state space where large rewards might be found.

4.1 Description of the Value-Based Search Procedure

The value-based search procedure selects decisions at the nodes of a graph by examining three attributes of the potential decisions. It considers the best decision found so far, the least explored decision, and the best unexplored decision.

Consider the i th node in the graph. Suppose there are k decisions. With each decision at this node associate three numbers, the immediate reward, the number of

Table 4.1: Data used in the Value-Based Search Procedure

Decision	Reward	Number of times the decision has been tried	Estimated Value at Destination
1	r_1	n_1	v_1
2	r_2	n_2	v_2
3	r_3	n_3	v_3
\vdots	\vdots	\vdots	\vdots
k	r_k	n_k	v_k

times the decision has been made previously, and the estimated value at the node resulting from this decision. Table (4.1) shows the structure of this data.

Since the number of decisions, k , is finite, the decisions can be ordered so that $r_1 \geq r_2 \geq \dots \geq r_k$. If the node has not been visited previously, then $n_j = 0$ for $j = 1, 2, \dots, k$ and v_j is undefined.

Let S be the set of decisions at node i ,

$$S = \{1, 2, \dots, k\}$$

Then define S_{old} , a subset of S where the associated number of times the decision has been tried is greater than zero,

$$S_{old} = \{j : n_j > 0\}$$

and S_{new} , the complement of S_{old} , the set of decisions that have not been tried,

$$S_{new} = \{j : n_j = 0\}$$

The value-based search selects between the best decision found so far, say $x \in S_{old}$, the least explored decision $y \in S_{old}$, and the best unexplored decision $z \in S_{new}$.

The best decision x found so far is

$$x = \arg \max_j \{r_j + v_j : j \in S_{old}\}$$

The number of times this decision has been tried is n_x . The decision that has been least explored is

$$y = \arg \min_j \{n_j : j \in S_{old}\}$$

It has been explored n_y times. The best unexplored decision is

$$z = \min_j \{j : j \in S_{new}\}$$

which is a consequence of the ordering of the decisions.

The value-based search uses a stochastic mechanism to choose one of these three potential decisions. The procedure by which it selects a decision is

1. Select x , the best decision found so far, with probability

$$\frac{1}{(1 + (n_x - n_y))^{p_x}}$$

else select decision y , the least explored decision in S_{old} . Call the resulting decision, v .

2. If S_{new} is empty, then

- (a) the value-based search chooses decision v and goes to Step 3.

If S_{new} is not empty, then

(b) the value-based search selects decision v with probability

$$\frac{1}{(1 + n_y)^{p_y}}$$

otherwise it selects decision z in the set S_{new} . Then it removes z from S_{new} , places it in S_{old} , and goes to Step 3.

3. Increment the number of time the resulting decision has been tried.

4. Make the selected decision and go to the next node in the graph.

The parameters p_x and p_y are real numbers, strictly greater than zero, that determine the rate at which the value-based search explores rarely explored or unexplored decisions.

Initially, before any decision has been tried, $S_{old} = \emptyset$. In this case $x = y = z = 1$ and $n_x = n_y = 0$. Then the value-based search explores decision $x = 1$ with probability equal to one since

$$\frac{1}{(1 + (0 - 0))^{p_x}} = 1$$

and

$$\frac{1}{(1 + 0)^{p_y}} = 1$$

Decision 1 then moves from S_{new} to S_{old} .

The next time this state is encountered $S_{old} = \{1\}$ and $x = y = 1$. The value-based search then selects decision x with probability equal to one and y with

probability equal to zero since

$$\frac{1}{(1 + (1 - 1))^{p_x}} = 1$$

Then it selects x with probability

$$\frac{1}{(1 + 1)^{p_y}} = \frac{1}{2^{p_y}}$$

and otherwise selects decision $z \in S_{new}$.

4.2 Proof that the Value-Based Search is Correct

To show that the value-based search procedure is correct, we must show that the probability of selecting any decision at every state is bounded away from zero for an unbounded number of iterations of the algorithm. To do this we will first show that the probability of selecting a decision from S_{new} is bounded away from zero. Consequently, in the limit, every decision from S_{new} will be selected with probability one. S_{new} almost surely becomes an empty set. Next we consider the probability of selecting the least explored decision. We will show that the probability of selecting this decision is also bounded away from zero. Then, in the limit, the least explored decision will be explored until it ceases to have this property with probability one. Consequently, the number of times every decision at all states in the graph is chosen tends to infinity as the number of iterations of the algorithm increases.

Proposition 4.1: Let n_s be the number of times the value-based search visits state $s \in S$. Then

$$\lim_{n_s \rightarrow \infty} S_{new} = \emptyset$$

Proof: Let r denote the number of elements in S_{new} . Suppose that there exists an $m > 0$ such that $r \geq m$ for all visits to the state s . This implies that there is an iteration N such that for all $n_s > N$ the probability of selecting a decision from S_{new} is zero for all subsequent iterations. Consequently, for $n_s > N$

$$0 = 1 - \frac{1}{(1 + n_y)^{p_y}}$$

Then $n_y = 0$. However, from the definition $n_y > 0$. Consequently, $m = 0$ and $S_{new} \rightarrow \emptyset$ as n_s tends to infinity. ■

Proposition 4.2: Let n_s be the number of times the value-based search visits state $s \in S$. Then

$$\lim_{n_s \rightarrow \infty} n_y = \infty$$

Proof: By definition

$$n_y = \min\{n_i : i \in S_{old}\}$$

Suppose that there exists a positive integer M such that $n_y \leq M$. This implies that on some iteration, N , the probability of exploring decision x , the best decision found so far, must equal one for all subsequent iterations. In other words, for $n_s > N$

$$1 = \frac{1}{(1 + (n_x - n_y))^{p_x}}$$

which implies $n_x = n_y$. This cannot be true since, with probability one, n_x is incrementing by one with each visit to state s . Consequently, n_y becomes unbounded as $n_s \rightarrow \infty$. ■

When the number of times a state is visited tends to infinity, the value-based search explores each decision at that state infinitely often. Since the initial node is visited on every iteration, all children of the initial state are visited infinitely often. Similarly, all children of these child nodes are visited infinitely often. Inductively, it follows that the value-based search explores each decision at every node infinitely often.

The reason correct search algorithms can guarantee finding optimal decision sequences follows immediately from this inductive argument. In passing to the limit, all correct searches are exhaustive i.e., all correct search algorithms eventually select every path in the graph, including the optimal path.

4.3 Discussion

The decision sequence that results from selecting decisions with largest immediate reward we call the *myopic decision sequence*. It is relatively simple to discover the myopic path and compute the accumulated reward along this path. Any practical search procedure should lead to discovery of decision sequences with at least the value of the myopic path. The value-based search selects this path on the first iteration of the algorithm and then begins to explore away from this decision sequence. The search parameters p_x and p_y determine the “rate” of new exploration. For larger values of p_y , the probability of selecting an unexplored decision increases over the selection of either the best decision found so far or the least explored decision. Similarly, for larger values of p_x the probability of selecting the least explored decision increases over the best decision found so far. Small values of p_x and p_y reflect a conservative search scheme while larger values direct the search to explore away from the best decision, already found, far more often.

While an algorithm iterates, using the value-based search, a change in the estimated value of the initial state reflects a change in the best decision found so far at some state on the best path found. If the estimated value does not change, on some iteration, the value-based search did not find a better path than the current best path. If the algorithm continues to iterate with no improvement to the estimated value of the initial state, then the probability of the best path found being the optimal path increases. However, if the estimated value of the initial state increases, then a better path has been found and paths around it must be again be adequately explored to insure that the new path is probably the best.

Chapter 5

COMPUTATIONAL EXPERIMENTS

To exemplify the search procedures and evaluative information propagation structures, we will consider seven graph examples. These graphs were created using random mechanisms for the selection of an exponentially growing number of nodes at a time step, the number of arcs emanating from a node, and the length of the arcs from the nodes. The graph examples have one initial node, at time $k = 0$, and no arcs leave the terminal nodes, at $k = N$.

Seven algorithms comprise the computational experiments. The Monte Carlo search with the differing information propagation structures constitute the first four algorithms. Using the value-based search with the evaluative information propagation structures gives the last three of these algorithms.

The data recorded, during these tests, include the number of iterations before the algorithm discovers an optimal decision sequence and the number of nodes the algorithm did not visit in the process of finding the optimal decision sequence. Because these algorithms were designed to overcome the need to consider all nodes in the graph, the estimated value of the initial node, after the algorithm visits a fraction of the nodes in the graph, is also noted by these tests. The iteration count where this occurs is also recorded. Since the search mechanism of the algorithms is stochastic, each algorithm was repeatedly run to compare each algorithm's expected performance.

Table 5.1: Graph Examples

Graph Name	Description	Appendix	Number of Nodes	Horizon
Example Graph	Small example graph	A	45	10
T1	Tall graph example	C	1156	75
T2	Tall graph example	E	1123	75
T3	Tall graph example	G	28671	100
W1	Wide graph example	I	1409	15
W2	Wide graph example	K	1315	15
W3	Wide graph example	M	26140	20

5.1 The Graph Examples

The seven graph examples can be classified into three categories, one small graph example, three tall graphs, and three wide graphs. The small graph contains 45 nodes spread over a time horizon of ten. For the tall graphs, the number of nodes per time step grows relatively slowly and the graphs have a large horizon. The number of nodes per time step grows more rapidly in the wide graphs and their horizon is correspondingly smaller so that graphs have essentially the same number of nodes.

The topological properties of these graphs along with the myopic and optimal paths through them are given in the appendices. Table (5.1) lists the names of seven graphs along with the appendix where more information about the graph can be found. Also, the total number of nodes in the graph and its horizon is listed.

5.2 The Algorithms

Table (5.2) contains acronyms for the different algorithms that we will test.

Table 5.2: Algorithm Acronyms

Acronym	Algorithm
MC	Monte Carlo search
MC-EV	Monte Carlo search with estimated values
MC-TEV	Monte Carlo search with target and estimated values
MC-CEV	Monte Carlo search with complete estimated values
VB-EV	Value-based search with estimated values
VB-TEV	Value-based search with target and estimated values
VB-CEV	Value-based search with complete estimated values

5.3 The Types of Computational Experiments

Table (5.3) contains the acronyms that denote the four different analyses of the results from the computational experiments. To generate meaningful statistics, each experiment has 80 data points. The fraction of graph nodes visited for the third and fourth experiments is 25% for the tall graphs and 10% for the wide graphs. (The small example graph was not included in this test. One iteration of any algorithm visits over 20% of the graphs nodes in this graph.)

5.4 Computational Experiments on the Example Graphs

The statistics resulting from the computational experiments are extensive. Accordingly, the complete set of statistics can be found in the appendices. Table (5.4) references the appropriate appendix to each example graph.

5.4.1 The Small Example Graph

Along the optimal path, in the small example graph, there is one decision at node 0, one decision at node 1, three decisions at node 2, two decisions at node 3,

Table 5.3: Statistics Generated from the Computational Experiments

Acronym for the Mean	Acronym for the Standard Deviation	Experiment
MN-IC	SD-IC	The iteration count when the algorithm converged
MN-NNV	SD-NNV	The number of graph nodes not visited by the algorithm when the algorithm converged
MN-EV	SD-EV	The estimated value of the initial node after the algorithm visits a specified fraction of the graph nodes
MN-ITR	SD-ITR	The number of iterations that occur when algorithm visits the specified fraction of the graph nodes

three decisions at node 9, three decisions at node 13, three decisions at node 15, four decisions at node 21, five decisions at node 23, and three decisions at node 31. The probability of finding the optimal decision sequence, using the MC algorithm, is the reciprocal of the product of the number of decisions at each state on the optimal trajectory, $1/9720$. As an example, suppose we wish to be assured of a 99% chance

Table 5.4: Statistical Data for the Graph Examples

Graph Name	Appendix
Example Graph	B
T1	D
T2	F
T3	H
W1	J
W2	L
W3	N

of finding the optimal decision sequence, then using Theorem 3.1, the MC algorithm must be iterated at least

$$\log_{(1-1/9720)} 0.01 = 44760$$

times. Referring to Table (B.1) in the appendix, the MC algorithm requires on average 5960.64 iterations to find the optimal trajectory. Using this number in conjunction with Theorem 3.1 allows us to find the probability of finding the optimal decision sequence in this number of iterations by solving

$$\log_{(1-1/9720)} \epsilon = 5960.64$$

for $1 - \epsilon$. This is approximately .458421. In other words, if the algorithm iterates 5960.64 times, the probability of having found the optimal decision sequence is about 46%.

Adding the evaluative information propagation structures to the MC algorithm greatly enhances the probability of finding an optimal decision sequence,¹ as is evidenced in Table (B.1). Instead of almost 6000 iterations to find the optimal decision sequence, these algorithms converge in less than 30 iterations. The MC-EV algorithm converges on average in 26.45 iterations, the MC-TEV algorithm in 17.18 iterations, and the MC-CEV algorithm in 12.53 iterations. These average iteration counts demonstrate the consequences of Theorems 3.6, 3.14, and 3.19, which state that the probability of convergence increases by using the MC-EV algorithm instead

¹For this analysis, we are assuming that the average number of iterations needed to find the optimal decision sequence reflects the probability of finding an optimal decision sequence in a fixed number of iterations. Strictly speaking, each algorithm should be run for a fixed number of iterations and the number of times each algorithm converges counted to compute a statistical point estimate for the probability of convergence in that number of iterations. However, the average number of iterations needed for the algorithm to converge is a more intuitive concept and the probabilistic statements of convergence in a fixed number of iterations can be seen using this statistic.

of the MC algorithm, and by using the MC-TEV algorithm instead of the MC-EV algorithm, and finally by using the MC-CEV algorithm instead of the MC-TEV algorithm.

The value-based search is capable of finding an optimal decision sequence in fewer iterations than the Monte Carlo decision search (compare Table (B.2) with Table (B.1).) Poor selection of the parameters p_x and p_y can degrade the performance of the algorithm. Generally, however, the value-based search outperforms the Monte-Carlo search by a factor of between two and three.

Both the Monte-Carlo search and the value-based search algorithms must visit a large fraction of the states before finding an optimal decision sequence. Comparing the average number of states not visited in Table (B.1) and Table (B.4) shows that by adding the differing evaluative information propagation structures, the algorithm needs to visit fewer states before finding the optimal decision sequence. The value-based search also visits less states than the Monte Carlo search before discovering the optimal decision sequence.

5.4.2 The Tall Graph (T1) Example

The effects of Theorems 3.6, 3.14, and 3.19 are again seen in the average number of iterations each algorithm requires before finding the optimal decision sequence, as shown in Table (D.1). Similar to the small graph example, each algorithm visits a large fraction of states before discovering an optimal decision sequence. The differing algorithms distinguish themselves more for this graph than the small example graph. Convergence occurs on average in 1151.91 iterations versus 190.29 iterations when comparing the MC-EV and MC-CEV algorithms, respectively. Also for the MC-EV and MC-CEV algorithms, on average 8.00 states are not visited versus 71.31 states

not visited, respectively. About a sixfold increase in performance results from using the MC-CEV algorithm instead of the MC-EV algorithm for this graph.

The value-based search outperforms the Monte Carlo search by a factor of around two, when using better parameters. Again it is possible for it to perform more poorly than the Monte Carlo search. However, for most parameter choices the value-based search does better than the Monte Carlo search and for the best parameter choices it converges twice as quickly as the Monte Carlo search.

In contrast to the small graph example, the addition of target value information to estimated value information did not reduce the need to visit states when using the value-based search (see Table (D.4).) The use of complete estimated values substantially increases the number of states not visited, by around a factor of ten. In any case, however, the algorithms must visit 90% or more of the states before discovery of the optimal decision sequence.

Although all these algorithms converge, the computational effort and memory requirements for these algorithms can be extensive. Instead of finding the optimal decision sequence, a more modest goal is to find a decision sequence that results in a larger reward than the reward associated with the easily found myopic decision sequence. For this graph, if myopic decisions are made a reward of 159.67 results. On the iteration when the MC-CEV algorithm first exceed visiting 25% of the states in this graph the average reward is 172.65. The optimal decision sequence results in a reward of 296.63. Using the value-based search process achieves rewards greater than 240 after visiting 25% of the states in the graph (see Table (D.6).) These rewards can be realized in as few as six iterations of the algorithm. In this case the value-based search provides a better starting decision sequence, the myopic decision sequence, than a random decision sequence resulting from the Monte Carlo search procedure.

5.4.3 The Tall Graph (T2) Example

The convergence analysis for graph T2 shows that graph T1 is not anomalous. The consequences of Theorems 3.6, 3.14, and 3.19 are again seen in Tables (F.1) and (F.2). Using the value-based search results in quicker convergence, for reasonable parameter choices. The algorithms still visit a large fraction of the states before finding the optimal decision sequence. In contrast to graph T1, the value of the best path found in graph T2, after examining 25% of the states, shows a smaller improvement over the value of the myopic decision sequence.

5.4.4 The Tall Graph (T3) Example

Graph T3 has nearly 30 times as many nodes as those in graphs T1 and T2. Because of the size of this graph, only the algorithms that use complete estimated values and the tests that visited a fraction on the nodes were performed. The myopic decision sequence has value 284.07 and the optimal decision sequence, value 497.87. The MC-CEV algorithm finds a decision sequence with value 434.03 after examining 25% of the states. The value-based search performs better and finds paths with values greater than 480 (see Tables (H.1) and (H.2).) Both find these paths in less than 200 iterations.

5.4.5 The Wide Graph (W1) Example

The effects of Theorems 3.6, 3.14, and 3.19 are again evident in the average number of iterations each algorithm requires before finding the optimal decision sequence, as noted in Table (J.1). Although these algorithms still visit a large fraction of the states, upon discovery of the optimal decision sequence a larger fraction of the states have not been visited for this graph topology.

The value-based search outperforms the Monte Carlo search by a factor of five or six, when using better parameters (see Table(J.2).) This is a substantial increase over the graphs with the tall topologies where the increase was about a factor of two.

The addition of target value information to estimated value information reduces the need to visit states, when using the value-based search (see Table (J.4).) The use of complete estimated values, in contrast to the tall graphs, adds little to the number of states not visited. The VB-TEV and VB-CEV algorithms visit about 1/3 of the states before discovery of the optimal decision sequence, an improvement over the tall graph examples which visit 90% of the states before converging.

