NOTE TO USERS

The original manuscript received by UMI contains pages with slanted, broken and indistinct print. Pages were microfilmed as received.

This reproduction is the best copy available

UMI
PARTIALLY OBSERVED MARKOVIAN DECISION PROCESSES WITH APPLICATIONS

by

Adelita Posada-Bolivar

A Thesis Submitted in Conformity with the Requirements for the Degree of Doctor of Philosophy
Graduate Department of Mechanical and Industrial Engineering
University of Toronto

© Copyright by Adelita Posada-Bolivar, 1998
The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author’s permission.

L’auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

L’auteur conserve la propriété du droit d’auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

0-612-35285-4
Abstract

This dissertation is concerned with new solution techniques for the finite-horizon partially observed Markov decision process (POMDP), and the finite-horizon partially observed semi-Markov decision process (POSMDP). These two problems tend toward computational intractability. In fact, computational results seem to be confined to very small problems. An article by Papadimitriou and Tsitsiklis [31](1987) explores the computational implications of POMDP, and report that it is PSPACE-complete. A PSPACE-complete problem is "...even less likely to be solved in polynomial time than the NP-complete problems" [31] (1987). Solution procedures generally fall into two categories: exact solution procedures and approximation procedures. The proposed research is in the second category. We propose heuristic methods to determine good approximate solutions for the POMDP and POSMDP.

An exploratory investigation of the performance of the following solution procedures for the POMDP is provided: One-Pass algorithm, Relaxed Region algorithm, Linear Support algorithm, S. Mukherjee and K. Seth’s algorithm, and our Proposed algorithm. This investigation shows the interrelations between these algorithms and the different development paths that have been taken in formulating them. In addition, a numerical comparison is performed. This comparison is based on two types of numerical evaluations. The first performance criterion is CPU runtime used during the operation of each solution procedure. We perform this evaluation because the main concern of our research is improving the speed of solution procedures for the POMDP. The second type of evaluation involves the determination of the differences in terms of the Total Expected Reward between these algorithms.

Test problems were separated into four basic groups:
1. The first group contains several sets of data without any special assumption about problem structure.

2. The second group consists of "close-to-perfect information" POMDPs.

3. The third group considers the special case of the completely unobserved Markov decision process (CUMDP).

4. The fourth group considers the completely observed Markov decision process.

Although the development of the heuristics is the essential theme of this dissertation, a secondary but integral part of the work is the development of a mathematical optimization control for roll surface degradation in steel processing. This problem has been deliberately chosen to show the application of the model and heuristic methods described by this work to large scale problems. Comparisons with the actual control for roll surface degradation indicates that the proposed heuristic applied to this problem can produce better results than the ones obtained with the current method.
ACKNOWLEDGEMENTS

I wish to thank my advisor, Prof. Erick E. Pickett, for his guidance and help in this enterprise. I also wish to thank Prof. J.G.C. Templeton, Prof. D.E. Lane, Prof. M.J.M. Posner, and Prof. B.W. Karney for their kindness in acting as members of my committee.

I am indebted to Professor Michel Gendreau for guiding this research. His advice, interest and enthusiasm have been invaluable resources and I am deeply grateful. I am also indebted to Professor M.W. Carter for his insightful comments and feedback, and for his continuing friendship. His direction has proved invaluable to me time and time again.

I thank Dofasco Inc. in Hamilton, Ontario for allowing me to work at the Finishing Mill. Special thanks to Ron Webber, his own passion to understand has been a motivating factor behind this thesis.

I would also like to thank my husband, Leo, and my two sons Daniel and Alejandro, for their love, patience, and constant confidence throughout the course of my doctoral studies. It is tempting, and perhaps appropriate, to include Leo’s name as co-author in recognition of his constant moral support.

My deepest appreciation, though, must be reserved for my father, God, for his love and companion. You are the best (Gracias Dios, Muchas Gracias).

This research was partially supported by the Universidad de las Americas-Puebla, Mexico, and the National Bureau of Science and Technology of Mexico (CONACYT). Their support is gratefully acknowledged.
CONTENTS

4.2 THE MONAHAN ALGORITHM ........................................... 55
4.3 THE RELAXED REGION ALGORITHM ................................. 56
4.4 MUKHERJEE AND SETH'S ALGORITHM .............................. 59
4.5 $MPAO^*$ ALGORITHM .................................................. 61
4.6 LINEAR SUPPORT ALGORITHM ....................................... 66

5 POMDP PROPOSED HEURISTIC ....................................... 70
  5.1 PROBLEM DEFINITION .............................................. 74
  5.2 DESCRIPTION OF THE HEURISTIC ................................. 79
  5.3 BOUNDS ON THE SOLUTION ......................................... 85
    5.3.1 PRECISE INFORMATION ABOUT THE STATE OF THE CORE PROCESS .................................................. 85
    5.3.2 NO INFORMATION ABOUT THE STATE OF THE CORE PROCESS .................................................. 87

6 EVALUATION OF SOME POMDP SOLUTION TECHNIQUES ............. 89
  6.1 CLASSIFICATION OF THE TEST PROBLEMS ......................... 92
    6.1.1 THE GENERAL POMDP - GROUP 1 ............................. 93
    6.1.2 THE CLOSE-TO-PERFECT INFORMATION PROBLEM - GROUP 2 .................................................. 96
    6.1.3 THE CUMDP PROBLEMS - GROUP 3 ............................ 97
    6.1.4 THE PERFECT INFORMATION PROBLEM - GROUP 4  ........ 99
  6.2 INTERPRETATION OF THE RESULTS ................................ 103

7 ROLL SURFACE CONTROL AT HOT STRIP MILL ..................... 108
  7.1 PROCESS DESCRIPTION ............................................. 109
  7.2 STEEL METALLURGY .............................................. 114
  7.3 CURRENT HOT STRIP MILL OPERATING POLICY .................... 115
  7.4 ACTUAL CONTROL OF THE IMPORTANT FACTORS WHEN ROLLING .................................................. 119
## CONTENTS

7.5 PROBLEM DESCRIPTION ............................................................................ 122
7.6 FORMULATION OF THE HOT STRIP MILL CONTROL PROBLEM AS A POMDP ........................................... 124
7.7 MODEL INPUTS AND OUTPUTS ................................................................. 137
    7.7.1 MODEL INPUTS ................................................................................ 137
    7.7.2 MODEL OUTPUTS ............................................................................ 140

8 A SIMULATOR FOR THE HOT STRIP MILL ............................................... 142
    8.1 CUMULATIVE DAMAGE MODEL FOR THE FINISHING ROLL SURFACE TO BE USED IN THE SIMULATION ................................. 146
        8.1.1 TEMPERATURE DIFFERENCE $\Delta T_i$ DURING ONE REVOLU-
                TION ....................................................................................... 151
        8.1.2 FORMULATION OF THE THERMAL MODEL ............................. 151
        8.1.3 NUMERICAL SOLUTION AND VALIDATION OF THE THER-
                MAL MODEL ........................................................................ 159
        8.1.4 DETERMINATION OF HEAT PENETRATION DEPTH $\overline{W}_i$ ....... 160
        8.1.5 CORRELATION BETWEEN ROLL SURFACE BREAK-
                DOWN AND THERMAL FATIGUE DAMAGE ................................. 160
        8.1.6 PROBABILITY DISTRIBUTIONS FOR THE MODEL ............... 162
    8.2 THE SIMULATOR PROGRAM .................................................................. 164

9 ANALYSIS OF RESULTS ............................................................................ 169
    9.1 EVALUATION OF THE PERFORMANCE OF OUR PROPOSED HEURISTIC ALGORITHM WITH THE MILL PROBLEM ......................... 170
    9.2 DETERMINATION OF THE SIZE OF THE PRODUCT BLOCK ............ 175
    9.3 DETERMINATION OF THE FREQUENCY OF SAMPLING .................. 176
    9.4 EVALUATION OF THE CURRENT HOT STRIP MILL OPERATING PROCEDURE .................................................. 181
CONTENTS

10 TOPICS FOR FUTURE RESEARCH

10.1 POSMDP PROPOSED HEURISTIC ALGORITHM ............. 184
  10.1.1 PROBLEM DEFINITION ................................ 184
  10.1.2 POSMDP PROPERTIES ................................. 196
  10.1.3 COMPUTATIONAL EXPRESSIONS FOR $C'(\pi(t))$  .... 196
  10.1.4 POSMDP SOLUTION TECHNIQUES ........................ 198

10.2 OTHER RECOMMENDATIONS FOR FUTURE RESEARCH ...... 205

11 CONTRIBUTIONS AND CONCLUSIONS ..................... 207

12 REFERENCES ........................................... 210

A TEST DATA FOR FINITE HORIZON ALGORITHMS ............. 218

B CALCULATION OF THE EXPECTED REWARD .................... 258

C OBSERVED SEQUENCES OF ROLLS’DETERIORATION ............ 263
List of Figures

3.1 DECISION DIAGRAM FOR COMPLETELY OBSERVABLE MARKOV DECISION PROCESS .................. 30
3.2 PARTIALLY OBSERVABLE MARKOV DECISION PROCESS SHOWING SEPARATION INTO AN ACTUATOR AND AN ESTIMATOR ................................. 41
3.3 PRESERVATION OF STRAIGHT LINES ...................... 43

5.1 THE ESTIMATOR PART (SCHEMATIC) .................. 86
7.1 THE STEEL PRODUCTION PROCESS ..................... 110
7.2 CHANGES IN GRAIN STRUCTURE ......................... 116
7.3 COFFIN SHAPED PROFILE .............................. 118
7.4 QUALITY CLASSIFICATION OF STRIP SURFACE .... 127
7.5 QUALITY CLASSIFICATION OF ROLL SURFACE DE- TERIORATION ........................................... 128
7.6 ROLL WEAR PROFILE AND GRINDING STAGES ....... 132

8.1 ROLL GAP ................................................. 148
8.2 FLOW OF HEAT INTO A WORK ROLL .................. 152
8.3 HEAT LOSSES FROM A WORK ROLL .................. 153
8.4 COOLING OF MILL ROLLS .............................. 155
LIST OF FIGURES

8.5 SOLID ELEMENT DEFINED BY CYLINDRICAL COORDINATES ........................................ 157
8.6 WORK ROLLS SURFACE TEMPERATURE .......................................................... 161
8.7 SURFACE RATING OF THE ROLLS vs THERMAL FATIGUE DAMAGE ................................ 163
8.8 CUMULATIVE PRODUCTION vs CUMULATIVE THERMAL FATIGUE ................................. 165
## List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.1</td>
<td>The General POMDP - Data Sets 1-3</td>
<td>94</td>
</tr>
<tr>
<td>6.2</td>
<td>The General POMDP - Data Sets 4-6</td>
<td>95</td>
</tr>
<tr>
<td>6.3</td>
<td>Performance of PHA- General POMDPs</td>
<td>96</td>
</tr>
<tr>
<td>6.4</td>
<td>Close to Perfect information POMDP - Data Sets 7-10</td>
<td>98</td>
</tr>
<tr>
<td>6.5</td>
<td>Performance of PHA- “close to perfect information” POMDPs</td>
<td>99</td>
</tr>
<tr>
<td>6.6</td>
<td>The CUMDP - Data Sets 11-14</td>
<td>100</td>
</tr>
<tr>
<td>6.7</td>
<td>Performance of PHA- “CUMDP”</td>
<td>101</td>
</tr>
<tr>
<td>6.8</td>
<td>The Perfect Information Problem - Data Sets 15-18</td>
<td>102</td>
</tr>
<tr>
<td>6.9</td>
<td>Performance of PHA- perfect information POMDPs</td>
<td>103</td>
</tr>
<tr>
<td>6.10</td>
<td>Performance of PHA- hypotheses testing</td>
<td>104</td>
</tr>
<tr>
<td>7.1</td>
<td>Range of slab Dimensions</td>
<td>111</td>
</tr>
<tr>
<td>8.1</td>
<td>Correlation between Roll Surface Breakdown and Thermal Fatigue Damage</td>
<td>162</td>
</tr>
<tr>
<td>8.2</td>
<td>Conditional Distribution Function of roll surface deterioration given the Thermal Fatigue</td>
<td>164</td>
</tr>
<tr>
<td>9.1</td>
<td>Results using Simulation</td>
<td>171</td>
</tr>
<tr>
<td>9.2</td>
<td>Results using Proposed Heuristic Algorithm</td>
<td>171</td>
</tr>
<tr>
<td>9.3</td>
<td>Sensitivity Analysis - Inspection Cost of the Rolls</td>
<td>173</td>
</tr>
<tr>
<td>Table</td>
<td>Title</td>
<td>Page</td>
</tr>
<tr>
<td>-------</td>
<td>----------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>9.4</td>
<td>Sensitivity Analysis - Replacement Cost of the Rolls</td>
<td>174</td>
</tr>
<tr>
<td>9.5</td>
<td>Sensitivity Analysis - Roll Cost</td>
<td>174</td>
</tr>
<tr>
<td>9.6</td>
<td>PB optimal size</td>
<td>176</td>
</tr>
<tr>
<td>9.7</td>
<td>$t_{\text{min}}$ Data - Minimum Accepted Quality 3OF</td>
<td>178</td>
</tr>
<tr>
<td>9.8</td>
<td>$t_{\text{min}}$ Data - Minimum Accepted Quality 3CF</td>
<td>178</td>
</tr>
<tr>
<td>9.9</td>
<td>$t_{\text{min}}$ Data - Minimum Accepted Quality 3W</td>
<td>178</td>
</tr>
<tr>
<td>9.10</td>
<td>$t_{\text{min}}$ best fits</td>
<td>179</td>
</tr>
<tr>
<td>9.11</td>
<td>$t_{\text{min}}$ Chi-Square Test</td>
<td>179</td>
</tr>
<tr>
<td>9.12</td>
<td>Comparison of performance between proposed and current Sampling Plans</td>
<td>181</td>
</tr>
<tr>
<td>9.13</td>
<td>Comparison of performance between proposed and current control procedure</td>
<td>182</td>
</tr>
</tbody>
</table>
Chapter 1

INTRODUCTION

This dissertation will develop heuristic algorithms which apply not only to the partially observable Markov decision process (POMDP), but also to the partially observable semi-Markov decision process (POSMDP).

Markov decision processes (MDP) and semi-Markov decision processes (SMDP) are well-studied models of sequential decision making under uncertainty, which have proved to be versatile and effective in fields ranging from business to medicine. These two models assume, however, that the precise value of the current state can be identified at any time with perfect accuracy. For the semi-Markov process, time between state transition is permitted to have a general distribution, which is functionally dependent on both the present and the next state of the process. The Markov process is a special case of the semi-Markov process, in that times of transition from one state to the next are independent of the next state and are exponentially distributed.

There are applications, however, which can be formulated as a Markov decision pro-
cess or as a semi-Markov decision process in all respects except that only partial knowledge of the state is available to the observer. For example, in health care a physician may determine a treatment plan for a patient on the basis of signs, symptoms, and laboratory test results, which are only probabilistically related to the underlying physiological-psychological state of the patient. In maintenance and repair scheduling, an operator or inspector decides on the basis of the finished product whether or not to examine internal components of a machine. In education, a professor determines the best methodology to teach a specific subject by the results obtained from some tests applied to the students. Even though these results can only provide partial information about knowledge acquired with each methodology.

Problems that can be formulated using MDP (SMDP), but which suffer from an imperfect state observation, are usually referred to as partially observable Markov decision process POMDP (partially observable semi-Markov decision process POS-MDP) respectively. In this work, it is assumed that the core process is a finite state, discrete time controlled Markov (semi-Markov) process.

In partial maintenance problems, the objective is to determine actions which optimally alter a process performance for the case where inspection of the process state is uncertain and where the process performance can only be altered by partial or total replacement of its components. We assume that it is impossible to carry out direct inspection of these components due, for example, to their inaccessibility, or to the prohibitive expense of a thorough inspection. However, it may be possible to measure the effect of the process deterioration on the product and thereby provide indirect information concerning the state of the process.

The motivation of this research comes from the fact that although models and algo-
rithms have certainly been developed and explored, for some practical problems the
algorithms are not useful. "The applicability is limited in two ways. First, POMDP
and POSMDP are very data intensive, and second, solution of the POMDP or POS-
MDP model of most realistic problems can require substantial computational effort" [57](1991). This limitation, and the potential modeling applicability of the POMDP
(POSMDP) model, serves as a motivation for the study of numerical solution tech-
niques and hence for this dissertation. Survey papers of Monahan [38](1982), and
White [57](1991) deal specifically with POMDP.

For the POMDP, the algorithm which is most often referred to is one developed by
Sondik [43](1971): however, a large system state space will generally have an ad-
verse effect on the performance of the algorithm. In addition, in computing optimal
control policies using Sondik's algorithm, it has been observed that most of the time
is spent in finding subsets of linear constraints defining the optimal region from a
large number of linear constraints using linear programming. The number of linear
constraints and the computational time increases geometrically with the number of
actions and the time-horizon. Lark [24](1990) examines various algorithms for the
POMDP with different size problems, finding that problems that exceed the follow-
ing upper bound quickly exceed computer memory limits.

<table>
<thead>
<tr>
<th>Components of the POMDP</th>
<th>Upper Bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of States of the Core Process</td>
<td>10</td>
</tr>
<tr>
<td>Number of Alternatives</td>
<td>5</td>
</tr>
<tr>
<td>Number of Decision Periods</td>
<td>10</td>
</tr>
<tr>
<td>Number of States of the Observation Process</td>
<td>10</td>
</tr>
</tbody>
</table>

The same kind of problems, but magnified, are found with POSMDP algorithms
( the algorithm that is most often referred to, developed by White [51](1974), is
a generalization of Sondik's One Pass algorithm). Therefore, another approach is warranted.

To show the application of the model and methods described by this work, we develop an optimization control for roll surface degradation in the steel processing at the final operation stage (the Finishing Mill). This process consists of seven roll stands which transform steel slabs into strip product. The main feature of the Finishing Mill is to ensure the profile and the surface quality of the product. The surface condition of the rolls deteriorate with the rolling of successive slabs, and this deterioration is reflected in the strip quality. Eventually the rolls must be replaced so as to maintain the strip quality at an acceptable level. The deterioration is tracked using strip product observations and, if necessary, a decision is made to replace one or more roll sets. The problem in this situation is to infer the amount of roll deterioration from the observed product and decide which roll set, if any, to replace.

A major cost in operating the finishing mill is in roll maintenance and replacement. It is important that the decision of what rolls to change and when to change them be based on a sound economical and objective basis.

This research is organized as follows:

Chapter 2 presents a survey of computational algorithms and applications for the partially observed Markov decision process (POMDP) having finite state, action, and observation sets. Several computational procedures presented are convergence accelerating variants of, or approximations to, Sondik's algorithm.

Chapter 3 is concerned with the mathematical formulation of POMDP. This chapter contains two sections:
CHAPTER 1. INTRODUCTION

1. The motivation behind the use of POMDP.

2. A discussion of the formal development of the model (notation, operators, major properties, and assumptions).

Chapter 4 is concerned with the presentation and discussion of the more well known algorithms associated with the partially observed Markov optimization problem.

Computational considerations, limitations and restrictions are discussed. In addition, the computer code for these algorithms are developed by using the technical computing environment MATLAB.

Chapter 5 presents a new heuristic method to determine good approximate solutions to the partially observed Markov optimization problem.

Computational considerations, limitations and restrictions are discussed. In addition, the computer code for this heuristic is developed by using the technical computing environment MATLAB.

Chapter 6 compares the performance of the various algorithms associated with the partially observed Markov decision process. The comparison is based on two types of numerical evaluations:

CPU time This evaluation is performed because the main concern of this research is improving the speed of solution procedures for the POMDP.

Expected Reward This evaluation is performed to determine the differences in terms of the total expected reward between the algorithms.
Test problems are separated into four basic classes:

The General POMDP.
The "Close-to-Perfect Information" Problem.
The Completely Unobserved Markov Decision Process.
The Perfect Information Problem.

Chapter 7 shows how the finishing roll surface deterioration control at the Hot Strip Mill can be formulated as a POMDP using the ideas developed in previous chapters. This chapter describes the process, the steel metallurgy, the current situation at the Hot Mill, and the problem in which we are interested.

Chapter 8 provides a quantitative model of surface deterioration per each roll. The generation of these models has three important objectives:

- To determine the policy that maximizes the expected reward for rolled product. Simulation and the models of surface deterioration are used to accomplish this objective.