For this graph, myopic decision making results in a reward of 47.76. When the MC-CEV algorithm visits more than 10% of the states in this graph, the average reward is 50.66. The optimal decision sequence results in a reward of 72.25. The value-based search process achieves rewards greater than 62 (see Table (J.6).) These rewards are realized in as few as 15 iterations of the algorithm.

5.4.6 The Wide Graph (W2) Example

The computational experiments for graph W2 are similar to W1. The convergence analysis for graph W2 shows that graph W1 is not anomalous. The consequences of Theorems 3.6, 3.14, and 3.19 are again seen in Tables (L.1) and (L.2). Using the value-based search results in quicker convergence, for any reasonable parameter choice. In contrast to graph W1, the VB-CEV algorithm visits fewer states than the VB-TEV algorithm (see Table (L.4).)

5.4.7 The Wide Graph (W3) Example

Graph W3 has about 20 times as many nodes as those in graph examples W1 and W2. The myopic decision sequence has value 59.35 and the optimal decision sequence, value 94.30. The MC-CEV algorithm finds a decision sequence with value 75.56 after examining 10% of the states. The value-based search performs better and finds paths with values greater than 87 (see Tables (N.1) and (N.2).) These decision sequences are found in about 300 iterations of the algorithm.

5.5 The Value-Based Search Parameters, p_x and p_y

As noted in the previous discussion, the value-based search is generally able to find the optimal decision sequence in fewer iterations than the Monte-Carlo search. Examination of the data in the appendices also indicates that certain values of the search parameters, p_x and p_y , can again increase the overall performance of the value-based search.

Better performance of the value-based search tends to occur for larger values of p_y . Consequently, selection of an unexplored decision at a state occurs more frequently. With some exceptions (see graph W2) increases in performance occur for larger values of p_x , also. Larger values of p_x tend to keep the decisions in the set S_{old} explored equally often. (If $n_x = n_y$, then decision x is explored with probability one. However when $n_x > n_y$, decision y is explored more frequently with increasing values of p_x .)

As might be expected, when using complete estimated values, the performance of the value-based search is relatively insensitive to the parameter p_x and more sensitive to p_y . This is because an algorithm that uses complete estimated values finds

a decision sequence with larger value only by adding new states to the current sub-problem.

The minimum number of iterations needed to find an optimal decision sequence generally occurs for parameters on the edge of the parameter space presented in the appendices. An examination of this data shows a rapid improvement in performance when going from the smallest parameter values to somewhat larger values. The increase in performance then begins to slow as the parameters are increased further. Examination of data using larger values of the parameters, values larger than those shown in the appendices, suggests this trend continues for further increases of the parameter values.

Perhaps the search that would result by letting the parameter values go to infinity would minimize the mean number of iterations. This new search would necessarily add states to the list of states visited most quickly, possibly exhausting computer memory. However this has not been verified by computational experiments.

5.6 Discussion

These examples demonstrate that it is possible to discover an optimal decision sequence using a generate-and-test type algorithm. Only a fraction of the nodes in the graph need be visited before discovering an optimal decision sequence. For extremely large problems this may still be impractical or impossible. Consequently for these large problems, the modest goal of finding a decision sequence better than the myopic decision sequence is more realistic.

The statistical analysis of the computational experiments suggest that it is possible to significantly increase the rate of convergence by adding appropriate evaluative information propagation schemes. Adding the value-based search heuristic again in-

creases the rate of convergence. Practical success, in using algorithms of this type, probably occurs most frequently by adding problem specific search heuristics that direct the search procedure into areas of the state space where optimality is thought to occur.

Chapter 6

CONCLUSIONS

6.1 Using Information

The four methods for remembering value information from one iteration to another, no value information (except path length), estimated values, target and estimated values, and complete estimated values have a demonstrable effect on the rate of convergence of these algorithms. Our analysis reveals that this occurs because each of these evaluative information structures requires the search process, used by these algorithms, to visit states on the optimal trajectory in certain orders. Using only path length information in searching for an optimal decision sequence constrains the search process to find the complete optimal trajectory on a single iteration. Transitioning through the use of estimated values, target and estimated values, to complete estimated values allows us to completely relax this constraint to the point where the order of visiting the states on the optimal trajectory is immaterial.

For algorithms that attempt to optimize the performance of a dynamic system using a generate-and-test type strategy we conclude: Appropriate use of the information collected from simulations of a dynamic system can significantly increase the probability of finding an optimal decision sequence.

6.2 Directions for Future Research

6.2.1 Forgetting Information

The list of states these algorithms use to compute value information steadily increases in size. However, each algorithm need access only those states on an optimal trajectory to find an optimal decision sequence (obviously.) Other states have no import on the computation of estimated, target, or complete estimated values for the states on the optimal trajectory. With this in mind, at some point it may be possible to remove states from the list of states and thereby forget their values. For large problems, that may exhaust computer memory, this mechanism may provide a means to hold only the “important states” in computer memory.

6.2.2 Stochastic Estimation for the Markov Decision Problem

Using the notation from Chapter 2, the value function for the Markov Decision Problem satisfies

$$\begin{aligned}
 J(i) &= g_i(i) \text{ for } i \in \partial S \\
 J(i) &= \max_u \{g(i, u) + \sum_{j \in S} p_{ij}(u) J(j)\} \text{ for } i \in S \setminus \partial S
 \end{aligned} \tag{6.1}$$

whereas the value function for the longest path in an acyclic graph satisfies

$$\begin{aligned}
 J(i) &= r_i \text{ for } i \in \partial S \\
 J(i) &= \max_{j \in S} \{r_{ij} + J(j)\} \text{ for } i \in S \setminus \partial S
 \end{aligned} \tag{6.2}$$

For a generate-and-test type algorithm it was possible to use (6.2) to define estimated values. The estimated value computation always determines a current value for $\bar{J}(i)$ having current values of $\bar{J}(j)$. Therefore estimated values are always well defined quantities.

Computing an estimated value in a similar way, for the Markov Decision Problem, introduces the difficulty that, for a given simulation only one realization of the stochastic disturbance occurs, however, the value computation requires a summation (6.1) over the values of all states resulting from every possible stochastic disturbance. For these other states, current estimated values may not exist.

In the deterministic case, convergence of the estimated value for a state to its value occurs because the sequence of estimated values is monotonically nondecreasing and has a least upper bound equal to the value of the state. For the Markov Decision Problem, if an arbitrary value is assigned to the states not yet visited during a simulation, to compute the summation in (6.1), the sequence of estimated values loses the property of being monotonic. Consequently, devising a convergent process becomes more difficult.

For these problems, an update rule of the following form has been suggested [2, 5] for all nonterminal states (at terminal states $J_{n+1}(\cdot)$ is the terminal reward),

$$J_{n+1}(i) = J_n(i) + a[\max_u \{g(i, u) + \sum_{j \in S} p_{ij}(u) J_n(j)\} - J_n(i)] \quad (6.3)$$

where $0 < a < 1$ and n is the number of times state i has been visited. For this update scheme to work, all $J_0(\cdot)$ must be assigned finite values e.g., $J_0(i) \equiv 0, \forall i \in S$.

Using (6.3) to define a sequence of estimated values for the states in a Markov Decision Problem and then examining its convergence properties, both theoretically

and computationally might provide a means to develop a theory similar to the one developed here, for the Markov Decision Problem.

6.2.3 Increasing the Probability of Finding Optimal States

The concept of complete estimated values allows us to focus the search mechanism on the process of discovering states on an optimal trajectory. The search process need not concern itself with having to visit optimal states in certain orders. Search processes that direct themselves into areas of the state space where optimality is thought to occur, now become more capable of increasing the probability of discovering an optimal decision sequence.

The Monte Carlo search is essentially oblivious to the nature of the problem at hand. It merely selects decisions randomly. The value-based search directs decision selection using the myopic path as a starting point. If optimality is “far from” the myopic path, convergence may be quite slow.

For problems with an underlying physical nature, it may be possible to appeal to this physical nature to devise problem specific search heuristics. By using the physical nature of the problem, it may be possible to design problem specific search algorithms that, in some sense, adequately sample different areas of the state space while searching for optimal states.

Bibliography

- [1] Bellman, R., *Dynamic Programming*. Princeton University Press, Princeton, N.J., 1957.
- [2] Barto, A. G., Sutton, R. S., and Anderson, C. W. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics*, 13:834-846, 1983.
- [3] Barto, A. G., Sutton, R. S., and Brouwer, P. S. Associative Search Network: A Reinforcement Learning Associative Memory. *Biological Cybernetics*, 40:201-211, 1981.
- [4] Sutton, R. S. Learning to Predict by the Methods of Temporal Differences. *Machine Learning*, 3:9-44, 1988.
- [5] Watkins, C. W. C. Q-Learning. *Machine Learning*, 5:1-99, 1989.
- [6] Narendra, K. S., and Thathachar, M. A. L. *Learning Automata: An Introduction*. Prentice Hall, Englewood Cliffs, New Jersey, 1989.
- [7] Fleming W. H., and Rishel, R. W. *Deterministic and Stochastic Optimal Control*. Springer-Verlag, New York, 1975.
- [8] Bertsekas, D. P. *Dynamic Programming: Deterministic and Stochastic Models*. Prentice Hall, Englewood Cliffs, New Jersey, 1987.

- [9] Feller W. *An Introduction to Probability Theory and Its Applications. Vol 1.* John Wiley and Sons, New York, 1968.
- [10] Tolwinski, B. and R. Underwood. An algorithm to estimate the optimal evolution of an open pit mine, In *Proceedings of 1992 APCOM Conference*, pages 399-409, Tucson, Arizona, April 1992.

Appendix A

SMALL GRAPH EXAMPLE

A.1 The Small Graph's Topology

- Time horizon: 10
- Number of nodes: 45
- Total number of arcs: 121
- Number of different paths: 8883
 - Table (A.1) shows the distribution of nodes in the graph.
- The number of arcs from each node is a random integer from the set $\{2, 3, 4, 5\}$.
(Terminal nodes have no arcs.)
- Arc lengths are drawn randomly from a uniform probability distribution on $[0,1]$. With probability $1/10$, the length of the arc is multiplied by 10. This provides a mechanism to place isolated large rewards in the graph.

A.2 Paths in the Example Graph

- The myopic path through the graph is in Table (A.2).
 - Its value is 6.55.
- Table (A.3) contains the optimal path through this graph.
 - Its value is 29.14.

Table A.1: Topological Structure of the Small Example Graph

Time Index	Node Indexes	Number of Nodes
0	0-0	1
1	1-1	1
2	2-2	1
3	3-5	3
4	6-9	4
5	10-14	5
6	15-17	3
7	18-21	4
8	22-26	5
9	27-37	11
10	38-44	7

Table A.2: The Myopic Path through the Small Example Graph

Time	Node	Decision	Value
0	0	1	6.648365
1	1	1	6.566820
2	2	1	5.567156
3	3	1	3.982025
4	9	1	3.426801
5	10	1	2.906217
6	16	1	2.132511
7	21	1	1.764519
8	22	1	1.056520
9	27	1	0.255776
10	44	-	0.000000

Table A.3: The Optimal Path through the Small Example Graph

Time	Node	Decision	Value
0	0	1	29.138735
1	1	1	29.057190
2	2	1	28.057526
3	3	1	26.472395
4	9	3	25.917171
5	13	1	25.846766
6	15	1	19.605120
7	21	2	11.022095
8	23	1	10.423322
9	31	1	0.982727
10	40	-	0.000000

Appendix B
SMALL GRAPH COMPUTATIONAL TESTS

- 6.55 is the length of the myopic path through the graph.
- 29.14 is the length of the longest path through the graph.

Table B.1: Monte Carlo Search Tests

Algorithm	MN-IC	SD-IC	MN-NNV	SD-NNV
MC	5960.64	10025.66		
MC-EV	26.45	13.33	4.65	3.58
MC-TEV	17.18	9.73	6.54	3.56
MC-CEV	12.53	6.81	8.28	4.88

Table B.2: Value-Based Search Tests (MN-IC)

Algorithm: VB-EV, Test: MN-IC

		px									
		0.1	0.3	0.5	0.7	0.9	1.1	1.3	1.5	1.7	1.9
py	0.1	37.51	29.99	27.71	27.73	31.73					
	0.3	21.38	18.69	18.61	18.55	17.48	19.43	17.39			
	0.5	20.40	15.07	13.04	14.93	14.82	14.80	14.55	13.85		
	0.7	19.21	13.70	13.21	14.06	13.35	13.71	12.47	12.90		
	0.9	18.12	12.28	11.44	13.11	12.04	12.20	10.54	12.00	12.18	12.07
	1.1		11.99	11.21	11.82	10.64	10.93	11.39	10.66	11.06	10.34
	1.3		11.68	10.75	11.70	10.30	10.78	10.41	9.85	10.54	
	1.5			10.66	10.38	10.36	9.62	10.46	9.81		
	1.7					9.68	9.75	9.51			
	1.9					10.00	9.35				

Algorithm: VB-TEV, Test: MN-IC

		px									
		0.1	0.3	0.5	0.7	0.9	1.1	1.3	1.5	1.7	1.9
py	0.1	20.31	20.05	20.34	22.60	22.35					
	0.3	11.88	10.72	11.05	11.81	11.82	12.94	11.80			
	0.5	11.06	9.65	9.34	9.32	9.94	10.46	9.60	9.24		
	0.7	8.57	7.69	8.68	8.64	8.71	8.78	7.94	8.00		
	0.9	9.19	7.79	7.21	7.22	7.15	7.40	8.04	7.30	8.00	6.76
	1.1		6.24	6.41	6.17	7.06	6.64	7.56	6.94	7.50	7.12
	1.3		6.95	6.90	6.50	6.29	6.11	6.53	6.75	7.26	
	1.5			6.16	6.17	6.10	5.99	5.96	6.22		
	1.7					5.88	5.96	6.30			
	1.9					5.50	5.84				

Algorithm: VB-CEV, Test: MN-IC

		px									
		0.1	0.3	0.5	0.7	0.9	1.1	1.3	1.5	1.7	1.9
py	0.1	17.55	16.93	21.64	22.85	22.76					
	0.3	8.53	9.80	10.90	11.84	11.68	11.05	11.38			
	0.5	7.38	7.31	7.88	8.34	7.96	8.21	9.56	8.54		
	0.7	5.85	5.69	6.88	7.16	7.69	7.66	7.51	7.86		
	0.9	5.46	6.11	6.11	6.34	6.11	6.49	6.35	7.05	6.91	6.97
	1.1		5.24	5.09	5.46	5.71	5.89	6.04	6.34	6.65	6.17
	1.3		5.25	5.31	5.67	5.26	5.75	6.08	6.19	6.01	
	1.5			4.78	5.42	5.51	4.88	5.39	4.94		
	1.7					5.12	5.12	5.29			
	1.9					4.89	4.41				

Table B.3: Value-Based Search Tests (SD-IC)

Algorithm: VB-EV, Test: SD-IC

		px									
		0.1	0.3	0.5	0.7	0.9	1.1	1.3	1.5	1.7	1.9
py	0.1	36.47	14.41	14.73	14.28	17.25					
	0.3	10.39	11.46	8.29	9.34	8.62	9.41	8.41			
	0.5	9.06	6.08	4.80	7.13	6.70	6.19	7.09	7.23		
	0.7	8.98	5.60	5.98	7.22	7.61	6.79	5.28	5.76		
	0.9	8.16	6.88	4.61	7.33	4.20	6.35	5.35	5.57	5.24	5.87
	1.1		4.09	4.69	4.87	4.91	5.27	5.87	4.40	5.20	4.03
	1.3		3.82	3.69	5.81	4.44	5.28	4.03	4.79	5.35	
	1.5			5.13	4.20	4.86	3.62	5.54	4.72		
	1.7					3.66	3.65	4.21			
	1.9					4.17	3.07				

Algorithm: VB-TEV, Test: SD-IC

		px									
		0.1	0.3	0.5	0.7	0.9	1.1	1.3	1.5	1.7	1.9
py	0.1	12.84	12.70	14.08	13.65	12.53					
	0.3	8.01	6.31	6.42	6.68	6.09	6.37	7.27			
	0.5	7.63	4.63	4.83	5.25	5.72	5.37	4.89	5.21		
	0.7	5.72	4.57	4.44	4.11	5.08	4.32	4.12	4.10		
	0.9	7.18	4.61	2.90	3.14	3.62	3.84	4.28	3.84	4.21	3.28
	1.1		3.01	2.52	3.14	3.76	3.47	3.70	3.20	3.88	3.52
	1.3		3.26	3.30	2.91	2.93	2.81	2.89	2.99	3.33	
	1.5			2.12	2.29	2.62	2.37	2.34	2.68		
	1.7					2.71	2.49	2.90			
	1.9					1.96	2.44				

Algorithm: VB-CEV, Test: SD-IC

		px									
		0.1	0.3	0.5	0.7	0.9	1.1	1.3	1.5	1.7	1.9
py	0.1	10.75	8.04	14.91	13.73	12.40					
	0.3	3.60	5.11	5.68	8.11	7.32	6.55	6.32			
	0.5	3.76	3.28	3.69	4.67	4.57	4.26	5.33	4.60		
	0.7	1.98	2.41	3.37	3.64	3.40	3.80	3.65	3.90		
	0.9	2.35	2.92	3.03	3.02	2.59	3.00	3.06	3.13	3.44	3.30
	1.1		1.72	1.79	2.35	2.54	2.46	2.70	3.23	3.49	2.71
	1.3		2.12	1.93	2.76	2.30	2.66	2.94	2.92	2.83	
	1.5			2.03	2.31	2.70	1.96	2.58	1.97		
	1.7					2.23	2.69	2.33			
	1.9					1.81	1.56				

Table B.4: Value-Based Search Tests (MN-NNV)