- To determine the optimal number of slabs rolled between two consecutive Finishing Mill roll changes. (optimal size of the Product Block).

- To propose a manner by which the frequency of sampling can be regulated.

Chapter 9 compares the results of our proposed heuristic for the Hot Strip Mill problem against the results of the simulator. In addition, this chapter provides
the optimal length of a Product Block, and a sampling plan by using the simulation program developed in chapter 8.

Chapter 10 provides a discussion of future research and extensions of the results within the dissertation.
This chapter deals with the mathematical formulation of POSMDP, the presentation and discussion of the more well known algorithms associated with the partially observed semi-Markov optimization problem, and the presentation of a new heuristic that can be used to solve large POSMDP problems.

Chapter 11 presents some conclusions and contributions.
Chapter 2

LITERATURE REVIEW

This chapter surveys algorithms and applications dealing with partially observable Markov decision processes (POMDP). The objectives of this chapter are: to show the interrelations between the various algorithms, to indicate the different development paths that have been taken in formulating them, and to provide insight concerning possible new solution techniques for the POMDP.

The Partially Observable Markov Decision Process is a natural extension of the Markov Decision Process that allows for incomplete information regarding the state of the system. At each decision time, the decision maker must choose an action based only on the incomplete information at hand. Bellman [5] (1957) developed the computational approach for analysing Markov decision processes with a finite planning horizon. It has been shown (see Sondik [43](1971), Bertsekas [7](1976), Monahan [28](1982)) that a POMDP can be converted into a MDP. Following an action, the decision maker observes an outcome which gives some imperfect knowledge related to the underlying core state of the system. Using Bayes’ rule, the decision maker can then re-examine the probabilities concerning the possible underlying core state.
The next decision would then be based on the new updated probabilities. This is the key to the transformation of a POMDP to a MDP. If these updated probabilities associated with each possible system state are used as a state descriptor, the resulting problem has the same form as an MDP and can, theoretically, be solved with the same algorithm. This transformation is not without cost. The resulting MDP would have a state space $\Pi$, continuous and not finite.

This chapter is divided in two sections:

**Section One** reviews some of the current algorithmic alternatives for solving discrete-time, finite POMDPs. The algorithms are classified in the following classes:

1. Exact algorithms for the General finite-horizon POMDP.

2. Approximate algorithms for the General finite-horizon POMDP.

3. Algorithms for Special Cases of the finite-horizon POMDP.

**Section Two** reviews some applications of the finite-horizon POMDP.

### 2.1 CURRENT ALGORITHMS FOR THE FINITE POMDPs

This section reviews some of the currently available algorithms for the finite POMDP with the optimality criterion of maximizing the expected reward during $T$ time peri-
ods \((T < \infty)\). The solution techniques are largely based on dynamic programming, and inherit their computational difficulties.

The algorithms are classified in the following classes:

### 2.1.1 EXACT ALGORITHMS FOR THE GENERAL FINITE HORIZON POMDP

Algorithms for exact solution of the finite-horizon problem are available, but these are computationally intractable for even modest-sized problems.

Sondik, in his Ph.D. dissertation [43] (1971), was the first to address and solve the computational difficulties associated with POMDP. He developed an exact algorithm, called the one-pass algorithm, for the finite horizon POMDP, and discovered that POMDP are not as computationally intractable as the general nondenumerable state MDP, because the optimal expected total reward as a function of the probabilities associated with each possible system state is piecewise linear and concave, which implies that it has a finite representation (see chapter 4 of this thesis).

It is common in the literature to mention the one-pass algorithm in parallel with Smallwood and Sondik's algorithm [42](1973). This latter algorithm is similar to the one-pass algorithm, except that one constraint set is omitted. This omission is an error, as pointed out by Mukherjee et al. [29] (1991). Smallwood and Sondik demonstrate that if only a finite number of control intervals remain, then the optimal reward function is a piecewise-linear, and concave function of the current state probabilities of the internal Markov process. Their algorithm to calculate the optimal control policy is a successive approximations procedure that iteratively determines the finite representation of the total reward function. A numerical example for a machine-maintenance problem is presented.
Satia and Lave [35](1973) developed an implicit enumeration algorithm for computing $\varepsilon$-optimal solutions to the finite horizon POMDP. The algorithmic tool chosen is stochastic decision tree popularized by, among others, Raiffa and Schlaifer (1961) with Branch and Bound. For large problems, decision trees will have computational difficulties. They also briefly discussed using the control-limit structure of the optimal policy in a machine replacement problem.

Monahan [28](1982) suggested that each support in the reward function should be tested, and discarded if dominated. His procedure is easy to understand and implement. However, it is intractable for all but the smallest problems, because it is time-consuming to determine the supports in the reward function.

Lark [24] (1990) developed a method for solving finite-horizon POMDPs which is based upon best-first heuristic search theory, and he provided an exploratory investigation of the performance of the One-Pass algorithm, Monahan’s algorithm, and some modifications of the Monahan algorithm (see chapter 4 of this thesis). In addition, he presents a heuristic search based procedure for solving the finite horizon, completely unobserved MDP (CUMDP).

Mukherjee and Seth [29](1991) pointed out that the algorithm given by Smallwood and Sondik [42] does not include all the constraints required to obtain optimal policies for partially observable Markov processes. They modify the algorithm of Smallwood and Sondik to correct the error, and to eliminate redundancies (see chapter 4 of this thesis).
2.1.2 APPROXIMATE ALGORITHMS FOR THE GENERAL FINITE HORIZON POMDP

All feasible algorithms involve reducing the possible states of the system to a finite grid of points. However, some methods adjust this grid dynamically as the algorithm progresses, and some keep the grid fixed. There is a natural advantage at the theoretical level to adjusted grid methods, because information garnered from early iterations may be used to choose a grid of points that adds resolution and flexibility where it is most needed. However, these advantages come at the cost of increased algorithm complexity. Sondik and Mendelssohn [45] (1979) suggest a policy improvement algorithm using a grid of points that includes only certain states of the state space \( \Pi \) that will actually be reached by a specified policy, which could be effective when the number of visited points is not large.

Cheng [10](1988) developed an algorithm called linear support algorithm, which can be used for finding both exact and approximate solutions. This algorithm adds gradient vectors based on the vertices of the generated regions that introduce the most error into the calculations. The algorithm is essentially choosing a finite number of points in \( \Pi \) to investigate based upon dynamic error bound calculations (see chapter 4 of this thesis). In addition, he developed an algorithm called the relaxed region algorithm, which is a modification of Sondik’s one-pass algorithm. Instead of finding an exact support region in the state space, a larger relaxed region which contains this exact region is found. In later steps, this relaxed region is modified (see chapter 4 of this thesis).
In contrast, a fixed grid method, being essentially a stationary, discrete approximation to a large state space, is a natural and often used approximation method in dynamic programming. Applications of these methods to solve POMDPs are the following:

Kakalik [22] (1965), and Eckles [14] (1968) consider a discrete space of probability distributions over the state of the process, and use an approximate value function based upon linear interpolation between this discrete space in their numerical work. The equations obtained are analogous to a completely observable problem and therefore can be solved using Howard's techniques; the disadvantage of their approaches is that since the computational speed of Dynamic Programming algorithms depends on the number of states, the "curse of dimensionality" obviously will overwhelm such attempts for any but the smallest problems.

Brumelle and Sawaki [8] (1978) developed a partition method for finite horizon POMDP, and a modified policy iteration algorithm for the infinite horizon POMDP.

Lovejoy [26] (1991) presented a procedure for approximating the uncountably infinite state space by a finite grid of points. He constructed this grid by using the Freudenthal triangulation concept and then by calculating the optimal strategy (and reward) at each point in the grid.

This grid is constructed to accomplish the following:

1. generate both upper and lower bounds for the optimal cost function.

2. generate approximate nonstationary and stationary policies.
3. bound the value loss relative to optimal for using these policies in the decision problem.

Lovejoy [26] affirms that for problems with \( n \) core states, the algorithm has computational requirements per iteration proportional to \( n^2 \) times the number of grid points used; however, the performance of the algorithm should be evaluated and compared with other solution procedures.

2.1.3 ALGORITHMS FOR SPECIAL CASES OF THE FINITE-HORIZON POMDP

Sawaragi and Yoshikawa [37](1970) developed the theory of POMDP with an uncountable action space and a countable core process state space.

Rhenius [33] (1974) considered POMDP where both the action and core process state spaces were Borel spaces.

Wang [49](1976), [50] (1977) developed a computational procedure for determining optimal policies for certain finite-state machine replacement problems. The models assumed two actions: do nothing, replace. Inspection is not allowed. The procedure described appears to be quite efficient for solving these special problems with more than two unobservable states.
2.2 APPLICATIONS OF THE POMDP

POMDP have been used to model a wide range of problems (Monahan [28] (1982)). Drake [13](1962) considered the decoding of a Markov source over a noisy channel. He formulated a three-alternative optimization problem and found the optimal control minimizing the expected cost of decoding the channel. In this case, the internal state of the Markov process corresponds to the state of the Markov source and the discrete observations represent the outputs of the noisy channel.

One of the major applications is the machine replacement and quality control problems (Eckles [14] (1968), Ross [34] (1971), White (52)(1977), [53](1978), [55](1979)), Ohnishi et al.[30] (1986)).

Pollock [32](1970) developed a two-state core process POMDP to model an optimal search effort for an object that moves between two regions. Before each move, the decision maker can choose which region to examine in order to locate the object.

Smallwood, Sondik, and Offensend [39](1970) used POMDP concepts in the development of a methodology for the analysis of the following health-care problems:

1. Medical Diagnosis and Treatment.
   This problem is concerned with identifying the state of the patient, and then prescribing treatment so as to influence the future state of the patient.

2. Health-Service Programs.
   This problem area is concerned with the allocation of resources by government agencies for various preventive health programs affecting certain population
groups, for example, clinical research, immunization, and health education.

This problem is concerned with designing a medical facility to supply the health service prescribed by the physician during the diagnosis and treatment of an individual patient.

4. Regional Health-System Design.
This problem area is concerned with designing a system to provide the requested health services for the population of a large geographical area.

Smallwood [40](1971) developed a two core state POMDP model to determine optimal teaching strategies.

White (1976) applied the theory of POMDP to design questionnaires in situations where responses may not be truthful.

Segall [38](1976) studied the problem of where to locate a common data file. He assumed that the demand rate for the data at one of the computer sites is an unobservable finite-state Markov chain, while the demand rate at the other site is a deterministic function of time. A POMDP model was formulated to determine the optimal file location over time.

Lane [23] (1989) presents an application of a POMDP for the intraseasonal decisions of the British Columbia commercial fishing fleet. The overall state of seasonal abun-
dance can not be observed directly and it is inferred from the observed catches. As the season progresses, more catches are made and thus, more information is obtained about the overall state of seasonal abundance. This new information about catch potential is then used to make decisions about when and where to fish in upcoming periods of the season.
Chapter 3

PARTIALLY OBSERVABLE MARKOV DECISION PROCESS

This chapter is devoted to the study of the finite-horizon, finite-state, finite-action, and finite-observation partially observed Markov decision process (POMDP).

The rich content of this model and its numerous extensions amply justify its study. A fundamental issue in this chapter is the determination of a sufficient statistic for POMDP. “A sufficient statistic is a representation of all the information contained in the past history that is useful for action selection” White and Scherer [58] (1994).

The chapter introduces the model and its associated control problem by studying each one of its components and properties. The state of the partially observable Markov process is proved to be the vector of state occupancy probabilities of the Markov process.
3.1 COMPLETELY OBSERVABLE MARKOV DECISION PROCESS

The definition of an optimal control problem, for either deterministic or stochastic systems, requires three components:

1. A decision or control model.

2. A set of admissible control actions.

3. An objective function.

These components for the optimal Markov control problem can be defined following Hernandez-Lerma [17](1989):

1. A DECISION OR CONTROL MODEL

A discrete-time, stationary, Markov control model consists of four elements 
\((X, A, P, c)\), where:

(a) \(X\) is the state space of the core process. The elements of \(X\) are called states \((s \in X)\).

(b) \(A\) is the action (or control) set. With each state \(s \in X\) we associate a non-empty subset \(A(s)\) of \(A\) \((A(s) \subseteq A)\), whose elements are the admissible actions when the system is in state \(s\).
System moves to a new state \( s(t+1) = s' \) according to \( p(\cdot | s, a) \).

- time \( t \)
- \( s(t) = s \)
- \( a(t) = a \)

Reward \( c(s', a') \) is received.

- action \( a(t+1) = a' \) is selected according to system state.

Figure 3.1: DECISION DIAGRAM FOR COMPLETELY OBSERVABLE MARKOV DECISION PROCESS

(c) \( P \) is the state transition probability matrix.

(d) \( c \) is the one step reward function.

The Markov Control Model \((X, A, P, c)\) is interpreted as representing a controlled stochastic system which is observed at times \( t = 0, 1, \ldots, T \); the state and control action at time \( t \) are denoted by \( s(t) \) and \( a(t) \), respectively, and the system evolves as follows: If the system is in state \( s(t) = s \) at time \( t \) and the control action \( a(t) = a \in A_t \) is chosen, then we receive a reward \( c(s, a) \) and the system moves to a new state \( s(t + 1) \) in \( X \) according to the probability distribution \( p(\cdot | s, a) \).

Once the transition into the new state, say, \( s(t + 1) = s' \), has occurred, a new control \( a' \in A(s') \) is chosen and the process is repeated (ref. fig. 3.1).
2. A SET OF ADMISSIBLE CONTROL ACTIONS

Represents the available decisions for each state.

3. AN OBJECTIVE FUNCTION

It is a function "measuring" the system's performance when a given action is used and the initial state of the system is given.

3.2 PARTIALLY OBSERVABLE MARKOV DECISION PROCESS

In the situation outlined in section 3.1, we assume that the state of the decision process, \( s(t) \), can be costlessly ascertained with certainty and without delay by the decision maker. While there are many "real world" problems for which the state of the process can be known precisely at each stage, there are many problems in which information about the state of the process can only be known imprecisely. Also, there are problems for which perfect state information is costly to obtain, such as the maintenance problem discussed below. Therefore, to our description of the Markov Control Model, we add the notion of an observation process \( z(t) \). We get information about \( s(t) \) through an observation or measurement process \( z(t) \).
3.2.1 COMPONENTS OF THE POMDP's

A partially observable Markov decision control process possesses the following components:

THE CORE PROCESS

The Core Process is an underlying Markov process. The process is characterized by the states labeled 1, 2, ..., \( N_s \). The process is described by \( N_s \times N_s \) transition matrices \( P^a = [p^a_{ij}] \) for each possible action \( a \), where \( p^a_{ij} \) is the probability that the state of the core process at time \( t + 1 \) will be \( j \) given that the state at time \( t \) is \( i \) and the decision made at time \( t \) is \( a \).

This process cannot be observed directly. The observer sees instead one of the \( N_z \) states of the observation process.

THE OBSERVATION PROCESS

This process is observed when the process is examined at time \( t \). The relationship between the states of the Core Process and the outcomes from the Observation Process at time \( t \) is modeled by defining the probability that outcome \( z \) occurs if the core process is in state \( i \), and that the action selected at time \( t - 1 \) is \( a \). In other words, if we assume that the observation process has \( N_z \) states labeled 1, 2, ..., \( N_z \), the decision taken at time \( t - 1 \) (\( a_{t-1} \)) is \( a \); and \( z(t) \) is the state of the observation process at time \( t \), then \( z(t) \) is related to the unknown \( s(t) \) by:

\[
R^a = [r^a_{iz}]
\]
where,

\[ r^a_{iz} = pr(z(t) = z|s(t) = i, a_{t-1} = a) \quad \forall t \]

The \( N_s \times N_z \) matrices \( R^a \) for each possible action \( a \), completely describes the output process.

If the Core Markov Process is completely observable then \( N_s = N_z \) and \( R = I \) where \( I \) is the identity matrix.

Both \( P^a \) and \( R^a \) \( \forall a \) are stochastic matrices with the following characteristic:

\[
\sum_{j=1}^{N_z} p^a_{ij} = \sum_{z=1}^{N_z} r^a_{iz} = 1 \quad \forall \quad i, \quad 1 \leq i \leq N_s
\]

**THE STATE OF KNOWLEDGE**

It is well-known that the distribution over the current core process state, conditioned on past history, is a sufficient statistic (see, for example, Bertsekas [7](1976), and White and Scherer [58](1994)). Since the core state space \( X = (1, 2, \ldots, N_s) \) is assumed to be finite, this sufficient statistic has the desirable property that it is described by a fixed, finite number of probabilities. The undesirable feature of this sufficient statistic is that its state space is uncountably infinite.

The probability that the Core Process is in a particular state can be calculated using the Markov property and basic probability relations.

Let all the information concerning the state of the process up to time \( t \), be represented by \( \xi(t) \). A portion of this information can be encoded in the form of the row vector \( \pi(t) \), where the \( i^{th} \) element of this vector is the probability that the process is in state \( i \) at time \( t \), given all the information about the
where:

\[ \pi_i(t) \geq 0 \quad \forall i \]
\[ \sum_{i=1}^{N_z} \pi_i(t) = 1 \quad \forall t \]

Let \( \Pi \) be the set of all \( \pi(t) \) vectors. That is, \( \pi(t) \in \Pi \).

Let \( \pi(0) = [\pi_1(0), \pi_2(0), \ldots, \pi_{N_z}(0)] \) be the probability vector of the initial state of the core process, that is, \( \pi_i(0) = pr(s(0) = i) \) for \( i = 1, 2, \ldots, N_z \). Assume that \( \pi(0) \) is known by the decision maker. As a result, the following data sequence characterizes \( \xi(t) \):

\[
\xi(0) = [\pi(0), z(0)] \\
\xi(t) = [\pi(0), z(0), a_0, z(1), a_1, \ldots, z(t-1), a_{t-1}, z(t)] \\
\xi(t+1) = [\xi(t), a_t, z(t+1)]
\]

(3.1)

It should be noted that \( \{\xi(t), t = 0, 1, 2, \ldots, T\} \) is a finite horizon Markov Decision Process.

The relation between \( s(t) \), \( z(t) \), \( \xi(t) \) and \( \pi(t) \) can be summarized in the following manner:

The change in the state of the Core Process from \( s(t) \) to \( s(t+1) \), when \( a_t = a \) is selected, is assumed to happen instantaneously at the frontier between time \( t \) and \( t + 1 \). Once \( s(t + 1) \) has been assumed by the Core Process, \( z(t + 1) \) is produced according to \( s(t + 1) \) and \( R^a \).
In the absence of other information, $\xi(t + 1)$ as a function of $\xi(t)$ and $z(t + 1)$ can be encoded as the vector $\pi(t + 1)$ in the following way:

The probability that $s(t + 1) = j$, given $\xi(t + 1)$, where $\xi(t + 1)$ is a function of $\xi(t)$ and $z(t + 1) = z$, can be obtained by using Bayes’ rule as follows:

If we denote:

$$pr\{s(t + 1) = j|\xi(t + 1) = f[\xi(t), a_t = a, z(t + 1) = z]\} = T_j(\pi(t)|z, a)$$

Then

$$\pi_j(t + 1) = T_j(\pi(t)|z, a) = \frac{\sum_{i=1}^{N_s} \pi_i(t)p_{ij}^a r_{jz}^a}{\sum_{i=1}^{N_s} \sum_{k=1}^{N_s} \pi_i(t)p_{ik}^a r_{kz}^a} \tag{3.2}$$

The proof is presented in section 5.1

We can write the above probability in matrix form as follows:

$$\pi(t + 1) = T(\pi(t)|z, a) = \frac{\pi(t)P^a R_z^a}{\pi(t)P^a R_z^a \mathbb{1}} \tag{3.3}$$

where:

$R_z^a$ is a diagonal matrix with $r_{jz}^a$ as its diagonal elements. That is:

$$R_z^a = \begin{bmatrix}
  r_{1z}^a & \cdots & \cdots & \cdots \\
  \cdot & r_{2z}^a & \cdots & \cdots \\
  \vdots & \vdots & \vdots & \vdots \\
  \cdot & \cdot & \cdots & a_{N_zz}
\end{bmatrix}
$$

1 is an $N_s$ dimensional column vector with all elements being 1.