Algorithm: VB-EV, Test: MN-NNV

		px									
		0.1	0.3	0.5	0.7	0.9	1.1	1.3	1.5	1.7	1.9
py	0.1	10.99	10.89	11.55	11.25	10.86					
	0.3	9.91	10.31	9.30	9.21	10.10	8.32	10.01			
	0.5	7.72	8.61	9.90	8.71	8.47	8.68	9.47	10.18		
	0.7	6.46	8.07	8.18	7.91	8.94	7.97	8.41	8.41		
	0.9	6.10	8.34	8.64	7.79	7.41	8.47	10.07	8.18	7.90	8.44
	1.1		7.15	7.91	7.62	8.46	8.59	8.29	8.36	8.57	8.46
	1.3		7.09	7.44	7.41	7.95	7.99	7.85	9.04	8.12	
	1.5			7.53	7.94	7.41	8.19	7.99	8.43		
	1.7					7.45	7.61	8.09			
	1.9					6.90	7.30				

Algorithm: VB-TEV, Test: MN-NNV

		px									
		0.1	0.3	0.5	0.7	0.9	1.1	1.3	1.5	1.7	1.9
py	0.1	15.64	15.36	15.75	14.10	13.53					
	0.3	15.55	16.00	15.20	14.09	14.46	12.96	14.44			
	0.5	13.90	13.95	13.94	14.66	14.01	12.60	13.61	14.24		
	0.7	14.93	15.07	13.46	13.09	13.74	13.43	14.50	14.90		
	0.9	13.79	14.56	14.05	14.49	14.89	14.59	13.74	14.71	13.49	15.11
	1.1		15.64	14.85	15.74	14.16	15.00	13.03	14.05	13.70	13.89
	1.3		13.56	13.40	14.28	14.60	15.04	14.21	13.85	12.57	
	1.5			13.89	14.03	14.51	14.46	14.55	14.05		
	1.7					14.88	14.30	13.66			
	1.9					14.80	14.24				

Algorithm: VB-CEV, Test: MN-NNV

		px									
		0.1	0.3	0.5	0.7	0.9	1.1	1.3	1.5	1.7	1.9
py	0.1	17.44	16.59	15.43	14.15	13.91					
	0.3	18.00	16.81	15.29	14.96	15.20	14.97	14.81			
	0.5	17.74	17.27	16.31	15.74	16.26	15.62	13.99	15.29		
	0.7	18.07	18.38	16.11	15.95	14.30	14.96	15.16	15.09		
	0.9	18.35	16.57	17.05	16.31	16.40	15.49	15.95	14.56	15.29	14.94
	1.1		17.66	17.79	17.09	16.70	16.25	16.02	15.84	15.03	15.85
	1.3		17.38	16.68	16.55	17.04	16.12	15.56	15.01	15.45	
	1.5			17.86	16.23	16.38	17.55	16.66	17.24		
	1.7					16.77	17.09	16.36			
	1.9					16.98	18.30				

Table B.5: Value-Based Search Tests (SD-NNV)

Algorithm: VB-EV, Test: SD-NNV

		px									
		0.1	0.3	0.5	0.7	0.9	1.1	1.3	1.5	1.7	1.9
py	0.1	5.56	5.29	5.99	5.85	6.78					
	0.3	5.77	5.57	5.35	5.14	6.07	4.91	5.81			
	0.5	4.66	4.06	4.81	5.26	4.86	4.49	6.01	6.60		
	0.7	3.40	3.54	4.52	4.04	5.73	4.71	4.29	4.51		
	0.9	3.40	3.93	4.83	4.09	3.17	4.49	5.04	4.21	4.20	4.51
	1.1		3.60	4.04	4.38	4.63	4.72	4.44	4.50	4.89	4.02
	1.3		3.61	3.42	3.66	3.49	3.79	3.36	4.24	4.22	
	1.5			3.23	4.63	3.41	3.45	4.28	3.75		
	1.7					3.24	3.16	3.72			
	1.9					3.08	3.04				

Algorithm: VB-TEV, Test: SD-NNV

		px									
		0.1	0.3	0.5	0.7	0.9	1.1	1.3	1.5	1.7	1.9
py	0.1	5.21	6.58	6.76	6.62	6.55					
	0.3	7.05	6.13	6.69	6.46	6.72	6.62	7.30			
	0.5	6.47	6.10	6.42	6.62	7.21	6.76	6.07	6.72		
	0.7	6.34	6.02	6.26	6.26	6.46	6.53	6.68	6.69		
	0.9	6.45	6.67	5.86	6.15	7.21	6.60	6.95	6.99	6.97	6.35
	1.1		6.33	5.98	6.53	6.53	6.52	6.38	6.16	6.58	6.37
	1.3		6.66	5.78	6.10	5.96	6.53	6.49	5.62	6.19	
	1.5			5.44	5.45	5.56	5.78	5.86	5.67		
	1.7					5.70	6.00	5.65			
	1.9					5.53	5.56				

Algorithm: VB-CEV, Test: SD-NNV

		px									
		0.1	0.3	0.5	0.7	0.9	1.1	1.3	1.5	1.7	1.9
py	0.1	5.30	4.91	7.27	6.92	6.84					
	0.3	4.55	5.45	5.52	6.75	6.99	6.42	6.58			
	0.5	4.91	5.54	5.69	6.66	6.48	6.12	6.80	6.36		
	0.7	4.43	4.39	5.58	6.00	6.18	6.59	6.29	6.74		
	0.9	4.70	5.55	5.87	6.26	5.63	5.72	6.17	5.98	6.46	6.61
	1.1		3.96	4.89	5.44	5.83	5.80	5.80	6.72	6.92	5.98
	1.3		4.73	4.76	5.45	5.67	5.92	6.93	6.45	6.31	
	1.5			5.01	5.68	5.34	4.94	5.83	5.77		
	1.7					5.72	5.98	5.56			
	1.9					4.89	4.59				

Appendix C

TALL GRAPH (T1) EXAMPLE

C.1 Graph T1's Topology

- Time horizon: 75
- Number of nodes: 1156
- Total number of arcs: 6353
- Number of different paths: $\approx 1.87 \times 10^{47}$
 - Table (C.1) shows the distribution of nodes in the graph.
- The number of arcs from each node is a random integer from the set $\{2, \dots, 10\}$. (Terminal nodes have no arcs.)
- Arc lengths are drawn randomly from a uniform probability distribution on $[0,1]$. With probability $1/10$, the length of the arc is multiplied by 10. This provides a mechanism to place isolated large rewards in the graph.

C.2 Paths in Graph T1

- The myopic path through the graph is in Table (C.2).
 - Its value is 159.67.
- Table (C.3) contains the optimal path through this graph.
 - Its value is 296.63.

Table C.1: Graph T1's Topological Structure

Time Index	Node Indexes	Number of Nodes	Time Index	Node Indexes	Number of Nodes
0	0-0	1	38	160-165	6
1	1-1	1	39	166-178	13
2	2-3	2	40	179-188	10
3	4-4	1	41	189-198	10
4	5-6	2	42	199-208	10
5	7-7	1	43	209-221	13
6	8-8	1	44	222-229	8
7	9-11	3	45	230-239	10
8	12-12	1	46	240-255	16
9	13-14	2	47	256-266	11
10	15-17	3	48	267-282	16
11	18-20	3	49	283-292	10
12	21-22	2	50	293-307	15
13	23-26	4	51	308-326	19
14	27-30	4	52	327-340	14
15	31-32	2	53	341-357	17
16	33-35	3	54	358-380	23
17	36-38	3	55	381-402	22
18	39-41	3	56	403-430	28
19	42-44	3	57	431-445	15
20	45-49	5	58	446-475	30
21	50-54	5	59	476-505	30
22	55-60	6	60	506-524	19
23	61-64	4	61	525-545	21
24	65-67	3	62	546-568	23
25	68-71	4	63	569-604	36
26	72-77	6	64	605-626	22
27	78-81	4	65	627-671	45
28	82-87	6	66	672-700	29
29	88-91	4	67	701-741	41
30	92-96	5	68	742-789	48
31	97-104	8	69	790-822	33
32	105-113	9	70	823-876	54
33	114-121	8	71	877-920	44
34	122-127	6	72	921-960	40
35	128-138	11	73	961-1025	65
36	139-149	11	74	1026-1092	67
37	150-159	10	75	1093-1155	63

Table C.2: The Myopic Path through Graph T1

Time	Node	Decision	Value	Time	Node	Decision	Value
0	0	1	159.6742	38	161	1	98.5877
1	1	1	159.6572	39	178	1	94.5798
2	2	1	159.2108	40	186	1	92.6595
3	4	1	158.5154	41	193	1	91.6649
4	5	1	153.0627	42	199	1	90.9458
5	7	1	152.5022	43	219	1	83.8853
6	8	1	152.2178	44	228	1	82.9486
7	10	1	151.3093	45	237	1	75.0303
8	12	1	150.3579	46	248	1	68.8887
9	13	1	149.3636	47	260	1	67.7253
10	17	1	148.1530	48	281	1	66.7576
11	18	1	147.2529	49	288	1	65.7927
12	21	1	146.3415	50	307	1	64.9269
13	24	1	145.3668	51	311	1	57.7627
14	28	1	143.4981	52	337	1	56.9487
15	31	1	142.9962	53	353	1	56.7990
16	35	1	140.6799	54	360	1	55.8364
17	38	1	136.3783	55	398	1	52.2718
18	39	1	135.5521	56	417	1	46.1819
19	42	1	134.6691	57	437	1	45.2623
20	47	1	133.9538	58	466	1	44.4376
21	52	1	133.0029	59	484	1	34.8359
22	55	1	132.1069	60	512	1	33.8483
23	63	1	131.2100	61	532	1	32.8761
24	66	1	130.5244	62	549	1	30.7331
25	68	1	129.6110	63	596	1	29.7414
26	77	1	128.6979	64	606	1	28.5003
27	79	1	124.1003	65	639	1	27.5535
28	87	1	123.2044	66	689	1	24.7748
29	89	1	122.2149	67	739	1	23.9565
30	93	1	121.2563	68	744	1	23.0008
31	103	1	114.6222	69	802	1	22.0335
32	107	1	113.7913	70	837	1	20.2842
33	115	1	103.8841	71	911	1	14.7945
34	124	1	102.9518	72	946	1	13.8469
35	128	1	102.0174	73	974	1	4.0587
36	142	1	101.1669	74	1044	1	2.4339
37	153	1	100.5220	75	1153	-	0.0000

Table C.3: The Optimal Path through Graph T1

Time	Node	Decision	Value	Time	Node	Decision	Value
0	0	1	296.6302	38	163	1	201.6697
1	1	1	296.6132	39	167	1	195.0249
2	2	1	296.1668	40	188	1	189.5395
3	4	1	295.4714	41	191	9	183.3327
4	5	1	290.0187	42	202	1	183.3157
5	7	1	289.4582	43	221	2	175.6455
6	8	1	289.1738	44	228	1	169.6358
7	10	1	288.2653	45	237	1	161.7174
8	12	1	287.3139	46	248	4	155.5759
9	13	1	286.3196	47	257	1	155.0867
10	17	3	285.1090	48	277	2	145.6470
11	19	1	284.5154	49	292	1	145.1394
12	22	4	275.8945	50	295	8	136.2231
13	24	1	275.7371	51	325	1	136.0966
14	28	2	273.8684	52	330	2	128.5610
15	32	1	273.4601	53	354	1	127.6191
16	33	1	267.2108	54	375	1	117.8382
17	38	2	266.6895	55	384	1	108.4632
18	41	1	266.2030	56	405	2	107.5872
19	43	1	264.3640	57	438	1	103.4162
20	49	1	263.5692	58	453	1	94.1899
21	50	1	262.8498	59	497	1	93.2914
22	59	1	255.4048	60	511	4	84.9562
23	61	1	251.9083	61	534	1	84.6743
24	65	1	251.0746	62	547	1	76.8176
25	68	1	250.2681	63	594	1	69.6727
26	77	1	249.3550	64	618	1	62.8386
27	79	5	244.7574	65	632	4	57.5367
28	84	1	244.3882	66	679	1	56.7482
29	88	1	237.8068	67	736	1	48.7090
30	93	1	236.8694	68	785	1	40.1544
31	103	1	230.2353	69	810	6	34.0647
32	107	1	229.4044	70	850	3	33.6096
33	115	3	219.4971	71	889	1	32.7418
34	123	3	218.6942	72	947	2	23.5887
35	137	1	218.0644	73	976	1	15.8635
36	141	1	210.6160	74	1041	1	7.6925
37	154	2	202.5588	75	1155	-	0.0000

Appendix D

TALL GRAPH (T1) COMPUTATIONAL EXPERIMENTS

- 159.67 is the length of the myopic path through the graph.
- 296.63 is the length of the longest path through the graph.

Table D.1: Monte Carlo Search Tests

Algorithm	MN-IC	SD-IC	MN-NNV	SD-NNV
MC-EV	1155.91	239.62	8.00	1.15
MC-TEV	607.95	149.59	13.21	4.23
MC-CEV	190.29	89.78	71.31	41.52
Algorithm	MN-EV	SD-EV	MN-ITR	SD-ITR
MC-CEV	172.65	11.35	6.00	0.00

Table D.2: Value-Based Search Tests (MN-IC)

Algorithm: VB-EV, Test: MN-IC

		px									
		0.1	0.3	0.5	0.7	0.9	1.1	1.3	1.5	1.7	1.9
py	0.1	9147.01	1734.86	1211.50	1010.46	952.73					
	0.3	7054.75	1420.00	1030.24	869.48	799.50	751.99	726.76			
	0.5	4163.93	1372.31	976.09	865.59	762.45	712.83	666.84	641.10		
	0.7	8170.76	1316.33	938.08	835.14	742.50	690.27	646.64	622.64		
	0.9	5718.11	1304.22	979.71	811.02	728.39	683.80	630.88	588.35	578.60	556.76
	1.1		1333.83	921.96	799.09	717.45	674.94	597.76	603.88	566.52	556.24
	1.3		1389.01	957.95	795.11	712.21	665.73	637.77	590.74	565.29	
	1.5			917.58	781.94	709.15	644.10	629.96	572.34		
	1.7					709.99	641.40	613.64			
	1.9					695.67	643.00				

Algorithm: VB-TEV, Test: MN-IC

		px									
		0.1	0.3	0.5	0.7	0.9	1.1	1.3	1.5	1.7	1.9
py	0.1	1943.84	762.14	594.55	570.56	511.96					
	0.3	1556.28	618.33	498.48	438.90	409.39	380.05	380.99			
	0.5	1448.55	634.36	470.41	424.94	392.12	376.59	361.55	346.39		
	0.7	1624.64	573.76	439.74	420.59	381.57	366.88	343.59	330.36		
	0.9	1551.99	558.10	463.02	399.56	382.94	357.24	335.23	323.05	310.79	302.98
	1.1		639.77	463.99	409.02	368.61	346.26	335.79	320.62	307.04	305.07
	1.3		578.06	460.80	412.24	368.06	351.55	337.74	319.43	312.62	
	1.5			465.06	386.81	373.88	349.20	323.75	307.95		
	1.7					370.14	345.40	339.56			
	1.9					358.84	338.05				

Algorithm: VB-CEV, Test: MN-IC

		px									
		0.1	0.3	0.5	0.7	0.9	1.1	1.3	1.5	1.7	1.9
py	0.1	443.94	285.19	258.73	245.09	243.75					
	0.3	236.06	169.70	144.54	136.44	133.43	123.39	137.32			
	0.5	185.97	131.15	120.78	107.28	109.11	105.66	107.75	107.35		
	0.7	160.90	117.97	104.81	103.64	96.79	93.97	87.76	94.22		
	0.9	149.72	109.33	101.08	95.26	89.31	90.58	81.67	86.15	84.66	88.00
	1.1		105.70	97.61	91.31	90.36	82.15	83.96	84.12	80.80	81.54
	1.3		97.99	88.12	85.85	85.64	82.22	81.79	82.53	78.28	
	1.5			87.20	85.89	80.39	81.70	78.51	78.90		
	1.7					81.42	79.88	78.84			
	1.9					81.51	79.20				

Table D.3: Value-Based Search Tests (SD-IC)

Algorithm: VB-EV, Test: SD-IC

		px									
		0.1	0.3	0.5	0.7	0.9	1.1	1.3	1.5	1.7	1.9
py	0.1	8676.38	644.88	303.17	184.00	171.94					
	0.3	7658.15	566.29	275.63	164.48	152.77	127.09	113.61			
	0.5	7645.16	486.47	245.19	164.20	131.32	123.52	92.90	100.86		
	0.7	7295.56	486.34	236.98	163.50	136.72	103.81	106.87	107.23		
	0.9	8738.06	409.01	214.61	177.02	125.60	119.61	106.67	95.44	82.85	68.33
	1.1		522.06	227.70	159.83	134.74	92.53	97.65	82.13	82.64	82.61
	1.3		509.46	219.16	162.26	126.64	104.17	108.21	104.28	92.67	
	1.5			224.78	161.81	124.81	90.05	88.42	86.19		
	1.7					135.66	107.73	92.07			
	1.9					121.51	94.22				

Algorithm: VB-TEV, Test: SD-IC

		px									
		0.1	0.3	0.5	0.7	0.9	1.1	1.3	1.5	1.7	1.9
py	0.1	1547.17	315.36	165.15	131.45	83.24					
	0.3	1404.15	244.62	131.32	100.76	75.21	60.57	55.39			
	0.5	1261.88	252.23	124.79	86.41	70.28	59.22	54.43	46.26		
	0.7	1312.79	244.99	104.29	96.13	74.33	56.67	53.31	52.02		
	0.9	1344.97	174.32	107.44	75.59	63.53	66.33	55.55	48.49	45.89	44.31
	1.1		185.74	120.65	83.04	70.94	60.93	56.03	50.86	46.36	47.43
	1.3		194.10	117.84	83.92	73.22	58.20	50.88	49.98	51.78	
	1.5			112.77	80.91	66.88	57.05	53.81	46.40		
	1.7					62.20	59.37	52.61			
	1.9					73.07	59.80				