As we can see, one of the main characteristics of the POMDP is the transformation of the information vector $\pi(t)$ from period to period via Bayes’ rule.
If we assume that the only source of information from the process is the sequence of outputs, and that the state of knowledge \( \pi(t) \) is computed after each output is received, then we can show that the sequence of states of knowledge \( \pi(0), \pi(1), \ldots \) is a finite horizon Markov Process. It is, however, an unusual Markov Process for although the state space \( \Pi \) (the set of all \( \pi(t) \) vectors) is a continuum, the conditional probability density function of the next state is a discrete probability distribution. To prove this assertion, we should note the following: Because \( \pi(t) \) is calculated after each output, the sequence of states of knowledge must be contained in \( \xi(t) \), since \( \xi(t) \) is the complete state of knowledge at time \( t \). The state of knowledge at time \( t + 1 \), given \( z(t + 1) = z \) and \( \xi(t) \), is \( T(\pi(t)|z, a) \). The probability that the next output, \( z(t + 1) \), will be \( z \) given the current state of knowledge \( \xi(t) \) is:

\[
pr\{z(t + 1) = z|\xi(t)\} = \{z|\pi(t), a\} = \sum_{i=1}^{N_s} \sum_{j=1}^{N_s} \pi_i(t) p_{ij}^s r_{jz}^s
\]

Thus the next state distribution function is determined by:

\[
pr(\pi(t + 1) \in \Upsilon|\xi(t)) = pr(\pi(t + 1) \in \Upsilon|\pi(t), \pi(t - 1), \ldots, \pi(0))
\]

\[
= pr(\pi(t + 1) \in \Upsilon|\pi(t))
\]

\[
= \sum_{z:T(\pi(t)|z) \in \Upsilon} \{z|\pi(t)\}
\]

(3.4)

where \( \Upsilon \) is an arbitrary set of vectors in \( \Pi \), and the sum is taken over those \( z(t + 1) \) for which \( T(\pi(t)|z, a) \in \Upsilon \).

Hence the sequence of random variables \( (\pi(t), t \geq 0) \) is a Markov process.
THE REWARD STRUCTURE

While the process is operating, rewards, which depend on the outputs generated and the hidden state transitions, are obtained.

If:

\[ C_{ij}^a \] is defined as the reward of making a transition from state \( i \) at time \( t \) to state \( j \) at time \( t + 1 \) and producing output \( z \), when alternative \( a \) was chosen at time \( t \),

\( \gamma_i^a \) is defined for all \( t \) as the expected immediate reward of making one transition and producing certain outputs, given \( s(t) = i \) and \( a_t = a \),

\( \gamma^a \) is a \( N_s \)-dimensional column vector with \( \gamma_i^a \) as its \( i \)-th element,

then:

\[ \gamma_i^a = \sum_{j=1}^{N_s} \sum_{z=1}^{N_z} C_{ij}^a p_{ij}^a \gamma_j^z \]  \hspace{1cm} (3.5)

The vector \( \gamma^a = [\gamma_i^a] \) is an important element of the control process since \( \pi(t) \gamma^a = \sum_{i=1}^{N_s} \pi_i(t) \gamma_i^a \) is the expected immediate reward of the next transition, given the state of knowledge \( \pi(t) \).

If after \( T \) periods, the process is terminated with a reward vector, \( \gamma(T) = [\gamma_i(T)] \), where \( \gamma_i(T) \) is the expected reward obtained if the process terminates in state \( i \), we can establish a recursive relationship for expected reward over time as follows:

Let:
$C'(\pi)$ be the expected optimal reward of operating the process from time $t$ to the end of the decision horizon, with state of knowledge $\pi(t)$, and termination reward vector $\gamma(T)$;

$T(\pi(t)|z,a)$ be the new state of knowledge;

$z$ be the observation at time $t + 1$;

$\pi(t)$ be the current state of knowledge.

As a result (maximization case),

$$C^T(\pi(T)) = \pi(T)\gamma(T)$$

$$C'(\pi(t)) = \max_{a \in A} \left[ \sum_{i=1}^{N_x} \sum_{j=1}^{N_x} \sum_{l=1}^{N_x} \pi_i(t) p_{ij}^a r_{lj}^a C^{t+1}(T(\pi(t)|z,a)) \right] , t \geq 1$$

(3.6)

This equation establishes that the expected reward from time $t$ to the end of the decision horizon is equal to the expected immediate reward of making one transition plus the expectation over $z$ of the expected reward from time $t + 1$ to the end.

In this representation, the state variable is known exactly; hence, the above equation is the recursive equation of a completely observable Markov Decision Process. In other words, with this equation, we convert the POMDP into an equivalent (completely observable) Markov Decision Process. The associated MDP for a POMDP is the MDP which shares the reward structure and probabilistic dynamics of the POMDP and has perfect state information at each
THE CONTROL PROBLEM

To control the process there are $N_a$ alternatives that are represented by a complete set of parameters for the process $(P^a, R^a, \gamma^a)$.

The controller must choose an alternative that is a set of parameters at each time period. The selected alternative describes the process operation only over that time period. At the next time period, another, but not necessarily different, alternative is selected. This control means choosing the alternatives that maximize the expected reward obtained for operating the process during $T$ time periods with termination reward vector $\gamma(T)$.

Let $X = (1, 2, ..., N_x)$ and $Z = (1, 2, ..., N_z)$ denote the core process state space and observation state space, respectively, and let $A$ denote the finite action set. Let $\Pi = \{ \pi(t) : \pi(t) \geq 0 , \sum_{i=1}^{N_x} \pi_i(t) = 1 \ \forall t \}$, i.e., the set of probability distributions on $X$. The control process is initiated with a known probability distribution over the state space $X$, $\pi(0) \in \Pi$. Let $\xi(t) = [\pi(0), z(0), a_0, z(1), a_1, ..., z(t - 1), a_{t-1}, z(t)]$ denote the information concerning the state of the process up to time $t$, which can be encoded in the form of the row vector $\pi(t)$. At the beginning of time period $t$, $\xi(t)$ or $\pi(t)$ contains all the information that the decision maker can use to assess the state of the core process. If, based on this information, the decision maker chooses action $a_t$, the following sequence of events is initiated:

1. A reward $\gamma_t^a$ is received if $s(t) = i$ and $a_t = a$.  

2. The core process transits to another state, \( j \), in accordance with the known transition probabilities \( p_{ij} = \text{pr}(s(t+1) = j|s(t) = i, a_t = a) \).

3. An observation \( z(t+1) = z, z \in Z \) is observed in accordance with the known probabilities \( r_{jz} = \text{pr}(z(t+1) = z|s(t+1) = j, a_t = a) \).

4. Time increments by one, \( \xi(t+1) = [\xi(t), a_t, z(t+1)] \) or \( T(\pi(t)|z, a) \) are used to choose action \( a_{t+1} \), and the process repeats.

If we define \( \delta_t(\pi) \) as the alternative that maximizes the expected reward from time \( t \) to the end of the decision horizon, the sequence of functions, \( \delta_t, 1 \leq t \leq T \) constitutes the complete solution of the control problem with \( T \) or fewer time periods remaining and terminal reward vector \( \gamma(T) \).

The relation between the sequence of functions \( \delta_t, 1 \leq t \leq T \) and the states of knowledge \( \pi(t), 1 \leq t \leq T \) is conceptually useful. It provides a decomposition of the optimal controller into two parts:

**Estimator Part** uses at time \( t \) the measurement \( z(t) \) and the control \( a_{t-1} \) to generate the probability distribution \( \pi(t) \).

**Actuator Part** generates the optimal control alternative \( \delta_t \) as a function of \( \pi(t) \).

Figure 3.2 shows the POMDP representation, with the decomposition of the optimal controller into an estimator, and an actuator.
Figure 3.2: PARTIALLY OBSERVABLE MARKOV DECISION PROCESS SHOWING SEPARATION INTO AN ACTUATOR AND AN ESTIMATOR.
3.2.2 POMDP PROPERTIES

In general, it is not easy to solve a Markov Decision Model with a continuous state space. Fortunately, the equivalent MDP of the POMDP possesses special properties that help us to determine useful computational algorithms to find an optimal policy. Sondik [43](1971)

1. Boundedness,

2. Monotonicity,

3. Contraction,

4. Piecewise Linearity and Concavity of the value function $C^t(\pi)$ for all $t$.

It is shown by Cheng [10](1988), and Whitt [62](1978) that these properties are present in a POMDP.

For us, the piecewise linearity and concavity of the value function $C^t(\pi)$ are the more important properties of a POMDP. Piecewise linearity of the value function $C^t(\pi)$ was first discovered by Astrom [4] (1965), but was formally introduced by Sondik [43] (1971).

To prove that $C^t(\pi(t))$ is piecewise linear and concave for all $t$, Sondik uses an induction on $t$, and the following lemma:

**Lemma:** $T(\pi(t)|z,a)$ preserves straight line segments. That is, if $0 \leq \beta \leq 1$, $\bar{\beta} = 1 - \beta$ and $\pi^1(t), \pi^2(t) \in \Pi$ then $\beta \pi^1(t) + \bar{\beta} \pi^2(t)$ is a straight line segment in
Figure 3.3: PRESERVATION OF STRAIGHT LINES

$\Pi$ with end points $\pi^1(t), \pi^2(t)$. If the transformation of this line segment for fixed $z(t+1) = z$ and $a_t = a$ is considered, then:

$$T(\beta \pi^1(t) + \beta \pi^2(t)|z, a) = \mu T(\pi^1(t)|z, a) + \mu T(\pi^2(t)|z, a)$$

where, as $\beta$ ranges between 0 and 1, $\mu$ ranges between 0 and 1. Hence the image of a straight line segment under $T(.|z, a)$ is a straight line segment. That is (ref. fig. 3.3):

$$\mu = \frac{\beta \{z|\pi^1(t)\}}{\beta \{z|\pi^1(t)\} + \beta \{z|\pi^2(t)\}}, \quad \bar{\mu} = 1 - \mu$$

$C^T(\pi(T)) = \pi(T)\gamma(T)$ is not only linear in $\pi(t)$, $\forall t$, but also piecewise linear and concave. Suppose that $C^{t+1}(\pi(t + 1))$ is piecewise linear and concave. Since the maximum of concave, piecewise linear functions is also concave, piecewise linear, and the sum of such functions preserves these properties, it is sufficient to show that $\{z|\pi(t), a\}C^{t+1}(T(\pi(t)|z, a))$ is concave, piecewise linear.