Algorithm: VB-CEV, Test: SD-IC

		px									
		0.1	0.3	0.5	0.7	0.9	1.1	1.3	1.5	1.7	1.9
py	0.1	146.25	87.92	69.56	76.19	70.23					
	0.3	75.09	46.57	33.51	35.14	35.30	32.92	32.02			
	0.5	56.08	34.27	31.39	23.79	20.69	24.59	27.13	25.03		
	0.7	50.78	27.68	23.37	23.61	21.09	22.57	18.05	20.43		
	0.9	45.83	26.14	24.69	22.65	20.04	17.63	16.61	19.20	19.56	20.18
	1.1		25.63	19.73	17.61	18.21	18.65	17.75	16.23	18.00	15.21
	1.3		21.67	19.10	16.20	15.09	15.48	15.74	15.00	17.60	
	1.5			17.41	15.22	14.17	13.71	13.63	14.75		
	1.7					16.72	14.44	14.09			
	1.9					14.60	15.57				

Table D.4: Value-Based Search Tests (MN-NNV)

Algorithm: VB-EV, Test: MN-NNV

		px									
		0.1	0.3	0.5	0.7	0.9	1.1	1.3	1.5	1.7	1.9
py	0.1	8.00	7.30	7.70	7.71	7.95					
	0.3	7.09	7.01	7.03	7.04	7.04	7.04	7.08			
	0.5	7.01	7.00	7.03	7.00	7.00	7.00	7.00	7.04		
	0.7	7.04	7.00	7.01	7.00	7.00	7.00	7.00	7.00		
	0.9	7.00	7.00	7.00	7.00	7.00	7.00	7.00	7.00	7.00	7.00
	1.1		7.00	7.00	7.00	7.01	7.00	7.00	7.00	7.00	7.00
	1.3		7.00	7.00	7.00	7.00	7.00	7.00	7.00	7.00	7.00
	1.5			7.00	7.00	7.00	7.00	7.00	7.00		
	1.7					7.00	7.00	7.00			
	1.9					7.00	7.00				

Algorithm: VB-TEV, Test: MN-NNV

		px									
		0.1	0.3	0.5	0.7	0.9	1.1	1.3	1.5	1.7	1.9
py	0.1	18.54	17.71	19.49	18.02	19.80					
	0.3	14.96	9.60	8.66	9.89	8.97	9.80	9.46			
	0.5	9.72	7.75	7.42	7.56	7.64	7.70	7.83	7.70		
	0.7	8.30	8.01	7.38	7.29	7.44	7.26	7.33	7.58		
	0.9	8.31	7.24	7.08	7.11	7.15	7.16	7.28	7.30	7.34	7.34
	1.1		7.12	7.09	7.06	7.06	7.15	7.08	7.16	7.12	7.25
	1.3		7.17	7.03	7.03	7.08	7.09	7.05	7.15	7.12	
	1.5			7.01	7.03	7.01	7.01	7.08	7.05		
	1.7					7.00	7.01	7.01			
	1.9					7.00	7.01				

Algorithm: VB-CEV, Test: MN-NNV

		px									
		0.1	0.3	0.5	0.7	0.9	1.1	1.3	1.5	1.7	1.9
py	0.1	88.54	110.03	112.38	122.09	118.91					
	0.3	84.90	96.91	110.67	120.19	121.41	136.70	112.42			
	0.5	76.14	96.24	102.56	118.19	110.50	118.86	115.56	113.33		
	0.7	75.59	84.97	97.79	97.41	107.78	115.22	126.89	111.26		
	0.9	67.76	81.36	89.29	97.67	105.51	100.20	121.40	110.35	113.22	105.61
	1.1		75.39	78.95	86.33	88.58	108.74	101.88	99.16	110.03	102.96
	1.3		76.99	88.80	89.08	87.76	94.21	96.05	91.59	106.62	
	1.5			82.08	79.67	90.70	84.97	94.09	93.65		
	1.7					85.50	87.35	87.34			
	1.9					78.97	84.00				

Table D.5: Value-Based Search Tests (SD-NNV)

Algorithm: VB-EV, Test: SD-NNV

		px									
		0.1	0.3	0.5	0.7	0.9	1.1	1.3	1.5	1.7	1.9
py	0.1	4.45	0.85	1.57	1.30	1.56					
	0.3	0.48	0.11	0.16	0.19	0.19	0.19	0.27			
	0.5	0.11	0.00	0.16	0.00	0.00	0.00	0.00	0.25		
	0.7	0.25	0.00	0.11	0.00	0.00	0.00	0.00	0.00		
	0.9	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	1.1		0.00	0.00	0.00	0.11	0.00	0.00	0.00	0.00	0.00
	1.3		0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
	1.5			0.00	0.00	0.00	0.00	0.00	0.00	0.00	
	1.7					0.00	0.00	0.00			
	1.9					0.00	0.00				

Algorithm: VB-TEV, Test: SD-NNV

		px									
		0.1	0.3	0.5	0.7	0.9	1.1	1.3	1.5	1.7	1.9
py	0.1	18.40	14.10	13.63	11.56	9.14					
	0.3	14.14	6.42	2.76	4.15	2.35	2.57	2.81			
	0.5	6.90	1.72	0.76	0.90	1.27	1.18	1.11	1.07		
	0.7	4.35	3.92	0.86	0.64	1.08	0.59	0.63	0.91		
	0.9	4.81	0.83	0.31	0.32	0.36	0.37	0.84	0.68	0.84	0.62
	1.1		0.91	0.33	0.24	0.24	0.51	0.27	0.43	0.43	0.58
	1.3		1.35	0.16	0.16	0.35	0.36	0.27	0.39	0.70	
	1.5			0.11	0.16	0.11	0.11	0.27	0.22		
	1.7					0.00	0.11	0.11			
	1.9					0.00	0.11				

Algorithm: VB-CEV, Test: SD-NNV

		px									
		0.1	0.3	0.5	0.7	0.9	1.1	1.3	1.5	1.7	1.9
py	0.1	51.74	56.15	49.99	60.20	58.73					
	0.3	46.15	51.49	46.83	50.64	58.05	59.69	52.64			
	0.5	38.96	46.04	54.15	51.01	40.02	53.16	51.85	49.87		
	0.7	44.45	41.65	46.57	46.31	45.99	55.01	51.11	50.64		
	0.9	38.81	42.82	47.29	53.36	52.74	43.30	46.81	49.41	51.84	53.06
	1.1		42.42	43.43	42.88	44.32	53.75	50.09	45.36	53.16	44.08
	1.3		46.86	50.42	43.36	41.14	44.68	46.31	40.09	57.09	
	1.5			48.21	39.51	40.11	39.28	40.09	43.13		
	1.7					48.75	45.48	46.92			
	1.9					43.17	47.47				

Table D.6: Value-Based Search Tests (MN-EV)

Algorithm: VB-CEV, Test: MN-EV

		px									
		0.1	0.3	0.5	0.7	0.9	1.1	1.3	1.5	1.7	1.9
py	0.1	235.80	235.76	236.40	234.39	234.07					
	0.3	237.20	235.94	236.65	236.55	236.43	237.75	236.07			
	0.5	236.35	237.12	238.16	238.25	237.31	236.29	237.57	237.16		
	0.7	237.32	237.10	236.16	236.01	237.38	237.53	236.97	236.50		
	0.9	238.52	236.62	236.67	237.44	237.55	237.01	236.60	237.13	236.44	237.73
	1.1		238.99	238.67	238.78	237.76	237.82	238.24	238.29	237.58	237.07
	1.3		238.69	238.99	240.28	241.47	240.26	237.34	238.92	239.80	
	1.5			240.10	240.96	239.73	240.48	239.70	240.26		
	1.7					240.73	242.25	240.75			
	1.9					242.00	242.84				

Table D.7: Value-Based Search Tests (SD-EV)

Algorithm: VB-CEV, Test: SD-EV

		px									
		0.1	0.3	0.5	0.7	0.9	1.1	1.3	1.5	1.7	1.9
py	0.1	8.51	6.63	8.51	8.34	7.49					
	0.3	8.23	8.06	7.89	8.49	6.96	7.52	8.14			
	0.5	7.60	9.17	8.51	8.51	8.74	7.15	7.80	8.26		
	0.7	7.38	8.41	9.56	9.04	7.62	9.36	9.71	8.14		
	0.9	6.31	7.73	8.40	8.38	6.93	8.35	8.27	7.82	7.29	8.17
	1.1		7.93	8.41	8.21	7.34	7.41	6.86	7.68	7.23	7.83
	1.3		7.32	6.87	7.35	7.84	6.52	6.03	6.35	5.96	
	1.5			5.79	6.94	6.01	5.94	6.54	6.36		
	1.7					5.86	6.25	6.63			
	1.9					6.25	7.43				

Table D.8: Value-Based Search Tests (MN-ITR)

Algorithm: VB-CEV, Test: MN-ITR

		px									
		0.1	0.3	0.5	0.7	0.9	1.1	1.3	1.5	1.7	1.9
py	0.1	12.35	12.35	12.10	12.04	11.96					
	0.3	7.91	7.96	8.07	7.99	8.01	8.03	7.88			
	0.5	7.04	6.92	6.99	6.96	7.04	6.99	7.04	7.04		
	0.7	6.39	6.47	6.47	6.56	6.42	6.47	6.39	6.41		
	0.9	6.12	6.15	6.08	6.11	6.14	6.10	6.14	6.08	6.06	6.09
	1.1		6.00	6.00	6.00	6.01	6.01	6.00	6.03	6.00	6.00
	1.3		6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00	
	1.5			6.00	6.00	6.00	6.00	6.00	6.00		
	1.7					6.00	6.00	6.00			
	1.9					6.00	5.99				

Table D.9: Value-Based Search Tests (SD-ITR)

Algorithm: VB-CEV, Test: SD-ITR

		px									
		0.1	0.3	0.5	0.7	0.9	1.1	1.3	1.5	1.7	1.9
py	0.1	1.21	1.11	1.16	1.07	1.06					
	0.3	0.58	0.58	0.55	0.49	0.56	0.57	0.56			
	0.5	0.46	0.38	0.30	0.43	0.34	0.41	0.43	0.30		
	0.7	0.49	0.50	0.50	0.50	0.50	0.50	0.49	0.50		
	0.9	0.33	0.36	0.27	0.32	0.35	0.30	0.35	0.27	0.24	0.28
	1.1		0.00	0.00	0.00	0.11	0.11	0.00	0.16	0.00	0.00
	1.3		0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
	1.5			0.00	0.00	0.00	0.00	0.00	0.00		
	1.7					0.00	0.00	0.00			
	1.9					0.00	0.11				

Appendix E

TALL GRAPH (T2) EXAMPLE

E.1 Graph T2's Topology

- Time horizon: 75
- Number of nodes: 1123
- Total number of arcs: 6113
- Number of different paths: $\approx 2.45 \times 10^{46}$
 - Table (E.1) shows the distribution of nodes in the graph.
- The number of arcs from each node is a random integer from the set $\{2, \dots, 10\}$. (Terminal nodes have no arcs.)
- Arc lengths are drawn randomly from a uniform probability distribution on $[0,1]$. With probability $1/10$, the length of the arc is multiplied by 10. This provides a mechanism to place isolated large rewards in the graph.

E.2 Paths in Graph T2

- The myopic path through the graph is in Table (E.2).
 - Its value is 210.08.
- Table (E.3) contains the optimal path through this graph.
 - Its value is 334.33.

Table E.1: Graph T2's Topological Structure

Time Index	Node Indexes	Number of Nodes	Time Index	Node Indexes	Number of Nodes
0	0-0	1	38	148-159	12
1	1-1	1	39	160-165	6
2	2-3	2	40	166-173	8
3	4-4	1	41	174-185	12
4	5-6	2	42	186-192	7
5	7-7	1	43	193-208	16
6	8-10	3	44	209-219	11
7	11-11	1	45	220-231	12
8	12-13	2	46	232-240	9
9	14-15	2	47	241-255	15
10	16-18	3	48	256-265	10
11	19-20	2	49	266-276	11
12	21-22	2	50	277-291	15
13	23-24	2	51	292-314	23
14	25-26	2	52	315-331	17
15	27-28	2	53	332-351	20
16	29-32	4	54	352-364	13
17	33-34	2	55	365-381	17
18	35-36	2	56	382-396	15
19	37-39	3	57	397-420	24
20	40-43	4	58	421-447	27
21	44-47	4	59	448-479	32
22	48-50	3	60	480-513	34
23	51-55	5	61	514-538	25
24	56-61	6	62	539-574	36
25	62-64	3	63	575-598	24
26	65-70	6	64	599-627	29
27	71-77	7	65	628-665	38
28	78-82	5	66	666-704	39
29	83-88	6	67	705-737	33
30	89-93	5	68	738-790	53
31	94-102	9	69	791-824	34
32	103-109	7	70	825-874	50
33	110-114	5	71	875-921	47
34	115-122	8	72	922-967	46
35	123-132	10	73	968-1026	59
36	133-138	6	74	1027-1073	47
37	139-147	9	75	1074-1122	49

Table E.2: The Myopic Path through Graph T2

Time	Node	Decision	Value	Time	Node	Decision	Value
0	0	1	210.0816	38	152	1	119.0604
1	1	1	209.2275	39	160	1	118.0763
2	3	1	199.7757	40	167	1	117.2653
3	4	1	199.0192	41	180	1	108.7949
4	6	1	198.2702	42	188	1	107.8283
5	7	1	197.8909	43	198	1	106.8926
6	10	1	197.0146	44	213	1	106.2054
7	11	1	196.9377	45	228	1	105.3717
8	13	1	196.1078	46	232	1	105.1456
9	14	1	195.3994	47	249	1	104.2535
10	18	1	194.0178	48	260	1	96.0556
11	20	1	187.8345	49	276	1	90.5275
12	22	1	187.6246	50	281	1	86.5506
13	24	1	187.0323	51	303	1	80.5223
14	25	1	182.8889	52	330	1	79.0873
15	28	1	173.7708	53	351	1	78.1548
16	30	1	172.9617	54	363	1	77.1708
17	34	1	172.0015	55	380	1	76.2439
18	36	1	171.6347	56	384	1	75.2794
19	39	1	171.1503	57	413	1	74.4322
20	42	1	161.8006	58	436	1	66.7788
21	44	1	161.2368	59	465	1	61.2769
22	48	1	160.5456	60	506	1	60.3785
23	54	1	159.6181	61	536	1	59.4970
24	58	1	158.8905	62	558	1	50.5432
25	62	1	158.2082	63	591	1	40.6857
26	66	1	155.3956	64	626	1	33.2621
27	76	1	152.8595	65	642	1	30.6555
28	79	1	146.9111	66	671	1	29.7898
29	88	1	143.7677	67	712	1	28.8249
30	89	1	142.9727	68	784	1	27.9240
31	98	1	136.8278	69	818	1	20.8532
32	103	1	136.1310	70	864	1	11.3098
33	114	1	135.1360	71	917	1	10.5798
34	122	1	130.5591	72	958	1	3.6554
35	126	1	126.9277	73	975	1	1.7725
36	135	1	126.1588	74	1038	1	0.8079
37	142	1	119.9342	75	1099	-	0.0000

Table E.3: The Optimal Graph T2

Time	Node	Decision	Value	Time	Node	Decision	Value
0	0	1	334.3291	38	154	1	201.3372
1	1	1	333.4750	39	165	8	194.2245
2	3	1	324.0231	40	166	1	194.1686
3	4	2	323.2667	41	178	1	184.5379
4	5	1	322.5395	42	186	7	176.0189
5	7	3	321.6114	43	200	1	175.5383
6	9	1	321.3582	44	215	2	170.1133
7	11	2	320.6555	45	223	3	162.6622
8	12	1	320.3701	46	240	1	161.8798
9	14	1	313.1733	47	255	1	152.5817
10	18	1	311.7917	48	264	1	144.6881
11	20	1	305.6084	49	272	1	137.6081
12	22	1	305.3984	50	289	4	132.0376
13	24	1	304.8062	51	314	1	131.4690
14	25	1	300.6627	52	326	2	123.4509
15	28	3	291.5447	53	333	1	122.8505
16	29	1	291.0098	54	364	3	114.0200
17	34	2	281.5970	55	371	1	113.3747
18	35	2	281.2913	56	391	1	106.7265
19	39	1	280.5884	57	402	1	97.9396
20	42	1	271.2387	58	443	4	89.5806
21	44	1	270.6750	59	450	1	88.8542
22	48	3	269.9837	60	494	7	79.3712
23	53	1	269.1031	61	535	1	78.8413
24	57	3	263.6098	62	557	1	69.8393
25	62	1	263.5759	63	594	1	60.7747
26	66	1	260.7633	64	621	1	57.0633
27	76	5	258.2272	65	645	1	48.3155
28	82	1	258.1742	66	679	3	47.4450
29	87	1	250.2973	67	718	1	46.8212
30	92	5	242.6753	68	758	1	37.5045
31	96	1	242.0300	69	812	2	28.0480
32	108	2	232.1905	70	839	1	27.1771
33	113	1	227.3515	71	899	2	21.5516
34	121	3	221.9677	72	936	3	16.2353
35	132	1	221.3336	73	1009	1	15.5351
36	133	3	211.5716	74	1069	1	6.1690
37	140	1	210.8715	75	1083	-	0.0000

Appendix F

TALL GRAPH (T2) COMPUTATIONAL EXPERIMENTS

- 210.08 is the length of the myopic path through the graph.
- 334.33 is the length of the longest path through the graph.

Table F.1: Monte Carlo Search Tests

Algorithm	MN-IC	SD-IC	MN-NNV	SD-NNV
MC-EV	1120.91	229.66	6.15	1.54
MC-TEV	548.79	125.79	11.65	4.49
MC-CEV	204.84	151.82	67.11	47.53
Algorithm	MN-EV	SD-EV	MN-ITR	SD-ITR
MC-CEV	218.41	13.36	5.99	0.11

Table F.2: Value-Based Search Tests (MN-IC)

Algorithm: VB-EV, Test: MN-IC

		px									
		0.1	0.3	0.5	0.7	0.9	1.1	1.3	1.5	1.7	1.9
py	0.1	3838.44	1192.17	998.15	930.08	905.80					
	0.3	4436.57	1227.01	921.40	832.81	767.20	753.30	715.46			
	0.5	5049.50	1181.49	850.75	786.33	716.04	714.41	679.34	649.74		
	0.7	4644.70	1135.61	874.71	760.05	717.04	686.05	631.64	613.12		
	0.9	4930.94	1197.97	852.40	758.12	708.48	669.11	657.56	611.35	578.41	592.27
	1.1		1155.51	862.33	739.64	707.16	655.85	641.14	593.73	574.15	575.64
	1.3		1169.40	838.06	765.70	679.41	670.39	630.91	606.65	590.66	
	1.5			862.62	753.12	697.10	640.89	620.44	599.33		
	1.7					686.67	669.91	609.38			
	1.9					689.56	632.76				

Algorithm: VB-TEV, Test: MN-IC

		px									
		0.1	0.3	0.5	0.7	0.9	1.1	1.3	1.5	1.7	1.9
py	0.1	1397.38	734.90	545.26	550.27	520.15					
	0.3	1362.51	625.50	484.46	444.34	415.35	401.82	382.51			
	0.5	1477.06	609.04	487.59	423.69	405.05	373.77	368.50	343.02		
	0.7	1506.26	562.95	468.25	416.54	374.45	370.74	363.21	343.85		
	0.9	1358.58	618.10	469.75	411.04	369.64	356.10	343.01	337.51	324.20	319.99
	1.1		587.01	473.24	403.02	364.07	350.56	335.10	325.12	316.99	314.96
	1.3		605.75	465.65	406.84	368.20	339.07	328.96	316.20	311.02	
	1.5			461.49	404.41	367.20	353.73	334.15	325.48		
	1.7					365.60	351.49	331.95			
	1.9					364.46	347.18				

Algorithm: VB-CEV, Test: MN-IC

		px									
		0.1	0.3	0.5	0.7	0.9	1.1	1.3	1.5	1.7	1.9
py	0.1	364.19	264.41	242.46	232.01	236.78					
	0.3	205.72	153.60	140.07	140.03	141.59	129.30	132.74			
	0.5	146.68	118.54	111.72	101.33	105.34	104.29	98.08	104.04		
	0.7	114.66	94.54	95.11	89.75	88.12	87.54	87.28	92.61		
	0.9	99.91	81.49	86.10	80.79	78.42	82.91	82.91	77.44	74.92	81.56
	1.1		75.14	79.80	74.00	73.39	70.84	75.60	75.40	69.58	71.53
	1.3		67.56	67.12	69.33	67.24	69.61	68.14	64.34	68.42	
	1.5			67.46	63.12	61.14	66.25	66.05	64.17		
	1.7					59.31	64.30	64.61			
	1.9					60.46	60.49				

Table F.3: Value-Based Search Tests (SD-IC)

Algorithm: VB-EV, Test: SD-IC

		px									
		0.1	0.3	0.5	0.7	0.9	1.1	1.3	1.5	1.7	1.9
py	0.1	5139.97	391.37	190.01	163.00	151.38					
	0.3	3853.75	425.42	189.54	159.63	132.22	105.67	97.80			
	0.5	5142.83	359.54	150.11	143.66	131.01	119.57	98.53	104.78		
	0.7	4356.51	358.39	192.34	137.78	111.95	101.08	90.92	80.65		
	0.9	4203.78	397.91	162.32	137.51	102.70	87.57	105.81	88.71	83.22	78.44
	1.1		342.40	161.09	112.46	113.65	94.80	93.70	87.14	84.71	90.72
	1.3		397.40	148.88	114.01	99.27	97.41	87.94	91.93	94.46	
	1.5			163.16	125.03	117.80	96.19	97.31	94.99		
	1.7					94.71	100.39	98.10			
	1.9					114.35	87.21				

Algorithm: VB-TEV, Test: SD-IC

		px									
		0.1	0.3	0.5	0.7	0.9	1.1	1.3	1.5	1.7	1.9
py	0.1	1192.17	295.53	138.73	121.27	87.82					
	0.3	1058.54	204.05	116.45	82.59	69.50	72.04	71.97			
	0.5	1516.24	194.07	101.12	65.26	67.26	61.49	61.28	59.19		
	0.7	1175.57	205.25	101.91	78.35	66.28	62.61	58.11	52.46		
	0.9	1101.35	204.50	100.89	84.55	63.51	64.67	62.14	55.29	53.26	46.03
	1.1		171.56	107.12	73.19	67.10	55.84	49.98	52.92	52.46	43.08
	1.3		155.12	114.94	88.00	68.61	60.04	64.89	49.39	55.28	
	1.5			96.91	68.21	61.24	60.69	62.36	46.43		
	1.7					58.85	65.39	54.40			
	1.9					65.60	60.69				

Algorithm: VB-CEV, Test: SD-IC

		px									
		0.1	0.3	0.5	0.7	0.9	1.1	1.3	1.5	1.7	1.9
py	0.1	191.20	105.06	72.11	84.92	89.22					
	0.3	91.34	65.01	55.16	50.42	46.53	43.73	49.37			
	0.5	70.62	42.20	38.51	36.03	38.46	33.24	32.15	32.56		
	0.7	49.76	32.62	35.85	28.71	34.26	26.11	32.48	31.40		
	0.9	43.36	31.39	31.23	28.00	25.42	27.02	25.26	28.02	24.06	26.16
	1.1		27.28	24.59	25.61	20.72	21.87	24.97	25.26	22.25	24.83
	1.3		24.95	24.45	23.28	18.01	23.39	19.69	19.76	19.88	
	1.5			23.03	20.01	17.01	22.89	19.29	21.06		
	1.7					18.12	21.23	19.40			
	1.9					18.64	14.94				

Table F.4: Value-Based Search Tests (MN-NNV)

Algorithm: VB-EV, Test: MN-NNV

		px									
		0.1	0.3	0.5	0.7	0.9	1.1	1.3	1.5	1.7	1.9
py	0.1	4.97	4.74	4.59	4.59	4.59					
	0.3	4.28	4.00	4.00	4.01	4.03	4.03	4.00			
	0.5	4.01	4.00	4.00	4.00	4.00	4.00	4.00	4.00		
	0.7	4.05	4.00	4.00	4.00	4.00	4.00	4.00	4.00		
	0.9	4.03	4.00	4.00	4.00	4.00	4.00	4.00	4.00	4.00	4.00
	1.1		4.00	4.00	4.00	4.00	4.00	4.00	4.00	4.00	4.00
	1.3		4.00	4.00	4.00	4.00	4.00	4.00	4.00	4.00	
	1.5			4.00	4.00	4.00	4.00	4.00	4.00	4.00	
	1.7					4.00	4.00	4.00			
	1.9					4.00	4.00				

Algorithm: VB-TEV, Test: MN-NNV

		px									
		0.1	0.3	0.5	0.7	0.9	1.1	1.3	1.5	1.7	1.9
py	0.1	18.82	11.59	13.78	12.32	12.18					
	0.3	9.97	5.14	5.21	5.10	5.20	5.55	5.64			
	0.5	7.04	4.26	4.34	4.29	4.35	4.36	4.44	4.53		
	0.7	5.89	4.34	4.11	4.14	4.06	4.10	4.12	4.12		
	0.9	4.95	4.22	4.04	4.03	4.06	4.09	4.14	4.11	4.08	4.19
	1.1		4.06	4.03	4.05	4.06	4.03	4.03	4.04	4.04	4.05
	1.3		4.21	4.15	4.00	4.03	4.03	4.04	4.01	4.05	
	1.5			4.03	4.00	4.00	4.01	4.00	4.00		
	1.7					4.01	4.00	4.03			
	1.9					4.00	4.00				

Algorithm: VB-CEV, Test: MN-NNV

		px									
		0.1	0.3	0.5	0.7	0.9	1.1	1.3	1.5	1.7	1.9
py	0.1	104.35	106.41	103.40	116.44	111.84					
	0.3	88.47	104.31	111.12	104.04	95.90	113.76	107.36			
	0.5	97.51	97.72	102.25	118.28	110.58	109.26	121.28	104.42		
	0.7	110.29	114.15	110.75	112.85	120.85	114.36	121.56	106.12		
	0.9	112.14	131.24	110.31	119.61	121.19	109.29	106.20	128.56	130.16	111.08
	1.1		130.55	106.56	124.03	120.24	129.36	117.34	116.58	135.15	130.05
	1.3		142.28	139.78	128.19	125.49	126.83	125.84	139.06	121.10	
	1.5			127.20	138.89	141.57	131.59	123.67	133.70		
	1.7					143.30	128.15	123.03			
	1.9					135.79	127.70				

Table F.5: Value-Based Search Tests (SD-NNV)

Algorithm: VB-EV, Test: SD-NNV

		px									
		0.1	0.3	0.5	0.7	0.9	1.1	1.3	1.5	1.7	1.9
py	0.1	2.31	1.83	1.08	1.18	0.85					
	0.3	1.24	0.00	0.00	0.11	0.16	0.16	0.00			
	0.5	0.11	0.00	0.00	0.00	0.00	0.00	0.00	0.00		
	0.7	0.31	0.00	0.00	0.00	0.00	0.00	0.00	0.00		
	0.9	0.22	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	1.1		0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	1.3		0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
	1.5			0.00	0.00	0.00	0.00	0.00	0.00	0.00	
	1.7					0.00	0.00	0.00			
	1.9					0.00	0.00				

Algorithm: VB-TEV, Test: SD-NNV

		px									
		0.1	0.3	0.5	0.7	0.9	1.1	1.3	1.5	1.7	1.9
py	0.1	20.09	10.76	9.76	7.94	6.66					
	0.3	14.17	2.23	2.40	1.64	1.39	2.23	2.29			
	0.5	7.33	0.79	1.19	0.64	0.83	0.75	0.69	1.15		
	0.7	6.18	1.15	0.53	0.44	0.29	0.38	0.51	0.37		
	0.9	2.15	1.06	0.25	0.16	0.24	0.28	0.61	0.36	0.27	0.66
	1.1		0.37	0.16	0.22	0.29	0.16	0.16	0.19	0.19	0.22
	1.3		1.49	1.14	0.00	0.16	0.16	0.19	0.11	0.45	
	1.5			0.16	0.00	0.00	0.11	0.00	0.00		
	1.7					0.11	0.00	0.22			
	1.9					0.00	0.00				

Algorithm: VB-CEV, Test: SD-NNV

		px									
		0.1	0.3	0.5	0.7	0.9	1.1	1.3	1.5	1.7	1.9
py	0.1	77.54	61.58	59.02	74.53	73.39					
	0.3	61.96	70.55	73.77	69.89	61.84	75.50	69.25			
	0.5	67.34	62.60	64.56	71.29	70.98	72.28	74.99	57.25		
	0.7	71.44	68.03	74.63	66.90	71.48	66.64	74.57	72.81		
	0.9	66.35	83.47	71.11	71.91	70.17	70.14	61.77	79.72	74.92	78.33
	1.1		77.77	66.67	75.63	73.58	73.22	77.09	73.18	79.74	84.16
	1.3		75.29	75.19	73.38	67.27	79.06	75.96	72.77	68.31	
	1.5			73.47	70.64	68.20	82.94	68.84	71.08		
	1.7					69.94	76.61	68.59			
	1.9					70.80	63.30				

Table F.6: Value-Based Search Tests (MN-EV)

Algorithm: VB-CEV, Test: MN-EV

		px									
		0.1	0.3	0.5	0.7	0.9	1.1	1.3	1.5	1.7	1.9
py	0.1	263.82	262.20	264.51	261.42	263.56					
	0.3	264.53	267.16	266.06	267.79	265.44	266.93	266.57			
	0.5	268.80	269.72	269.18	268.80	269.13	267.23	266.13	269.77		
	0.7	267.86	269.20	269.23	268.73	267.43	270.20	270.22	269.90		
	0.9	267.87	269.28	269.30	268.11	267.94	267.13	269.27	269.62	268.33	269.33
	1.1		269.86	269.22	270.15	268.33	270.94	269.11	269.93	270.31	269.14
	1.3		270.66	269.28	269.48	269.89	270.15	270.92	270.22	271.38	
	1.5			269.76	272.02	271.59	271.83	270.65	270.06		
	1.7					271.93	272.25	273.08			
	1.9					271.74	271.71				

Table F.7: Value-Based Search Tests (SD-EV)

Algorithm: VB-CEV, Test: SD-EV

		px									
		0.1	0.3	0.5	0.7	0.9	1.1	1.3	1.5	1.7	1.9
py	0.1	9.41	8.55	9.14	8.04	7.44					
	0.3	9.78	8.27	8.94	9.27	7.63	8.23	9.13			
	0.5	8.14	9.92	9.24	9.73	8.37	8.04	8.19	8.27		
	0.7	8.14	8.88	9.65	7.65	7.92	8.80	9.26	8.19		
	0.9	8.85	10.38	7.30	7.88	8.40	7.92	8.99	8.81	8.12	9.42
	1.1		8.64	8.11	7.94	8.02	7.13	7.09	8.04	6.24	7.51
	1.3		7.65	7.79	8.18	8.46	7.33	8.00	8.00	8.54	
	1.5			6.89	6.72	6.68	7.10	7.91	7.86		
	1.7					6.85	6.67	6.64			
	1.9					7.40	6.55				

Table F.8: Value-Based Search Tests (MN-ITR)

Algorithm: VB-CEV, Test: MN-ITR

		px									
		0.1	0.3	0.5	0.7	0.9	1.1	1.3	1.5	1.7	1.9
py	0.1	12.76	12.61	12.66	12.54	12.35					
	0.3	8.10	8.11	8.11	8.25	8.26	8.21	8.19			
	0.5	7.06	7.08	7.14	7.12	7.11	7.10	7.12	7.09		
	0.7	6.60	6.59	6.60	6.65	6.61	6.69	6.69	6.64		
	0.9	6.12	6.15	6.19	6.10	6.20	6.10	6.22	6.20	6.21	6.26
	1.1		6.00	6.00	6.05	6.01	6.03	6.03	6.03	6.00	6.01
	1.3		6.00	6.00	6.00	6.01	6.00	6.00	6.00	6.00	
	1.5			6.00	6.00	6.00	6.00	6.00	6.00		
	1.7					6.00	6.00	6.00			
	1.9					6.00	6.00				

Table F.9: Value-Based Search Tests (SD-ITR)

Algorithm: VB-CEV, Test: SD-ITR

		px									
		0.1	0.3	0.5	0.7	0.9	1.1	1.3	1.5	1.7	1.9
py	0.1	1.11	0.93	1.04	1.18	1.18					
	0.3	0.59	0.67	0.50	0.54	0.63	0.57	0.58			
	0.5	0.43	0.35	0.35	0.37	0.32	0.41	0.37	0.33		
	0.7	0.49	0.50	0.49	0.48	0.49	0.47	0.47	0.48		
	0.9	0.33	0.36	0.39	0.30	0.40	0.30	0.42	0.40	0.41	0.44
	1.1		0.00	0.00	0.22	0.11	0.16	0.16	0.16	0.00	0.11
	1.3		0.00	0.00	0.00	0.11	0.00	0.00	0.00	0.00	
	1.5			0.00	0.00	0.00	0.00	0.00	0.00		
	1.7					0.00	0.00	0.00			
	1.9					0.00	0.00				

Appendix G

TALL GRAPH (T3) EXAMPLE

G.1 Graph T3's Topology

- Time horizon: 100
- Number of nodes: 28671
- Total number of arcs: 162447
- Number of different paths: $\approx 2.45 \times 10^{69}$
 - Table (G.1) shows the distribution of nodes in the graph.
- The number of arcs from each node is a random integer from the set $\{2, \dots, 10\}$. (Terminal nodes have no arcs.)
- Arc lengths are drawn randomly from a uniform probability distribution on $[0,1]$. With probability $1/10$, the length of the arc is multiplied by 10. This provides a mechanism to place isolated large rewards in the graph.

G.2 Paths in Graph T3

- The myopic path through the graph is in Table (G.2).
 - Its value is 284.07.
- Table (G.3) contains the optimal path through this graph.
 - Its value is 497.87.