Let:
\[ f(\pi(t)) = \{z|\pi(t), a\}C^{t+1}(T(\pi(t)|z, a)) \]
\[ \pi(t)\beta = \beta \pi^1(t) + \bar{\beta} \pi^2(t) \]

(3.7)

for defined \( \beta \) as in the lemma.

\[
\begin{align*}
    f(\pi(t)\beta) & = \{z|\pi(t)\beta, a\}C^{t+1}(T(\pi(t)\beta|z, a)) \\
                 & = \{z|\pi(t)\beta, a\}C^{t+1}[\mu T(\pi^1(t)|z, a) + \bar{\mu} T(\pi^2(t)|z, a)] \\
                 & \geq \{z|\pi(t)\beta, a\}[\mu C^{t+1}[T(\pi^1(t)|z, a)] + \bar{\mu} C^{t+1}[T(\pi^2(t)|z, a)]] \\
                 & = [\beta\{z|\pi^1(t), a\} + \bar{\beta}\{z|\pi^2(t), a\}] [\mu C^{t+1}[T(\pi^1(t)|z, a)] + \bar{\mu} C^{t+1}[T(\pi^2(t)|z, a)]] \\
                 & = \beta\{z|\pi^1(t), a\}C^{t+1}[T(\pi^1(t)|z, a)] + \bar{\beta}\{z|\pi^2(t), a\}C^{t+1}[T(\pi^2(t)|z, a)] \\
                 & = \beta f(\pi^1(t)) + \bar{\beta} f(\pi^2(t))
\end{align*}
\]

(3.8)

This proves concavity.

Piecewise linearity is proved under the same scheme. Let \( C^{t+1} = \pi(t)\alpha(\pi) \). Then:

\[
\begin{align*}
    f(\pi(t)) & = \{z|\pi(t), a\}C^{t+1}[T(\pi(t)|z, a)] \\
              & = \{z|\pi(t), a\} \frac{\pi(t)P^aR^a}{\{z|\pi(t), a\}} \alpha[T(\pi(t)|z, a)] \\
              & = \pi(t)P^aR^a \alpha[T(\pi(t)|z, a)] \\
              & = \pi(t)\xi(\pi(t))
\end{align*}
\]

(3.9)

where \( \xi \) is a piecewise constant over \( \Pi \) thus proving piecewise linearity.
3.2.3 COMPUTATIONAL EXPRESSIONS FOR $C^t(\pi(t))$

The concavity of $C^t(\pi(t))$, and the fact that it is composed of a finite number of piecewise linear segments allows $C^t(\pi(t))$ to be computed in a very simple fashion:

$$C^t(\pi(t)) = (\pi(t))(\alpha^t(\pi))$$  \hspace{1cm} (3.10)

Where:

$\alpha^t(\pi)$ is a piecewise constant vector over $\pi$.

Sondik and Smallwood [42](1973) show that " the piecewise constancy of $\alpha^t(\pi)$ allows us to partition the state space $\Pi$ into sets, and deal with a state space consisting of a set of points rather than a continuum ".

Let:

$\hat{R}^t_k$ be the region $k$ in $\Pi$ at stage $t$, where $\alpha^t(\pi)$ has the vector value:

$$\alpha^t_k = [\alpha^t_{k1}, \alpha^t_{k2}, ..., \alpha^t_{kN_k}]$$

$\Lambda^t$ be the finite set of values of $\alpha^t(\pi)$. That is:

$$\Lambda^t = [\alpha^t_1, \alpha^t_2, ...]$$  \hspace{1cm} (3.11)

$\hat{R}^t$ be the associated partition of $\Pi$ at stage $t$.

Then:

$$\hat{R}^t_k = [\pi|C^t(\pi(t)) = (\pi(t))\alpha^t_k]$$

$$\hat{R}^t = \bigcup_k \hat{R}^t_k$$
CHAPTER 3. PARTIALLY OBSERVABLE MARKOV DECISION PROCESS

$C^t(\pi(t))$ is completely determined by the set $\Lambda^t$.

Associated with each element $\alpha^t_k$ in $\Lambda^t$ is the $k^{th}$ region of $\Pi$, i.e., $\tilde{R}^t_k \subset \tilde{R}^t$.

For each $t$, there exists a finite set $\Lambda^t$ of vectors such that:

$$C^t(\pi(t)) = \max \{ \pi(t)\alpha^t(\pi) : \alpha^t(\pi) \in \Lambda^t \}$$

From this result, it follows that the set $\Lambda^t$ can be determined from $\Lambda^{t+1}$ in the following manner:

$$C^t(\pi(t)) = \max_{a \in A} \left[ \pi(t)\gamma^a + \sum_{z=1}^{N_z} \sum_{i=1}^{N_i} \sum_{j=1}^{N_j} \pi_i(t)p^a_{ij}r^a_{ijz}C^{t+1}(T(\pi(t)|z, a)) \right] \quad (3.12)$$

$$C^t(\pi(t)) = \max_{a \in A} \left[ \pi(t)\gamma^a + \sum_{z=1}^{N_z} \{z|\pi, a\}C^{t+1}(T(\pi(t)|z, a)) \right] \quad (3.13)$$

where:

$$\{z|\pi(t), a\} = \pi(t)P^aR_z^a1$$

$$T(\pi(t)|z, a) = \frac{\pi(t)P^aR_z^a}{\{z|\pi, a\}}$$

$$1 = [1, 1, ..., 1]^T$$

$$R_z^a = \text{diag}[r^a_{iz}]$$

$$C^t(\pi(t)) = \max_{a \in A} \left[ \pi(t)\gamma^a + \sum_{z=1}^{N_z} \pi(t)P^aR_z^a\alpha^{t+1} \right]$$

$$C^t(\pi(t)) = \max_{a \in A} \left[ \pi(t)\gamma^a + \sum_{z=1}^{N_z} P^aR_z^a\alpha^{t+1} \right]$$

Where $\alpha^{t+1}$ is such that:

$$\pi(t)P^aR_z^a(\alpha^{t+1} - \alpha') \geq 0 \quad \forall a' \in \Lambda^{t+1}$$
CHAPTER 3. PARTIALLY OBSERVABLE MARKOV DECISION PROCESS

If we define:

$$\Lambda^t = \{\alpha^t : \alpha^t = \gamma^a + \sum_{i=1}^{N_a} P^a_R^a \alpha^{t+1}\}$$

Thus,

$$C^t(\pi(t)) = \max_{a \in A} \{\pi(t)\alpha^t\}$$
Chapter 4

POMDP EXISTING SOLUTION TECHNIQUES

This chapter is concerned with the presentation of some well-known potentially tractable algorithms associated with the partially observed Markov optimization problem presented in Chapter 3. In general, these algorithms involve reducing $\Pi$ to either a finite set of points or a finite partition.

We present the following algorithms:

1. One-Pass Algorithm [43].

2. Monahan Algorithm [28].

3. Relaxed Region Algorithm [10].
4. Mukherjee and Seth's Algorithm [29].

5. $MPAO^*$ Algorithm [24].

6. Linear Support Algorithm [10].

One-Pass, Monahan, Mukherjee, and $MPAO^*$ are exact algorithms for POMDPs. The first three algorithms do a complete specification of the set of gradients ($\alpha$ vectors) necessary to define the optimal reward function at each period of time. This set is finite for any finite time horizon, but its cardinality can grow exponentially in time. Unfortunately, an intractable number of gradients can be generated after only a few iterations in applied problems. $MPAO^*$ involves representing a POMDP as a decision tree. In this representation, solution strategies for the POMDP correspond to special subtrees of the decision tree. The reward of each strategy corresponds to the reward of the particular subtree which represents that strategy. It appears that $MPAO^*$ must generate and evaluate a large fraction of the total number of basic subtrees before finding a solution.

Linear Support is an approximate, variable grid algorithm that can be stopped when the number of gradient vectors generated arrives at a particular upper limit, or when the maximal difference between the exact solution and the approximate solution is less than a given error. This algorithm adds gradient vectors based on the vertices of the generated regions that produce the most error. As a result, this algorithm is choosing a finite number of points in $\Pi$ to investigate dynamic error bound calculations. If the maximal error is stored at each iteration, it is possible to calculate a bound on the total error between the generated reward function and the optimal reward function.
Relaxed Region is an approximate, variable grid algorithm similar to the One Pass algorithm. The algorithm reduces the number of generated regions by omitting some constraints used to define a region. This relaxation caused by omitting constraints risks generating too large regions, in that some operative boundaries might be missed.

4.1 THE ONE-PASS ALGORITHM

Sondik (1971) proves that for the finite horizon case the reward function is piecewise linear and concave on Π and, hence, can be represented by a finite number of vectors. This result guarantees that Π can be partitioned into a finite number of concave subsets and that the value function is linear on each of these subsets. The algorithm is named “One-Pass” because the optimal control for each state is found by using a single computational pass over the state space Π, for each time period the process is to operate.

Considering that $C^t(\pi(t))$ is piecewise-linear and concave, $C^t(\pi(t))$ is written as:

$$C^t(\pi(t)) = \max_k (\pi(t) \alpha^t_k) \quad (4.1)$$

where:

$$\alpha^t_k = [\alpha^t_{k1}, \alpha^t_{k2}, ..., \alpha^t_{kN_s}]^T$$

Sondik then shows that the dynamic programming recursive equation used to calculate the function $C^t(\pi(t))$ reduces to:

$$C^T(\pi(T)) = \pi(T)\gamma(T)$$
\[ C^t(\pi(t)) = \max_{a \in A} \{ \pi(t)[\gamma^a + \sum_{z=1}^{N_z} P^a R_z^a \alpha^{t+1}_{l(\pi^t, a, z)}] \} \quad t < T \]

where:

\( l(\pi, a, z) \) denotes the subscript \( k \) of the element of \( \Lambda^{t+1} \) (\( \Lambda^{t+1} = [\alpha_1^{t+1}, \alpha_2^{t+1}, \ldots] \)) that maximizes \( T(\pi(t) | z, a) \alpha_{k}^{t+1} \).

Sondik developed the following algorithm for finding the optimum \( \alpha \)-vectors and the corresponding control alternatives at a number of distinct points in the space of information vectors:

\begin{itemize}
  \item **Step 0:** Set \( t = T \).
    
    Define \( \Lambda^T = \Lambda^{T'} = \gamma(T) \).