Table G.1: Graph T3's Topological Structure

Time Index	Node Indexes	Number of Nodes	Time Index	Node Indexes	Number of Nodes
0	0-0	1	51	721-766	46
1	1-1	1	52	767-828	62
2	2-2	1	53	829-908	80
3	3-3	1	54	909-996	88
4	4-4	1	55	997-1089	93
5	5-5	1	56	1090-1188	99
6	6-8	3	57	1189-1288	100
7	9-11	3	58	1289-1374	86
8	12-13	2	59	1375-1469	95
9	14-15	2	60	1470-1599	130
10	16-17	2	61	1600-1729	130
11	18-21	4	62	1730-1889	160
12	22-23	2	63	1890-2051	162
13	24-26	3	64	2052-2213	162
14	27-31	5	65	2214-2331	118
15	32-34	3	66	2332-2512	181
16	35-40	6	67	2513-2737	225
17	41-46	6	68	2738-2887	150
18	47-49	3	69	2888-3058	171
19	50-54	5	70	3059-3257	199
20	55-58	4	71	3258-3458	201
21	59-63	5	72	3459-3756	298
22	64-68	5	73	3757-4077	321
23	69-78	10	74	4078-4279	202
24	79-87	9	75	4280-4583	304
25	88-96	9	76	4584-4908	325
26	97-104	8	77	4909-5395	487
27	105-112	8	78	5396-5661	266
28	113-125	13	79	5662-5990	329
29	126-135	10	80	5991-6389	399
30	136-149	14	81	6390-6806	417
31	150-163	14	82	6807-7323	517
32	164-180	17	83	7324-7975	652
33	181-190	10	84	7976-8393	418
34	191-202	12	85	8394-8929	536
35	203-216	14	86	8930-9833	904
36	217-234	18	87	9834-10754	921
37	235-248	14	88	10755-11704	950
38	249-275	27	89	11705-12640	936
39	276-293	18	90	12641-13367	727
40	294-322	29	91	13368-14566	1199
41	323-355	33	92	14567-15909	1343
42	356-388	33	93	15910-17403	1494
43	389-427	39	94	17404-18542	1139
44	428-449	22	95	18543-20211	1669
45	450-478	29	96	20212-21922	1711
46	479-523	45	97	21923-23498	1576
47	524-574	51	98	23499-25589	2091
48	575-629	55	99	25590-27137	1548
49	630-661	32	100	27138-28670	1533
50	662-720	59			

Table G.2: The Myopic Path through Graph T3

Time	Node	Decision	Value	Time	Node	Decision	Value
0	0	1	284.0708	50	665	1	162.9858
1	1	1	283.4280	51	753	1	162.0556
2	2	1	282.8277	52	827	1	161.2122
3	3	1	279.0354	53	884	1	160.2337
4	4	1	278.4634	54	934	1	155.2504
5	5	1	278.1216	55	1012	1	154.5313
6	8	1	277.3171	56	1168	1	153.7810
7	9	1	274.4151	57	1209	1	153.0157
8	13	1	273.9332	58	1357	1	148.3436
9	14	1	273.5353	59	1412	1	147.4169
10	16	1	269.4370	60	1470	1	144.3336
11	21	1	268.8016	61	1621	1	140.4715
12	22	1	265.2086	62	1776	1	139.4795
13	25	1	262.4592	63	2014	1	138.6042
14	27	1	261.6501	64	2179	1	131.7931
15	32	1	260.7850	65	2261	1	124.0582
16	37	1	251.3795	66	2422	1	117.1082
17	41	1	241.4543	67	2722	1	114.6319
18	49	1	240.7277	68	2829	1	113.6254
19	50	1	239.8929	69	2969	1	112.8681
20	55	1	239.2050	70	3257	1	110.6238
21	60	1	238.8196	71	3278	1	110.1296
22	65	1	237.8709	72	3602	1	109.1581
23	73	1	234.2355	73	4025	1	100.2082
24	80	1	224.6240	74	4118	1	99.2863
25	95	1	221.8175	75	4448	1	90.9084
26	104	1	217.3179	76	4878	1	90.0171
27	107	1	216.4182	77	5053	1	84.5072
28	122	1	215.4451	78	5646	1	75.7481
29	129	1	214.5619	79	5984	1	74.7527
30	147	1	214.0077	80	6059	1	66.7404
31	160	1	213.1073	81	6708	1	60.2238
32	172	1	207.1616	82	7315	1	56.4212
33	188	1	198.8386	83	7371	1	55.4565
34	191	1	197.8549	84	8073	1	45.8407
35	215	1	194.9413	85	8824	1	43.3403
36	227	1	194.1848	86	9686	1	42.4812
37	238	1	193.3983	87	10258	1	33.4063
38	272	1	192.4030	88	11191	1	32.4162
39	279	1	189.9145	89	12109	1	29.6577
40	314	1	189.4295	90	13177	1	27.6694
41	351	1	188.4390	91	14148	1	19.0952
42	373	1	178.7006	92	15571	1	18.1394
43	417	1	173.5649	93	16222	1	12.9400
44	430	1	172.5970	94	17501	1	7.3206
45	463	1	167.2727	95	19976	1	6.4230
46	482	1	166.3206	96	21057	1	5.6243
47	530	1	165.3733	97	22544	1	4.9077
48	586	1	164.4656	98	23619	1	1.6438
49	631	1	163.6746	99	26633	1	0.9341
				100	27902	1	0.0000

Table G.3: The Optimal Path through Graph T3

Time	Node	Decision	Value	Time	Node	Decision	Value
0	0	1	497.8750	50	679	1	292.1498
1	1	1	497.2321	51	729	1	287.5506
2	2	1	496.6318	52	818	1	282.4965
3	3	1	492.8396	53	851	1	277.1240
4	4	1	492.2675	54	974	1	268.8999
5	5	2	491.9258	55	1022	1	263.9052
6	6	1	491.4499	56	1144	1	254.0593
7	11	2	482.6624	57	1247	1	244.3584
8	12	1	482.3868	58	1360	1	243.6982
9	14	1	472.6127	59	1448	1	236.5792
10	16	1	468.5143	60	1495	2	228.5589
11	21	1	467.8790	61	1714	1	227.8201
12	22	3	464.2860	62	1833	1	221.4963
13	26	1	464.2839	63	1946	1	212.6435
14	29	3	455.0551	64	2143	6	203.6219
15	32	1	454.9059	65	2314	1	203.3286
16	37	1	445.5004	66	2458	1	196.2971
17	41	1	435.5752	67	2537	1	195.4062
18	49	2	434.8486	68	2884	7	185.9482
19	51	1	434.1850	69	2902	2	185.6070
20	56	3	433.2639	70	3070	1	176.6455
21	62	1	432.6903	71	3288	1	168.0701
22	64	1	425.1507	72	3616	1	159.5890
23	69	1	419.7743	73	4037	1	152.0348
24	81	1	415.9399	74	4199	5	151.1363
25	96	2	413.3354	75	4307	4	150.9826
26	104	4	405.9158	76	4858	1	150.2645
27	111	1	405.2157	77	5350	3	140.6487
28	118	1	395.6115	78	5524	2	140.4315
29	129	1	388.2459	79	5915	1	132.4686
30	147	1	387.6917	80	6001	1	122.9791
31	160	1	386.7913	81	6771	1	115.4518
32	172	1	380.8457	82	7213	1	106.7168
33	188	1	372.5227	83	7708	1	97.9836
34	191	5	371.5390	84	8247	1	96.6148
35	207	1	371.2159	85	8504	4	86.6368
36	218	2	361.8636	86	9682	4	86.2290
37	241	1	355.2630	87	10054	1	85.8763
38	271	3	345.4046	88	11667	1	76.1216
39	286	7	344.7056	89	12327	1	67.4022
40	317	1	344.2171	90	12903	7	66.4046
41	341	1	337.3523	91	14363	1	66.0790
42	373	2	329.0937	92	14942	1	58.8168
43	397	1	325.2154	93	17379	1	49.2752
44	442	2	320.0938	94	18258	1	41.9727
45	469	1	319.7730	95	18937	1	36.7879
46	503	4	310.0037	96	21470	1	28.7448
47	545	1	309.4582	97	22447	4	18.8943
48	595	1	299.9389	98	24194	1	18.8031
49	644	2	292.9306	99	25677	1	9.7696
				100	-	1	0.0000

Appendix H**TALL GRAPH (T3) COMPUTATIONAL EXPERIMENTS**

- 284.07 is the length of the myopic path through the graph.
- 497.87 is the length of the longest path through the graph.

Table H.1: Monte Carlo Search Tests

Algorithm	MN-EV	SD-EV	MN-ITR	SD-ITR
MC-CEV	434.03	5.77	193.89	1.29

Table H.2: Value-Based Search Tests (MN-EV)

Algorithm: VB-CEV, Test: MN-EV

		px									
		0.1	0.3	0.5	0.7	0.9	1.1	1.3	1.5	1.7	1.9
0.1		480.87	480.51	480.56	479.87	479.67					
0.3		481.34	480.71	480.64	480.38	480.63	480.09	479.76			
0.5		481.58	480.91	480.90	480.85	481.22	481.17	481.05	480.14		
0.7		481.38	481.64	481.20	480.97	480.96	481.00	480.46	480.56		
0.9		481.90	481.35	481.22	481.18	481.39	481.16	480.92	481.38	481.32	480.67
1.1			481.65	481.27	481.22	480.78	480.93	481.08	481.34	481.31	481.06
1.3			481.79	481.82	481.12	481.54	481.17	481.45	481.47	481.04	
1.5				481.40	481.04	481.95	481.39	481.20	481.53		
1.7						481.13	482.03	481.52			
1.9						481.84	481.36				

Table H.3: Value-Based Search Tests (SD-EV)

Algorithm: VB-CEV, Test: SD-EV

		px									
		0.1	0.3	0.5	0.7	0.9	1.1	1.3	1.5	1.7	1.9
0.1		2.26	2.67	3.03	3.13	3.11					
0.3		2.59	2.53	2.69	2.73	2.45	2.92	2.74			
0.5		2.31	2.59	2.66	2.52	2.66	2.87	2.48	2.66		
0.7		2.59	2.68	3.06	2.59	2.85	2.54	2.73	2.91		
0.9		2.40	2.83	3.06	2.77	2.97	2.72	2.63	2.76	2.63	3.21
1.1			2.47	2.59	2.56	2.61	3.11	2.72	2.55	3.01	3.47
1.3			2.81	3.15	2.58	2.75	2.93	2.84	2.89	2.42	
1.5				2.91	3.20	2.34	2.62	3.04	2.96		
1.7						2.84	2.54	2.44			
1.9						2.69	2.57				

Table H.4: Value-Based Search Tests (MN-ITR)

Algorithm: VB-CEV, Test: MN-ITR

		px									
		0.1	0.3	0.5	0.7	0.9	1.1	1.3	1.5	1.7	1.9
0.1		433.12	411.82	408.11	404.62	405.01					
0.3		269.30	261.23	259.26	259.19	258.57	257.86	258.01			
0.5		228.54	223.94	222.39	221.78	221.65	221.89	221.29	221.57		
0.7		210.50	206.36	205.49	205.03	204.99	205.06	204.69	204.40		
0.9		199.69	196.41	195.66	195.36	195.03	194.90	195.01	194.99	194.84	194.88
1.1			189.80	189.12	188.94	189.22	188.69	188.95	188.86	188.81	188.75
1.3			185.57	184.91	184.79	184.56	184.47	184.59	184.41	184.54	
1.5				181.81	181.80	181.41	181.53	181.28	181.30		
1.7						179.14	179.04	179.06			
1.9						177.31	177.32				

Table H.5: Value-Based Search Tests (SD-ITR)

Algorithm: VB-CEV, Test: SD-ITR

		px									
		0.1	0.3	0.5	0.7	0.9	1.1	1.3	1.5	1.7	1.9
0.1		9.53	6.34	8.16	8.32	6.87					
0.3		3.73	3.59	3.36	3.12	3.08	3.92	3.26			
0.5		2.71	2.65	2.34	2.46	2.37	2.50	2.32	1.98		
0.7		2.01	1.62	1.48	1.92	1.89	1.79	1.56	1.64		
0.9		1.69	1.35	1.42	1.54	1.33	1.37	1.32	1.35	1.46	1.48
1.1			1.24	1.30	1.28	1.25	1.11	1.24	1.30	1.24	1.08
1.3			1.20	1.22	1.19	0.99	1.22	1.01	0.98	1.10	
1.5				0.89	0.95	0.82	1.02	0.94	1.05		
1.7						0.99	0.89	0.74			
1.9						0.85	0.90				

Appendix I

WIDE GRAPH (W1) EXAMPLE

I.1 Graph W1's Topology

- Time horizon: 15
- Number of nodes: 1409
- Total number of arcs: 5338
- Number of different paths: $\approx 9.76 \times 10^9$
 - Table (I.1) shows the distribution of nodes in the graph.
- The number of arcs from each node is a random integer from the set $\{2, \dots, 10\}$.
(Terminal nodes have no arcs.)
- Arc lengths are drawn randomly from a uniform probability distribution on $[0,1]$. With probability $1/10$, the length of the arc is multiplied by 10. This provides a mechanism to place isolated large rewards in the graph.

I.2 Paths in Graph W1

- The myopic path through the graph is in Table (I.2).
 - Its value is 47.76.
- Table (I.3) contains the optimal path through this graph.
 - Its value is 72.25.

Table I.1: Graph W1's Topological Structure

Time Index	Node Indexes	Number of Nodes
0	0-0	1
1	1-1	1
2	2-3	2
3	4-8	5
4	9-11	3
5	12-17	6
6	18-27	10
7	28-44	17
8	45-78	34
9	79-124	46
10	125-191	67
11	192-287	96
12	288-441	154
13	442-584	143
14	585-901	317
15	902-1408	507

Table I.2: The Myopic Path through Graph W1

Time	Node	Decision	Value
0	0	1	47.759697
1	1	1	47.117405
2	3	1	38.411083
3	6	1	37.840477
4	10	1	36.906704
5	14	1	32.510818
6	18	1	31.576832
7	32	1	27.476669
8	50	1	22.478653
9	124	1	21.635031
10	148	1	20.655691
11	273	1	13.099613
12	384	1	10.517137
13	495	1	9.785456
14	693	1	1.139256
15	1055	-	0.000000

Table I.3: The Optimal Path through Graph W1

Time	Node	Decision	Value
0	0	1	72.254898
1	1	1	71.612602
2	3	2	62.906281
3	7	1	62.662468
4	10	6	56.054905
5	15	1	55.936859
6	24	1	47.209785
7	32	1	39.556355
8	50	6	34.558338
9	82	1	34.377548
10	141	3	28.010775
11	259	1	27.351818
12	365	1	17.578876
13	477	1	9.632466
14	826	1	8.676107
15	1059	-	0.000000

Appendix J

WIDE GRAPH (W1) COMPUTATIONAL EXPERIMENTS

- 47.76 is the length of the myopic path through the graph.
- 72.25 is the length of the longest path through the graph.

Table J.1: Monte Carlo Search Tests

Algorithm	MN-IC	SD-IC	MN-NNV	SD-NNV
MC-EV	726.60	354.90	338.55	124.90
MC-TEV	442.90	235.44	471.69	151.26
MC-CEV	315.06	184.15	587.06	178.04
Algorithm	MN-EV	SD-EV	MN-ITR	SD-ITR
MC-CEV	50.66	4.93	15.26	0.63

Table J.2: Value-Based Search Tests (MN-IC)

Algorithm: VB-EV, Test: MN-IC

		px									
		0.1	0.3	0.5	0.7	0.9	1.1	1.3	1.5	1.7	1.9
py	0.1	866.38	509.93	472.75	363.90	469.59					
	0.3	478.27	287.89	263.20	303.64	274.49	263.10	251.00			
	0.5	344.26	246.54	259.51	232.12	193.81	219.55	214.05	218.07		
	0.7	375.96	213.06	188.14	190.07	168.50	182.44	178.93	193.15		
	0.9	342.09	215.26	187.94	175.43	177.22	175.45	188.05	197.32	158.44	182.25
	1.1		178.89	189.31	187.00	166.74	157.05	158.40	156.36	183.07	148.64
	1.3		190.25	181.89	160.78	163.40	142.60	172.46	156.09	148.75	
	1.5			165.61	173.09	158.85	162.00	149.35	170.21		
	1.7					144.47	149.65	152.22			
	1.9					156.32	140.85				

Algorithm: VB-TEV, Test: MN-IC

		px									
		0.1	0.3	0.5	0.7	0.9	1.1	1.3	1.5	1.7	1.9
py	0.1	309.98	212.57	260.93	278.19	260.19					
	0.3	181.18	107.36	119.15	135.90	127.45	134.79	144.29			
	0.5	103.39	85.94	97.38	103.66	111.50	124.44	119.76	104.49		
	0.7	89.09	81.33	82.71	100.78	94.20	96.69	97.05	102.25		
	0.9	79.31	64.33	70.42	87.90	94.16	78.00	95.53	97.51	99.85	91.92
	1.1		63.39	76.84	70.17	81.50	79.11	78.69	84.59	75.22	99.62
	1.3		64.60	79.14	77.38	69.70	73.61	79.28	80.34	86.42	
	1.5			66.34	79.95	73.05	72.81	73.54	69.35		
	1.7					64.86	60.35	86.86			
	1.9					73.72	66.31				

Algorithm: VB-CEV, Test: MN-IC

		px									
		0.1	0.3	0.5	0.7	0.9	1.1	1.3	1.5	1.7	1.9
py	0.1	279.31	187.18	215.05	248.78	256.43					
	0.3	105.84	101.94	120.00	124.00	122.97	130.19	139.51			
	0.5	93.00	84.11	82.14	77.66	94.46	101.04	86.97	117.74		
	0.7	82.08	68.15	71.99	79.78	74.01	75.55	86.91	89.66		
	0.9	60.61	55.39	58.84	73.72	61.70	71.46	68.47	70.81	78.04	83.95
	1.1		58.55	61.11	61.46	64.17	70.20	62.77	72.16	67.83	64.79
	1.3		56.50	51.52	66.19	58.99	57.55	55.61	59.80	62.62	
	1.5			51.24	51.65	54.17	62.74	54.77	56.59		
	1.7					54.30	49.52	57.65			
	1.9					51.10	50.19				

Table J.3: Value-Based Search Tests (SD-IC)

Algorithm: VB-EV, Test: SD-IC

		px									
		0.1	0.3	0.5	0.7	0.9	1.1	1.3	1.5	1.7	1.9
py	0.1	802.67	384.16	296.08	216.00	244.86					
	0.3	425.67	188.27	168.51	182.63	144.22	139.99	140.03			
	0.5	329.01	177.13	168.34	127.20	109.41	114.57	113.87	117.16		
	0.7	298.12	154.91	114.02	108.16	79.06	98.28	91.58	112.08		
	0.9	375.04	141.50	92.12	108.43	110.65	102.96	106.12	99.89	87.28	101.87
	1.1		100.02	94.86	101.27	94.00	91.97	85.22	91.36	98.73	84.64
	1.3		103.42	97.75	89.51	85.87	83.77	88.30	85.59	80.06	
	1.5			96.37	91.25	75.66	91.89	71.85	90.37		
	1.7					66.87	84.65	81.67			
	1.9					84.45	76.44				

Algorithm: VB-TEV, Test: SD-IC

		px									
		0.1	0.3	0.5	0.7	0.9	1.1	1.3	1.5	1.7	1.9
py	0.1	240.55	167.67	173.30	194.93	157.48					
	0.3	163.16	81.18	82.08	84.30	85.43	104.54	95.83			
	0.5	78.19	54.99	76.96	73.34	76.11	86.58	79.64	73.98		
	0.7	69.65	59.51	51.99	69.61	73.80	69.72	66.76	60.14		
	0.9	66.19	43.82	43.69	58.75	69.34	55.35	56.82	71.16	66.42	54.97
	1.1		40.46	53.98	47.23	48.86	52.78	46.22	50.85	47.95	70.26
	1.3		52.04	52.24	43.27	46.24	64.49	59.71	52.20	61.09	
	1.5			39.97	50.47	43.58	54.04	55.13	45.05		
	1.7					44.43	35.36	62.59			
	1.9					51.98	45.79				

Algorithm: VB-CEV, Test: SD-IC

		px									
		0.1	0.3	0.5	0.7	0.9	1.1	1.3	1.5	1.7	1.9
py	0.1	212.59	136.66	149.99	153.21	166.64					
	0.3	66.16	73.42	72.88	82.63	81.39	76.04	83.19			
	0.5	64.97	56.47	47.42	54.51	64.77	63.48	61.51	66.95		
	0.7	57.02	49.42	49.42	48.73	52.62	44.38	59.16	54.31		
	0.9	39.65	29.63	38.38	44.58	40.64	47.88	39.72	42.01	49.52	49.98
	1.1		37.02	42.30	38.93	42.00	46.47	40.18	43.44	37.36	40.87
	1.3		30.73	31.01	39.73	36.15	34.56	36.62	37.60	38.49	
	1.5			31.80	22.54	30.74	36.51	31.53	39.00		
	1.7					34.89	27.12	36.34			
	1.9					26.90	27.60				

Table J.4: Value-Based Search Tests (MN-NNV)

Algorithm: VB-EV, Test: MN-NNV

		px									
		0.1	0.3	0.5	0.7	0.9	1.1	1.3	1.5	1.7	1.9
py	0.1	729.44	783.44	771.44	836.20	748.60					
	0.3	733.01	761.50	774.16	722.27	743.33	754.76	771.96			
	0.5	754.49	745.20	709.79	727.42	785.89	737.33	748.14	740.67		
	0.7	665.20	750.73	763.29	753.20	779.85	760.70	761.75	749.75		
	0.9	694.34	701.71	717.66	757.49	753.73	751.73	724.94	701.75	781.76	734.46
	1.1		736.27	696.35	702.23	743.34	768.76	756.71	766.23	707.04	782.66
	1.3		699.19	697.58	738.55	732.89	783.84	713.49	750.06	766.56	
	1.5			725.46	701.00	723.01	729.71	746.59	707.62		
	1.7					748.26	747.60	738.12			
	1.9					722.20	756.84				

Algorithm: VB-TEV, Test: MN-NNV

		px									
		0.1	0.3	0.5	0.7	0.9	1.1	1.3	1.5	1.7	1.9
py	0.1	954.35	1005.02	944.75	923.74	931.14					
	0.3	955.04	1034.51	1000.02	958.67	978.00	974.58	947.12			
	0.5	1015.69	1031.84	1005.51	981.36	960.14	935.85	940.61	977.89		
	0.7	1022.38	1021.99	1007.11	959.11	981.54	972.71	968.45	944.64		
	0.9	1034.94	1052.97	1029.29	975.44	960.85	1006.41	944.16	951.48	939.95	951.89
	1.1		1047.29	1002.15	1019.98	975.35	989.74	981.54	962.77	996.29	924.88
	1.3		1038.84	978.02	976.83	1010.04	1007.71	981.41	969.40	957.06	
	1.5			1014.92	967.54	987.00	996.09	993.39	1003.21		
	1.7					1016.05	1030.95	944.26			
	1.9					982.90	1008.70				

Algorithm: VB-CEV, Test: MN-NNV

		px									
		0.1	0.3	0.5	0.7	0.9	1.1	1.3	1.5	1.7	1.9
py	0.1	974.51	1029.75	989.41	944.59	943.04					
	0.3	1045.12	1041.60	991.51	987.01	991.49	968.01	951.23			
	0.5	1032.04	1039.16	1031.61	1052.47	1004.08	984.46	1024.08	939.67		
	0.7	1037.51	1061.92	1042.86	1013.85	1037.92	1026.84	997.98	981.40		
	0.9	1079.44	1080.26	1071.58	1016.09	1059.51	1025.88	1030.10	1023.06	1000.77	980.45
	1.1		1064.72	1054.28	1049.21	1041.19	1018.19	1043.33	1007.39	1019.65	1035.60
	1.3		1059.42	1080.28	1020.06	1048.69	1054.54	1065.12	1046.60	1034.89	
	1.5			1078.31	1066.71	1061.91	1026.17	1058.75	1056.56		
	1.7					1060.56	1077.12	1045.42			
	1.9					1066.12	1070.28				

Table J.5: Value-Based Search Tests (SD-NNV)

Algorithm: VB-EV, Test: SD-NNV

		px									
		0.1	0.3	0.5	0.7	0.9	1.1	1.3	1.5	1.7	1.9
py	0.1	284.11	250.33	216.44	189.76	193.04					
	0.3	309.67	218.63	201.54	211.35	189.92	186.25	192.38			
	0.5	290.78	230.16	229.92	197.94	183.89	177.49	188.27	195.80		
	0.7	292.27	232.36	196.14	190.11	166.41	176.08	180.88	216.11		
	0.9	311.81	210.23	165.06	203.60	207.27	199.07	208.24	198.38	187.63	199.08
	1.1		203.59	184.89	190.85	190.91	192.58	180.60	190.08	201.95	189.12
	1.3		211.65	179.27	176.47	189.00	179.36	198.72	194.67	190.84	
	1.5			192.64	187.37	174.63	204.96	172.54	207.83		
	1.7					163.97	198.66	196.01			
	1.9					194.53	179.67				

Algorithm: VB-TEV, Test: SD-NNV

		px									
		0.1	0.3	0.5	0.7	0.9	1.1	1.3	1.5	1.7	1.9
py	0.1	185.22	172.82	188.77	199.81	180.93					
	0.3	203.74	165.72	165.92	167.52	165.30	198.45	190.96			
	0.5	161.01	146.34	183.03	169.47	186.82	200.57	191.20	183.95		
	0.7	161.80	167.87	154.53	181.84	194.59	186.45	187.04	170.77		
	0.9	162.03	126.63	150.21	178.53	200.80	173.27	174.09	201.25	195.99	168.96
	1.1		136.90	177.93	154.64	161.88	175.41	156.91	167.19	158.53	198.69
	1.3		152.42	170.94	145.98	158.39	189.73	191.50	171.57	201.21	
	1.5			141.76	170.69	157.28	172.42	179.44	157.46		
	1.7					150.01	136.64	205.38			
	1.9					181.53	160.99				

Algorithm: VB-CEV, Test: SD-NNV

		px									
		0.1	0.3	0.5	0.7	0.9	1.1	1.3	1.5	1.7	1.9
py	0.1	173.33	165.32	173.33	177.46	195.43					
	0.3	133.58	155.03	157.05	177.60	173.43	164.56	174.49			
	0.5	153.09	162.46	136.28	158.89	168.05	172.99	167.94	178.59		
	0.7	159.74	155.01	157.55	151.41	165.92	147.95	186.48	171.40		
	0.9	137.94	111.21	136.19	153.92	141.67	165.15	145.54	148.85	170.08	168.57
	1.1		138.81	154.88	144.33	158.81	166.66	149.05	154.88	139.87	155.39
	1.3		122.24	121.18	147.53	138.37	137.67	146.96	154.69	150.00	
	1.5			131.96	100.99	125.88	145.99	127.63	152.84		
	1.7					142.72	116.98	148.30			
	1.9					119.18	119.78				

Table J.6: Value-Based Search Tests (MN-EV)**Algorithm: VB-CEV, Test: MN-EV**

		px									
		0.1	0.3	0.5	0.7	0.9	1.1	1.3	1.5	1.7	1.9
py	0.1	62.23	62.57	62.33	62.26	62.07					
	0.3	63.49	63.34	63.24	63.50	62.55	64.26	63.16			
	0.5	63.93	63.33	63.68	63.43	63.61	63.97	63.06	63.37		
	0.7	64.30	63.94	63.38	63.12	63.67	63.39	63.25	63.51		
	0.9	64.01	63.83	63.50	63.47	64.02	63.18	63.52	63.75	63.63	63.51
	1.1		63.62	63.66	63.21	63.69	63.28	63.40	63.58	63.27	63.44
	1.3		63.78	63.54	63.61	63.59	63.43	63.25	63.69	63.66	
	1.5			63.89	63.25	63.31	63.41	63.03	63.29		
	1.7					63.27	63.31	63.19			
	1.9					63.28	63.17				

Table J.7: Value-Based Search Tests (SD-EV)**Algorithm: VB-CEV, Test: SD-EV**

		px									
		0.1	0.3	0.5	0.7	0.9	1.1	1.3	1.5	1.7	1.9
py	0.1	4.34	4.28	3.80	3.56	3.47					
	0.3	4.08	3.41	3.16	2.88	2.64	3.53	3.10			
	0.5	2.89	2.69	2.64	2.77	2.49	2.54	2.12	2.47		
	0.7	2.89	3.06	2.97	2.20	3.00	1.85	2.17	2.24		
	0.9	2.15	2.39	2.48	2.23	2.42	1.77	2.42	2.30	2.32	2.40
	1.1		2.18	2.10	1.30	2.09	1.41	1.55	2.16	1.46	1.85
	1.3		1.49	2.01	1.68	1.59	1.93	1.89	1.92	2.18	
	1.5			2.25	1.72	1.55	1.78	1.21	1.19		
	1.7					1.46	1.33	1.18			
	1.9					1.34	0.64				

Table J.8: Value-Based Search Tests (MN-ITR)

Algorithm: VB-CEV, Test: MN-ITR

		px									
		0.1	0.3	0.5	0.7	0.9	1.1	1.3	1.5	1.7	1.9
py	0.1	35.21	35.06	34.05	35.01	34.80					
	0.3	21.84	21.39	21.64	21.85	21.48	21.36	21.62			
	0.5	18.29	18.29	18.32	18.39	18.31	18.25	18.54	18.55		
	0.7	17.39	16.74	16.81	16.91	17.02	17.27	16.74	17.12		
	0.9	16.34	16.19	16.15	16.26	16.23	16.23	16.38	16.29	16.23	16.10
	1.1		15.94	15.60	15.60	15.68	15.82	15.79	15.70	15.74	15.82
	1.3		15.47	15.47	15.44	15.51	15.50	15.29	15.50	15.46	
	1.5			15.18	15.24	15.19	15.24	15.15	15.28		
	1.7					15.09	15.07	15.06			
	1.9					14.99	14.97				

Table J.9: Value-Based Search Tests (SD-ITR)

Algorithm: VB-CEV, Test: SD-ITR

		px									
		0.1	0.3	0.5	0.7	0.9	1.1	1.3	1.5	1.7	1.9
py	0.1	4.31	3.91	4.52	3.87	4.54					
	0.3	1.92	1.97	2.01	2.00	2.03	2.36	1.59			
	0.5	1.44	1.28	1.21	1.29	1.54	1.43	1.42	1.40		
	0.7	1.00	1.26	0.93	1.14	1.16	1.14	1.18	0.97		
	0.9	0.87	1.07	0.98	0.88	0.87	1.11	0.91	1.02	1.11	1.21
	1.1		0.70	0.84	0.65	0.91	0.67	0.61	0.74	0.71	0.71
	1.3		0.55	0.66	0.67	0.62	0.76	0.78	0.73	0.78	
	1.5			0.52	0.66	0.96	0.53	0.51	0.48		
	1.7					0.53	0.65	0.49			
	1.9					0.46	0.32				

Appendix K

WIDE GRAPH (W2) EXAMPLE

K.1 Graph W2's Topology

- Time horizon: 15
- Number of nodes: 1315
- Total number of arcs: 8869
- Number of different paths: $\approx 1.73 \times 10^{13}$
 - Table (K.1) shows the distribution of nodes in the graph.
- The number of arcs from each node is a random integer from the set $\{2, \dots, 20\}$. (Terminal nodes have no arcs.)
- Arc lengths are drawn randomly from a uniform probability distribution on $[0,1]$. With probability $1/10$, the length of the arc is multiplied by 10. This provides a mechanism to place isolated large rewards in the graph.

K.2 Paths in Graph W2

- The myopic path through the graph is in Table (K.2).
 - Its value is 57.30.
- Table (K.3) contains the optimal path through this graph.
 - Its value is 81.75.

Table K.1: Graph W2's Topological Structure

Time Index	Node Indexes	Number of Nodes
0	0-0	1
1	1-2	2
2	3-5	3
3	6-10	5
4	11-14	4
5	15-24	10
6	25-35	11
7	36-56	21
8	57-90	34
9	91-122	32
10	123-188	66
11	189-254	66
12	255-379	125
13	380-587	208
14	588-812	225
15	813-1314	502

Table K.2: The Myopic Path through Graph W2

Time	Node	Decision	Value
0	0	1	57.301491
1	1	1	56.804649
2	4	1	55.922054
3	6	1	54.970058
4	13	1	54.277622
5	23	1	51.107971
6	30	1	50.467541
7	38	1	43.823051
8	65	1	41.778313
9	117	1	35.130772
10	160	1	25.262001
11	235	1	17.108982
12	327	1	11.331218
13	438	1	5.485092
14	810	1	0.978118
15	1259	-	0.000000

Table K.3: The Optimal Path through Graph W2

Time	Node	Decision	Value
0	0	2	81.753738
1	2	1	81.487068
2	4	5	78.235016
3	8	1	78.219543
4	11	2	71.409035
5	18	2	66.938683
6	34	1	61.387062
7	41	1	52.262943
8	66	19	44.351631
9	117	1	44.307381
10	160	2	34.438610
11	251	1	29.644154
12	285	3	21.081575
13	525	2	17.572252
14	602	1	9.983520
15	820	-	0.000000

Appendix L

WIDE GRAPH (W2) COMPUTATIONAL EXPERIMENTS

- 57.30 is the length of the myopic path through the graph.
- 81.75 is the length of the longest path through the graph.

Table L.1: Monte Carlo Search Tests

Algorithm	MN-IC	SD-IC	MN-NNV	SD-NNV
MC-EV	622.46	251.50	267.30	109.77
MC-TEV	364.05	158.88	416.36	128.30
MC-CEV	223.45	118.48	580.88	158.34
Algorithm	MN-EV	SD-EV	MN-ITR	SD-ITR
MC-CEV	52.20	5.50	13.26	0.52

Table L.2: Value-Based Search Tests (MN-IC)

Algorithm: VB-EV, Test: MN-IC

		px									
		0.1	0.3	0.5	0.7	0.9	1.1	1.3	1.5	1.7	1.9
py	0.1	826.26	434.34	426.85	435.11	401.70					
	0.3	560.65	364.26	296.84	324.46	344.94	330.38	271.85			
	0.5	564.10	280.21	262.73	255.47	261.98	250.74	266.34	251.53		
	0.7	510.90	324.07	268.02	294.89	233.34	244.96	265.98	271.55		
	0.9	509.69	279.55	270.01	263.70	262.26	249.97	266.55	254.81	235.50	227.86
	1.1		276.00	288.26	278.56	232.88	244.71	226.35	256.91	234.01	251.61
	1.3		286.41	240.43	248.68	268.19	246.07	232.79	253.50	227.62	
	1.5			247.12	224.06	279.84	235.29	259.07	230.38		
	1.7					243.47	253.32	265.54			
	1.9					263.71	240.10				

Algorithm: VB-TEV, Test: MN-IC

		px									
		0.1	0.3	0.5	0.7	0.9	1.1	1.3	1.5	1.7	1.9
py	0.1	157.18	200.22	211.80	238.50	221.04					
	0.3	105.11	103.54	132.78	163.97	181.36	167.88	176.69			
	0.5	88.90	85.51	120.47	141.10	154.93	153.29	147.64	150.91		
	0.7	79.69	91.67	105.21	107.64	127.51	119.11	130.65	162.70		
	0.9	149.62	109.50	121.08	110.22	118.46	127.89	135.38	125.94	130.56	134.30
	1.1		103.56	111.49	111.46	125.38	111.46	133.64	130.66	117.42	137.53
	1.3		106.34	105.12	110.49	117.58	123.21	139.97	123.41	120.56	
	1.5			106.97	108.97	108.19	109.86	119.66	135.54		
	1.7					118.25	121.61	124.66			
	1.9					113.60	118.83				

Algorithm: VB-CEV, Test: MN-IC

		px									
		0.1	0.3	0.5	0.7	0.9	1.1	1.3	1.5	1.7	1.9
py	0.1	97.24	105.54	111.62	134.30	134.75					
	0.3	52.31	59.74	57.88	72.97	64.41	73.34	77.84			
	0.5	43.55	47.41	48.67	51.67	57.81	54.79	59.60	61.65		
	0.7	40.02	41.09	40.81	45.25	42.12	48.49	45.46	48.86		
	0.9	36.35	37.59	47.55	44.15	43.45	43.84	46.50	47.24	43.71	46.91
	1.1		39.44	43.58	42.86	40.27	41.56	43.44	41.98	43.31	44.23
	1.3		39.27	38.89	41.15	40.83	44.60	43.25	44.95	43.46	
	1.5			43.09	40.65	40.56	41.66	41.33	39.36		
	1.7					42.85	39.23	40.25			
	1.9					39.17	41.12				

Table L.3: Value-Based Search Tests (SD-IC)

Algorithm: VB-EV, Test: SD-IC

		px									
		0.1	0.3	0.5	0.7	0.9	1.1	1.3	1.5	1.7	1.9
py	0.1	1420.85	379.08	425.72	232.28	195.92					
	0.3	964.51	243.74	138.53	181.34	252.51	233.23	148.09			
	0.5	921.55	207.53	135.81	146.18	125.55	119.45	132.91	107.94		
	0.7	797.07	184.53	136.02	158.79	100.42	133.39	170.39	173.51		
	0.9	748.68	152.77	136.31	153.19	154.24	100.26	147.26	106.61	116.58	87.86
	1.1		130.50	117.30	124.29	86.92	104.11	84.34	115.73	105.99	139.84
	1.3		175.47	99.15	110.14	123.30	102.50	98.95	125.46	105.45	
	1.5			102.77	81.72	134.49	109.11	134.84	113.29		
	1.7					145.11	112.31	130.35			
	1.9					110.38	106.23				

Algorithm: VB-TEV, Test: SD-IC

		px									
		0.1	0.3	0.5	0.7	0.9	1.1	1.3	1.5	1.7	1.9
py	0.1	126.62	181.74	207.90	239.71	143.73					
	0.3	163.13	70.80	154.14	141.06	107.64	94.55	135.26			
	0.5	137.66	66.38	80.32	155.15	109.74	109.99	102.39	89.49		
	0.7	56.84	71.70	72.28	76.41	75.96	67.88	68.83	117.77		
	0.9	156.64	81.40	99.94	82.10	91.13	90.38	88.79	71.55	73.48	77.57
	1.1		81.23	66.99	59.58	64.35	61.80	87.48	83.02	68.48	81.47
	1.3		58.08	59.81	49.26	53.88	70.62	82.48	72.23	75.61	
	1.5			47.19	48.90	51.91	54.12	66.10	93.77		
	1.7					71.32	66.46	70.38			
	1.9					75.58	57.48				