  \item **Step 1:** If \( t = 0 \) go to Step 11. Otherwise set \( t = t - 1 \).

  \item **Step 2:** Select an arbitrary state \( \pi(t) \) as an initial point, and put it into the \( \pi \)-search table at time \( t \) with an unmarked attribute.

  \item **Step 3:** Pick a \( \pi(t) \) from the \( \pi \)-search table at time \( t \), and mark it. Denote this \( \pi(t) \) as \( \pi^0(t) \).

  \item **Step 4:** With this selected state \( \pi^0(t) \), calculate the indices \( l(\pi^0(t), a, z) \) \( \forall a, z \) as following:
\end{itemize}
\[ l(\pi^0(t), a, z) = \{ k| \max_k \left( \frac{\pi^0(t)P^aR^a_z}{\pi^0(t)P^aR^a_{z-1}} \right) \alpha^{t+1}_k : \alpha^{t+1}_k \in \Lambda^{(t+1)} \} \]

**Step 5:** By using \( l(\pi^0(t), a, z) \) values, we calculate:

\[ \Lambda^t = \bigcup_{a \in A^t} \{ \gamma^a + \sum_{z=1}^{N_z} P^aR^a_z \alpha^{t+1}_{l(\pi,a,z)} \} \]

**Step 6:** Proceed to Step 10 if all the elements in \( \Lambda^t \) have a mark.

**Step 7:** Select \( \alpha^t_j \) from \( \Lambda^t \).

Let \( a' \) be the alternative from which \( \alpha^t_j \) was obtained.

Identify the region of the information vector space over which \( \alpha^t_j \) is optimal, and identify the \( \alpha \)-vectors of the neighboring regions. To do this, the algorithm considers two ways by which the expression:

\[ \alpha^t_j = \gamma^a' + \sum_{z=1}^{N_z} P^a'R^a'_{z} \alpha^{t+1}_{l(\pi,a',z)} \]

might change when \( \pi(t) \) changes from initial \( \pi^0(t) \) to \( \pi^1(t) \).

1. \( l(\pi^1(t), a', z) \) changes from \( l(\pi^0(t), a', z) \) for a given output \( z \)

This case occurs when we move \( \pi(t) \) away from \( \pi^0(t) \) until one of the corresponding points \([T(\pi(t)|1, a'), T(\pi(t)|2, a'), \ldots, T(\pi(t)|N_z, a')]\) in the space of the information vectors at time \( t+1 \) eventually crosses the boundary of a region. The condition for the point \( T(\pi(t)|z, a') \) to remain in the same region specified by \( l(\pi^0(t), a', z) \) is:

\[ T(\pi(t)|z, a')(\alpha^{t+1}_k) \geq T(\pi(t)|z, a')(\alpha^{t+1}_k) \ \forall \ k \]  \hspace{1cm} (4.3)
Or, by substituting [3.4] in [4.3], we obtain:

\[ \pi(t)P^a'R_z^a'[\alpha_{t+1}^j - \alpha_{k}^{t+1}] \geq 0 \quad \forall \ k \]  

(4.4)

There will be a set of these inequalities for each output \( z \).

2. The optimum control alternative changes.

The second condition to remain in the region of \( \alpha_j^t \) is that the control alternative \( a' \) must be optimal. The condition for \( a' \) to remain the optimal control alternative is:

\[ \pi[\alpha_j^t - \alpha_k^t] \geq 0 \quad \forall \ k \in \Lambda^t \]  

(4.5)

The sets of inequalities in [4.4] and [4.5] along with the following condition:

\[ \sum_{i=1}^{N_t} \pi_i(t) = 1 \quad \pi_i(t) \geq 0 \]  

(4.6)

are solved using linear programming to identify the region over which \( \alpha_j^t \) is optimal. All vertices of these generated region(s) are determined and added with an unmarked attribute to the \( \pi \)-search table at time \( t \) for later examination. Each generated vertex is used to determine new regions and their corresponding vertices. Each vertex is used to determine regions once, and the regions which generate this vertex do not need to be determined again.

**Step 8** If new vertices were found in Step 7, place \( \alpha_j(t) \) in \( \Lambda^t' \), and mark \( \alpha_j(t) \) in \( \Lambda^t \).

Otherwise mark \( \alpha_j(t) \) in \( \Lambda^t \).

**Step 9** Proceed to Step 6.
Step 10 Return to Step 3 if the $\pi$-search table has $\pi$-values with an unmarked attribute. Otherwise go to Step 1.

Step 11 Terminate. $\Lambda'$ possesses the $\alpha$-vectors for all the regions at time $t$ ($0 \leq t \leq T$).

This algorithm uses a linear programming approach to identify the regions in $\Pi$ which correspond to different policies. The number of constraints depends on the number of states for the observation process, the number of possible actions, and the subset of regions found at time $t + 1$. In order to find the boundaries of a given region, the LP is solved repeatedly, each time with a different constraint acting as the objective function. This process has to be repeated for each region. The number of regions is exponential with respect to the time horizon. Thus, the algorithm may involve the solution of an exponential number of LP's, making it intractable for many realistic problems.

Papadimitriou and Tsitsiklis [31] (1987) explore the computational complexity of POMDP, concentrating on Sondik's algorithm. They arrive at the same conclusion, that is, that the number of $\alpha$'s although finite for a finite horizon, may be an exponential function of the time horizon. Even worse, it may not be possible, even if a solution can be obtained, to devise a storage scheme so that the solution may be searched in polynomial time. This means that even if the problem can be solved, the solution may not be implementable.

The computer code for the One-Pass algorithm was developed in the present study by using the technical computing environment MATLAB. The results obtained with this code were compared with the results obtained by Sondik, by using the example
CHAPTER 4. POMDP EXISTING SOLUTION TECHNIQUES

presented in Sondik [43](1971), (see chapter 6).

4.2 THE MONAHAN ALGORITHM

The estimation of the set of \( \alpha \)'s at time \( t \) from the set of \( \alpha \)'s at time \( (t + 1) \) by using the One-Pass algorithm is not an easy task, because we need to calculate the indices \( l(\pi(t), a, z) \forall a, \) and \( z. \)

Monahan considers an alternative approach. Instead of finding \( \Lambda^t \) for some \( \pi(t) \), an arbitrary \( \alpha_m \in \Lambda^{(t+1)} \) is chosen for each observation \( z \), and the set \( \Lambda^t = \bigcup_{a \in A_t} \{ \gamma^a + \sum_z P^a R^a_m : \alpha_m \in \Lambda^{(t+1)} \} \) is calculated.

The cardinality of \( \Lambda^t \) (worst case) is \((\#A_t)(\#\Lambda^{(t+1)})^{N_z}\).

Where:
\(\#v\) represents the cardinality of set \( v \).

Hence, by induction, we can show that in the worst case,

\[ \#\Lambda^0 = (\#A_0)(\#A_1)^{N_z}\ldots(\#A_{T-1})^{N_z^{T-1}} \]

If \( \#A_t = K, t = 0, \ldots, T - 1, \) then:

\[ \#\Lambda^0 = K^{\frac{(T-1)}{(N_z^{T-1})}} \]

This approach is simple to use. However, when the number of actions, observations, stages, or \( \alpha \)'s in \( \Lambda^t \) is large, it is time-consuming to determine the \( \alpha \)'s that define regions.

The Monahan algorithm operates as follows:
Step 1: Set $t = T$.
Define:

$$\Lambda^T = \Lambda^{T'} = \gamma(T)$$

Step 2: If $t = 0$, go to Step 4. Otherwise, set $t = t - 1$, and calculate:

$$\Lambda^t = \bigcup \{ \gamma^a + \sum z P^a R^a_m \alpha : \alpha \in \Lambda^{(t+1)'} \}$$

Set $\Lambda^{t'} = \phi$

Step 3: If $\Lambda^t = \phi$ go to Step 2. Otherwise, select $\alpha \in \Lambda^t$, and by using linear programming determine the existence of $\pi_\alpha \in \Pi(t)$ such that:

$$\pi_\alpha \alpha \geq \pi_\alpha \alpha' \quad \forall \alpha' \in \Lambda^t \cup \Lambda^{t'}$$

If so, place $\alpha$ in $\Lambda^{t'}$ and remove $\alpha$ from $\Lambda^t$.
Repeat Step 3.

Step 4: Terminate. $\Lambda^{(t=0)'}$ is the set of expected reward vectors corresponding to nondominated strategies.

4.3 THE RELAXED REGION ALGORITHM

Cheng [10] (1988) modified the One-Pass algorithm to reduce the number of generated regions. The major difference between the Relaxed Region algorithm and Sondik’s algorithm is in their method of defining regions. The total number of regions generated from the One-Pass algorithm is usually bigger than the number of regions produced in the Relaxed Region algorithm.
From a computational point of view, there is a difference in the number of constraints used to define a region between Sondik's algorithm and the Relaxed Region algorithm. In the Relaxed Region algorithm only those constraints which concern the optimal action are considered. As a result, the One-Pass algorithm usually has a much larger number of constraints than the Relaxed Region algorithm for defining a region, and consequently it requires more computer memory and computational time.

This algorithm works as follows:

**Step 0:** Set $t = T$.

Define $\Lambda^T = \Lambda^{T'} = \gamma(T)$.

**Step 1:** If $t = 0$ go to Step 10. Otherwise set $t = t - 1$.

**Step 2:** Select an arbitrary state $\pi(t)$ as an initial point, and put it into the $\pi$-search table at time $t$ with an unmarked attribute.

**Step 3:** Pick an unmarked $\pi(t)$ from the $\pi$-search table at time $t$, and mark it. Denote this $\pi(t)$ as $\pi^0(t)$.