Algorithm: VB-CEV, Test: SD-IC

		px									
		0.1	0.3	0.5	0.7	0.9	1.1	1.3	1.5	1.7	1.9
py	0.1	39.50	63.46	59.65	82.06	94.73					
	0.3	15.08	36.93	26.01	41.87	38.57	39.65	48.95			
	0.5	15.25	19.06	22.92	25.41	32.34	32.38	29.00	32.87		
	0.7	14.37	14.81	18.55	21.19	18.99	25.49	21.59	24.92		
	0.9	13.42	13.20	22.33	22.31	21.68	17.98	19.87	20.77	18.99	19.88
	1.1		14.56	17.90	20.05	15.72	15.45	21.61	19.11	17.47	21.92
	1.3		17.43	16.45	17.37	14.85	19.83	18.04	20.39	21.13	
	1.5			16.12	16.02	15.47	16.86	15.52	13.31		
	1.7					15.61	13.43	14.18			
	1.9					14.51	18.04				

Table L.4: Value-Based Search Tests (MN-NNV)

Algorithm: VB-EV, Test: MN-NNV

		px									
		0.1	0.3	0.5	0.7	0.9	1.1	1.3	1.5	1.7	1.9
py	0.1	769.20	755.84	727.69	686.70	699.94					
	0.3	720.74	613.21	635.31	602.80	600.86	601.39	647.77			
	0.5	666.50	628.73	601.44	602.36	587.74	597.80	574.73	591.64		
	0.7	609.30	500.91	539.24	499.09	574.96	563.65	543.59	540.36		
	0.9	580.39	512.69	506.12	515.16	508.52	510.12	503.95	503.76	538.75	541.42
	1.1		473.96	446.12	451.09	510.99	498.88	520.95	479.20	518.98	498.51
	1.3		467.99	489.18	478.07	449.60	476.55	500.12	473.80	511.11	
	1.5			465.06	494.09	419.80	487.43	457.29	494.73		
	1.7					472.98	441.35	435.84			
	1.9					411.56	455.70				

Algorithm: VB-TEV, Test: MN-NNV

		px									
		0.1	0.3	0.5	0.7	0.9	1.1	1.3	1.5	1.7	1.9
py	0.1	988.25	929.70	905.00	870.83	871.80					
	0.3	966.33	934.46	898.48	832.86	788.95	797.95	796.50			
	0.5	956.62	931.33	844.27	830.52	780.24	768.99	782.35	771.02		
	0.7	928.45	887.24	851.76	846.83	787.16	804.76	775.00	718.58		
	0.9	789.61	824.06	802.12	816.27	800.60	771.29	742.51	764.35	750.70	742.26
	1.1		824.74	787.16	784.69	742.77	784.89	737.02	739.26	771.75	720.84
	1.3		787.36	791.21	764.91	745.71	737.39	698.08	743.00	752.85	
	1.5			764.92	760.38	765.74	761.17	735.92	710.16		
	1.7					735.60	721.17	714.00			
	1.9					747.35	716.92				

Algorithm: VB-CEV, Test: MN-NNV

		px									
		0.1	0.3	0.5	0.7	0.9	1.1	1.3	1.5	1.7	1.9
py	0.1	1051.65	1029.97	1011.80	984.25	982.94					
	0.3	1051.06	1033.01	1033.45	993.99	1017.74	990.65	980.96			
	0.5	1050.49	1033.36	1030.25	1017.88	999.79	1009.59	992.31	985.95		
	0.7	1042.99	1037.95	1042.45	1022.85	1036.14	1011.23	1023.04	1010.83		
	0.9	1051.33	1043.30	1002.44	1016.98	1019.96	1015.34	1005.25	1002.21	1017.29	1002.56
	1.1		1026.21	1008.49	1013.17	1023.06	1017.40	1009.92	1017.39	1008.76	1007.67
	1.3		1024.89	1024.25	1013.70	1013.62	998.54	1002.95	996.84	1005.66	
	1.5			997.14	1010.55	1010.21	1006.69	1005.99	1015.99		
	1.7					995.59	1012.58	1008.06			
	1.9					1011.25	1004.49				

Table L.5: Value-Based Search Tests (SD-NNV)

Algorithm: VB-EV, Test: SD-NNV

		px									
		0.1	0.3	0.5	0.7	0.9	1.1	1.3	1.5	1.7	1.9
py	0.1	316.54	259.40	190.16	180.06	146.71					
	0.3	333.76	255.93	179.38	181.18	211.36	169.11	153.20			
	0.5	345.53	269.57	194.27	171.61	172.13	153.71	131.06	143.40		
	0.7	324.18	234.23	178.75	153.36	158.21	155.55	167.36	185.20		
	0.9	327.51	218.66	201.57	189.67	158.34	139.14	172.79	142.54	154.92	148.51
	1.1		176.20	179.83	149.68	140.98	161.26	147.08	149.93	168.87	166.56
	1.3		225.55	158.57	166.48	164.42	160.79	176.45	173.07	169.17	
	1.5			159.33	143.69	161.19	179.50	188.38	174.42		
	1.7					172.37	161.30	199.10			
	1.9					139.58	169.80				

Algorithm: VB-TEV, Test: SD-NNV

		px									
		0.1	0.3	0.5	0.7	0.9	1.1	1.3	1.5	1.7	1.9
py	0.1	136.64	173.19	166.20	165.50	149.44					
	0.3	154.13	141.47	189.78	194.88	182.09	160.67	186.80			
	0.5	165.02	151.86	172.92	217.99	207.01	170.99	180.57	170.08		
	0.7	154.72	166.97	186.38	194.66	186.66	164.37	172.29	204.46		
	0.9	238.45	199.33	217.10	195.10	211.11	205.93	173.38	192.32	184.72	177.66
	1.1		193.30	171.43	177.59	174.59	184.72	210.19	199.26	199.97	195.76
	1.3		167.44	175.61	155.86	167.15	177.83	187.30	196.59	203.58	
	1.5			148.03	156.36	176.41	174.77	179.97	210.28		
	1.7					184.55	173.65	178.38			
	1.9					195.17	160.06				

Algorithm: VB-CEV, Test: SD-NNV

		px									
		0.1	0.3	0.5	0.7	0.9	1.1	1.3	1.5	1.7	1.9
py	0.1	61.05	95.02	92.40	116.39	127.23					
	0.3	49.65	93.10	76.21	105.15	99.26	100.73	105.26			
	0.5	57.29	68.33	84.79	87.16	102.31	103.59	95.59	105.81		
	0.7	59.48	65.69	75.87	84.30	81.80	96.33	85.60	96.08		
	0.9	62.95	63.28	91.58	93.34	90.76	81.56	85.45	85.86	81.07	87.25
	1.1		67.31	78.67	85.09	74.92	72.66	94.46	86.20	79.70	98.83
	1.3		81.81	78.26	82.69	72.30	89.50	81.28	90.80	95.11	
	1.5			74.75	76.94	73.89	83.21	74.76	65.54		
	1.7					75.56	66.73	70.20			
	1.9					72.04	87.12				

Table L.6: Value-Based Search Tests (MN-EV)

Algorithm: VB-CEV, Test: MN-EV

		px									
		0.1	0.3	0.5	0.7	0.9	1.1	1.3	1.5	1.7	1.9
py	0.1	73.11	73.76	73.77	73.34	72.69					
	0.3	72.70	73.60	72.69	73.31	73.81	73.61	74.01			
	0.5	72.13	72.72	73.17	72.91	73.22	74.15	73.88	73.19		
	0.7	72.57	73.17	72.99	73.11	73.62	73.97	73.35	73.89		
	0.9	72.32	72.57	73.15	73.08	72.89	73.64	73.61	73.77	72.91	73.63
	1.1		72.65	73.10	72.90	73.34	73.37	73.11	73.63	72.47	73.22
	1.3		71.57	73.07	73.06	72.94	72.78	72.61	72.99	73.32	
	1.5			72.17	72.56	72.51	72.57	72.92	72.99		
	1.7					72.79	73.17	72.97			
	1.9					72.41	72.72				

Table L.7: Value-Based Search Tests (SD-EV)

Algorithm: VB-CEV, Test: SD-EV

		px									
		0.1	0.3	0.5	0.7	0.9	1.1	1.3	1.5	1.7	1.9
py	0.1	3.31	3.45	4.00	3.15	3.30					
	0.3	2.86	3.39	2.95	3.54	3.26	3.08	3.28			
	0.5	2.76	3.54	3.38	3.01	3.40	3.32	3.41	3.09		
	0.7	2.89	3.30	2.94	3.50	3.07	3.69	3.24	3.40		
	0.9	2.71	3.12	3.07	3.33	2.86	3.34	3.27	3.27	2.97	2.94
	1.1		2.96	3.08	3.25	3.19	3.00	3.00	3.22	2.79	3.05
	1.3		2.43	3.32	3.20	3.06	2.65	2.87	2.72	3.35	
	1.5			2.92	2.98	2.99	2.98	2.92	3.06		
	1.7					2.92	3.11	2.86			
	1.9					2.90	2.67				

Table L.8: Value-Based Search Tests (MN-ITR)**Algorithm: VB-CEV, Test: MN-ITR**

		px									
		0.1	0.3	0.5	0.7	0.9	1.1	1.3	1.5	1.7	1.9
py	0.1	33.04	30.79	30.43	30.34	30.05					
	0.3	19.09	19.23	18.59	18.70	18.59	18.85	18.88			
	0.5	16.00	16.29	16.09	15.89	15.90	15.80	15.94	15.79		
	0.7	15.03	14.76	15.05	14.78	14.86	14.69	14.71	14.79		
	0.9	14.24	14.25	14.10	14.09	14.22	14.10	14.07	14.11	14.05	13.90
	1.1		13.93	13.68	13.80	13.68	13.85	13.64	13.55	13.64	13.62
	1.3		13.41	13.46	13.38	13.38	13.31	13.44	13.36	13.47	
	1.5			13.22	13.18	13.21	13.18	13.14	13.12		
	1.7					13.05	13.05	13.11			
	1.9					12.96	12.99				

Table L.9: Value-Based Search Tests (SD-ITR)**Algorithm: VB-CEV, Test: SD-ITR**

		px									
		0.1	0.3	0.5	0.7	0.9	1.1	1.3	1.5	1.7	1.9
py	0.1	4.89	3.42	3.71	3.79	3.85					
	0.3	2.18	1.83	1.53	1.79	1.93	1.68	1.79			
	0.5	1.21	1.27	1.42	1.21	1.32	1.18	1.07	1.25		
	0.7	1.12	0.80	1.02	0.91	0.99	0.98	0.77	0.85		
	0.9	0.68	0.74	0.79	0.78	0.62	0.85	0.76	0.84	0.69	0.72
	1.1		0.79	0.63	0.64	0.63	0.66	0.58	0.59	0.60	0.60
	1.3		0.63	0.57	0.49	0.51	0.52	0.50	0.58	0.59	
	1.5			0.50	0.50	0.52	0.41	0.44	0.40		
	1.7					0.35	0.39	0.50			
	1.9					0.34	0.44				

Appendix M

WIDE GRAPH (W3) EXAMPLE

M.1 Graph W3's Topology

- Time horizon: 20
- Number of nodes: 26140
- Total number of arcs: 99315
- Number of different paths: $\approx 7.37 \times 10^{13}$
 - Table (M.1) shows the distribution of nodes in the graph.
- The number of arcs from each node is a random integer from the set $\{2, \dots, 20\}$.
(Terminal nodes have no arcs.)
- Arc lengths are drawn randomly from a uniform probability distribution on $[0,1]$. With probability $1/10$, the length of the arc is multiplied by 10. This provides a mechanism to place isolated large rewards in the graph.

M.2 Paths in Graph W3

- The myopic path through the graph is in Table (M.2).
 - Its value is 59.35.
- Table (M.3) contains the optimal path through this graph.
 - Its value is 94.30.

Table M.1: Graph W3's Topological Structure

Time Index	Node Indexes	Number of Nodes
0	0-0	1
1	1-1	1
2	2-4	3
3	5-7	3
4	8-15	8
5	16-22	7
6	23-32	10
7	33-66	34
8	67-113	47
9	114-176	63
10	177-304	128
11	305-441	137
12	442-682	241
13	683-1131	449
14	1132-1941	810
15	1942-3046	1105
16	3047-4721	1675
17	4722-7239	2518
18	7240-11643	4404
19	11644-16512	4869
20	16513-26139	9627

Table M.2: The Myopic Path through Graph W3

Time	Node	Decision	Value
0	0	1	59.347881
1	1	1	58.684532
2	4	1	57.908386
3	5	1	56.959167
4	10	1	55.998535
5	20	1	50.012024
6	23	1	49.126682
7	41	1	48.279854
8	75	1	40.906582
9	140	1	39.909054
10	270	1	32.639545
11	387	1	31.724142
12	622	1	30.929867
13	835	1	22.353556
14	1901	1	21.548264
15	2658	1	20.911556
16	4040	1	20.022156
17	5933	1	19.076235
18	7615	1	9.323435
19	11866	1	0.949767
20	22421	-	0.000000

Table M.3: The Optimal Path through Graph W3

Time	Node	Decision	Value
0	0	1	94.296928
1	1	1	93.633575
2	4	1	92.857430
3	5	1	91.908211
4	10	1	90.947578
5	20	8	84.961067
6	30	1	84.507866
7	45	6	75.042244
8	98	1	74.887207
9	154	2	66.535515
10	274	1	65.651604
11	355	1	55.808163
12	540	1	45.957703
13	712	2	44.971802
14	1612	1	36.187660
15	2233	2	27.977264
16	3619	2	27.008881
17	6714	1	19.353312
18	8316	1	18.290352
19	15751	1	9.572130
20	24847	-	0.000000

Appendix N

WIDE GRAPH (W3) COMPUTATIONAL EXPERIMENTS

- 59.35 is the length of the myopic path through the graph.
- 94.30 is the length of the longest path through the graph.

Table N.1: Monte Carlo Search Tests

Algorithm	MN-EV	SD-EV	MN-ITR	SD-ITR
MC-CEV	75.56	3.97	305.19	3.17

Table N.2: Value-Based Search Tests (MN-EV)**Algorithm: VB-CEV, Test: MN-EV**

		px									
		0.1	0.3	0.5	0.7	0.9	1.1	1.3	1.5	1.7	1.9
0.1		88.85	88.11	87.83	88.18	87.89					
0.3		89.07	88.05	88.13	88.30	88.20	87.86	87.90			
0.5		89.13	88.36	88.09	88.11	87.71	87.99	88.31	87.75		
0.7		89.13	88.27	87.63	88.13	88.21	88.15	88.02	87.89		
0.9		88.91	88.27	88.16	87.88	88.01	87.83	88.06	88.11	87.99	87.98
1.1			88.13	88.14	88.09	88.28	87.99	87.86	88.47	87.78	88.51
1.3			88.18	88.10	88.23	87.94	88.22	88.26	87.93	87.89	
1.5				88.08	88.14	88.30	88.00	88.17	88.06		
1.7						87.88	88.06	88.50			
1.9						87.95	87.88				

Table N.3: Value-Based Search Tests (SD-EV)**Algorithm: VB-CEV, Test: SD-EV**

		px									
		0.1	0.3	0.5	0.7	0.9	1.1	1.3	1.5	1.7	1.9
0.1		0.93	1.31	1.10	1.51	1.38					
0.3		1.45	0.96	1.37	1.62	1.67	1.26	1.31			
0.5		1.50	1.64	1.20	1.41	0.89	1.22	1.58	1.05		
0.7		1.50	1.42	0.90	1.36	1.38	1.42	1.27	1.26		
0.9		1.33	1.43	1.41	1.13	1.47	1.08	1.48	1.51	1.34	1.38
1.1			1.24	1.38	1.41	1.73	1.34	1.13	1.70	1.14	1.90
1.3			1.14	1.31	1.45	1.20	1.51	1.70	1.15	1.11	
1.5				1.25	1.45	1.56	1.32	1.41	1.29		
1.7						1.32	1.40	1.91			
1.9						1.20	1.16				

Table N.4: Value-Based Search Tests (MN-ITR)

Algorithm: VB-CEV, Test: MN-ITR

		px									
		0.1	0.3	0.5	0.7	0.9	1.1	1.3	1.5	1.7	1.9
	0.1	783.94	696.81	687.84	679.77	681.25					
	0.3	470.25	432.77	421.68	418.73	416.26	420.09	419.35			
	0.5	395.36	364.11	360.51	357.74	357.90	356.43	354.34	355.24		
	0.7	358.50	334.54	331.02	329.05	328.15	327.64	327.79	327.40		
	0.9	340.66	316.90	314.57	313.39	311.93	311.73	311.99	311.23	311.31	309.68
	1.1		307.80	303.76	303.68	299.82	300.81	301.84	300.99	300.71	299.21
	1.3		300.93	297.02	296.09	295.68	293.36	292.89	294.75	294.39	
	1.5			292.71	291.24	290.45	288.41	290.34	289.75		
	1.7					286.29	286.84	283.01			
	1.9					283.71	283.52				

Table N.5: Value-Based Search Tests (SD-ITR)

Algorithm: VB-CEV, Test: SD-ITR

		px									
		0.1	0.3	0.5	0.7	0.9	1.1	1.3	1.5	1.7	1.9
	0.1	21.65	26.87	22.11	29.88	17.90					
	0.3	20.53	10.25	18.37	23.72	30.57	18.63	9.72			
	0.5	8.06	12.41	6.31	6.82	5.11	5.49	15.53	5.92		
	0.7	17.23	4.47	5.72	3.92	4.64	4.67	3.90	5.33		
	0.9	4.93	14.02	4.34	3.26	10.30	3.58	3.56	4.25	3.78	7.79
	1.1		3.42	2.81	3.20	14.05	7.31	2.97	8.24	4.65	11.90
	1.3		2.72	6.18	2.66	2.39	13.54	13.89	2.32	2.49	
	1.5			2.39	2.11	3.10	16.26	2.06	2.17		
	1.7					7.32	3.30	15.66			
	1.9					6.16	3.38				