**Step 4:** With this selected state $\pi^0(t)$, calculate the indices $l(\pi^0(t), a, z) \forall a, z$ as follows:

$$l(\pi^0(t), a, z) = \{k| \max_k \frac{\pi^0(t)P_aR^t_z}{\pi^0(t)P_aR^t_z\gamma} \alpha_k^{t+1} : \alpha_k^{t+1} \epsilon \Lambda^{(t+1)'}\}$$
Step 5: By using \( l(\pi^0(t), a, z) \) values calculate:

\[
\Lambda^t = \bigcup_{a \in \Lambda_t} \left\{ \gamma^a + \sum_{i=1}^{N_t} P^a R^a_i \pi^t_{l(\pi^0(t), a, z)} \right\}
\]

Step 6: Determine \( \alpha^*(t) \in \Lambda^t \) such that:

\[
\pi^0(t)\alpha^*(t) \geq \pi^0(t)\alpha'(t) \quad \forall \alpha' \in \Lambda^t
\]

Step 7: Use the following inequalities to identify the region of the information vector space over which \( \alpha^*(t) \) is optimal:

\[
\begin{align*}
\pi(t)P^a R^a_{t+k} [\alpha^t_{l+k} - \alpha^t_k] & \geq 0 \quad \forall \ k \\
\pi(t)[\alpha^*(t) - \alpha'(t)] & \geq 0 \quad \forall \alpha' \in \Lambda^t \\
\sum_{i=1}^{N_t} \pi_i(t) & = 1 \quad \pi_i(t) \geq 0
\end{align*}
\]

(4.7)

All vertices of this generated region(s) are determined, and added with an unmarked attribute to the \( \pi \)-search table at time \( t \) for later examination.

Step 8 if \( \alpha^*(t) \notin \Lambda^t \), place \( \alpha^*(t) \) in \( \Lambda^t \).

Step 9 Return to Step 3 if \( \pi \)-search table has \( \pi \)-values with an unmarked attribute. Otherwise go to Step 1.

Step 10 Terminate. \( \Lambda^t \) possesses the \( \alpha \)-vectors for all the regions at time \( t \) (\( 0 \leq t \leq T \)).

The computer code for the Relaxed Region algorithm in this present study was developed by using the technical computing environment MATLAB.
4.4 MUKHERJEE AND SETH'S ALGORITHM

Mukherjee and Seth [29](1991) modify the algorithm of Smallwood and Sondik [42](1973) to accomplish the following:

1. Reduce the number of $\alpha$-vectors that should be considered, by identifying and eliminating redundancies.

2. Correct an error that was found while they were solving the numerical example presented in Smallwood and Sondik [42](1973).

The error was detected because different $\alpha$-vectors were identified for different chosen values of the initial vector $\pi$.

The reason for the error is that the set of inequalities [4.4] and [4.5] failed in identifying all the $\alpha$-vectors.

The suggested algorithm is as follows:

**Step 0:** Set $t = T$.

Define $\Lambda^T = \Lambda^{T'} = \gamma(T)$.

**Step 1:** If $t = 0$ go to Step 10. Otherwise set $t = t - 1$.

**Step 2:** Generate a list $\Lambda^t$ of all possible $\alpha$-vectors at time $t$ (the number of $\alpha$-vectors is based on the number of the $\alpha$-vectors at time $(t+1)$, the number of actions, and the number of observations).

\[
\alpha_j^a(t) = \gamma^a + \sum_{i=1}^{N_i} P^a R^a_i \alpha_j^{i+1} \quad \forall l, a
\]
Step 3: Reduce the number of $\alpha-$vectors from $\Lambda^t$ by using the following reduction technique:

"If all the elements of the $\alpha-$vector are greater than or equal to the corresponding elements of any other $\alpha-$vector, say $\alpha'$, then $\alpha'$ is eliminated from the list of vectors which also implies that the dominated vector cannot be optimal " [29] (1991).

Step 4: Select an arbitrary state $\pi(t)$, and put it into the $\pi-$search table at time $t$ with an unmarked attribute.

Step 5: Pick an unmarked $\pi(t)$ from the $\pi-$search table at time $t$, and mark it. Denote this $\pi(t)$ as $\pi^0(t)$.

Step 6: With this selected state $\pi^0(t)$, determine $\alpha^*(t)\in\Lambda^t$ such that:

$$\pi^0(t)\alpha^*(t) \geq \pi^0(t)\alpha'(t) \quad \forall \alpha'\in\Lambda^t$$

Step 7: Use the following inequalities to identify the region of the information vector space over which $\alpha^*(t)$ is optimal:

$$\pi(t)[\alpha^*(t) - \alpha'(t)] \geq 0 \quad \forall \alpha'(t)\in\Lambda^t$$

$$\sum_{i=1}^{N_i} \pi_i(t) = 1 \quad \pi_i(t) \geq 0$$

(4.8)

All vertices of these generated region(s) are determined and added with an unmarked attribute to the $\pi-$search table at time $t$ for later examination.

Step 8 if $\alpha^*(t) \notin \Lambda^t$, place $\alpha^*(t)$ in $\Lambda^t$. 
**CHAPTER 4. POMDP EXISTING SOLUTION TECHNIQUES**

**Step 9** Return to Step 5 if $\pi$—search table has $\pi$—values with an unmarked attribute. Otherwise go to Step 1.

**Step 10** Terminate. $\Lambda'$ possesses the $\alpha$—vectors for all the regions at time $t$ ($0 \leq t \leq T$).

Due to memory limits, this algorithm is sometimes unable to solve problems. This is because the number of $\alpha$—vectors generated initially is large, and the reduction technique does not have the ability to substantially reduce the number of $\alpha$—vectors.

The computer code for this algorithm was developed in this study by using the technical computing environment MATLAB.

**4.5 MPAO* ALGORITHM**

This algorithm, developed by Lark [24](1990), represents POMDP's as an AND/OR tree, and uses the theory of Heuristic Search in graphs to solve them. Heuristic Search is a member of "Branch and Bound " procedures, which successively reduce the size of the set of potential solutions until the set of solutions is found. The reductions are achieved by using dominance relations on the values of the objective function that is to be optimized.

We now provide some information about AND/OR graphs that is required to understand the algorithm.

An AND/OR graph $G$ is a directed graph with a special node $s$, called the start (or root) node, and a nonempty set of terminal leaf nodes. AND/OR graphs are frequently used to represent sequential decision problems which can be decomposed
into subproblems. We restrict ourselves to AND/OR graphs that are loopfree, i.e., those which do not have any directed paths that begin and end at the same node. Let $N = \{n_i : i = 1, 2, ..., m\}$ be the finite set of nodes in $G$. These nodes are of two types: OR and AND.

**OR nodes** correspond to points at which decisions are made (decision nodes).

**AND nodes** correspond to points at which signals are received.

Let $L \subseteq N \times N$ represent the set of direct arcs; the element $(n_i, n_j)$ is the direct arc from node $n_i$ to node $n_j$ in $G$. Each direct arc $(n_i, n_j)$ in $G$ has a finite arc cost $c(n_i, n_j) \geq 0$.

The following are some relations between nodes:

1. If $(n_i, n_j)$ is the direct arc from node $n_i$ to node $n_j$ in $G$, $n_j$ is a successor of $n_i$, and $n_i$ is a parent of $n_j$.

2. If there is a path from node $n_i$ to node $n_k$, we say that $n_i$ is an ancestor of $n_k$, and $n_k$ is a descendant of $n_i$.

3. If $SCS : N \rightarrow 2^N$ is the successor set function, where:

$$SCS(n) = \{n' \in N : (n, n') \in L\}$$

We can refer to $\Gamma \subseteq N$ as a set of goal nodes $n$ of $N$ if $SCS(n) = \emptyset$. 
4. If there exists a node $s \in N$ such that $s$ has no parents, and such that every node $n_i \neq s$ is a descendant of $s$, we say that $s$ is the start node of $N$.

Consider graph $G = \{N, L, s, \Gamma\}$ to be the graph with node set $N$, arc set $L$, start node $s$, and goal node $\Gamma$.

Let $G' = (N', L', m, \Gamma')$ be a subgraph of $G$.

We say that $G'$ is a basic subgraph of $G$ if $G'$ satisfies the following properties:

1. $s \in N'$ ($m = s$)

2. If $n$ is an AND node in $G$, and $n \in N'$ then all the immediate successors of $n$ in $G$ are in $G'$ ($SCS(n) \subseteq N'$)

3. If $n$ is an OR node in $G$, and $n \in N'$. then exactly one of the immediate successors of $n$ in $G$ is in $G'$ ($SCS(n) \subseteq N'$).

We say that $G'$ is a solution subgraph of $G$ if $G'$ satisfies the following properties:

1. $s \in N'$ ($m = s$)

2. If $n$ is an AND node in $G$, and $n \in N'$, then all the immediate successors of $n$ in $G$ are in $G'$ ($SCS(n) \subseteq N'$)

3. If $n$ is an OR node in $G$, and $n \in N'$, then exactly one of the immediate successors of $n$ in $G$ is in $G'$ ($SCS(n) \subseteq N'$).
We define the cost of $G'$ as:

$$C(G') = \sum [c(n, n') : (n, n') \in L'] + \sum [c_{\Gamma}(n) : n \in \Gamma']$$

This algorithm for the minimization case works as follows:

**Step 0 (Initialization)**

- Select a set of states of knowledge \{\pi_1, \pi_2, ..., \pi_K\}. Each \pi_k \in \Pi, k = 1, ..., K.

- Set:
  
  $OPEN = G_s, COPEN = C(G_s), PREF = \phi, CPREF = \phi$.

  where:

  $G_s = N_s, L_s, s, \Gamma_s$, where $N_s = \{s\}, L_s = \emptyset$, and $\Gamma_s = \{s\} \cap \Gamma$.

  Note that $G_s$ is a solution base of $G$.

- Set $k = 0$.

**Step 1**

1. Set $k = k + 1$; if $k > K$ go to Step 1-3.

2. If $OPEN = \phi$ terminate. Otherwise, determine if there is a basic subgraph $G' \in OPEN$ such that:
\[ \pi_k C(G') \leq \pi_k C(G^*) \quad \forall G^* \in OPEN \cup PREF \]

If not, go to Step 1-1; Otherwise select \( G' \) for expansion, and go to Step 2.

3. If \( OPEN = \emptyset \), terminate; Otherwise choose \( G' \in OPEN \), and determine if there is a \( \pi^* \in \Pi \) such that:

\[ \pi^* C(G') \leq \pi^* C(G^*) \quad \forall G^* \in PRE F \]

If not, discard \( G' \) and \( C(G') \) from \( OPEN \) and \( COPEN \) respectively, and repeat Step 1-3. Otherwise go to Step 1-4.

4. Set \( \pi_k = \pi^* \).

If \( G' \) is a solution subgraph, place \( G' \) and \( C(G') \) in \( PRE F \) and \( CPREF \), and go to Step 1-2. Otherwise go to Step 2.

Step 2

If \( G' \) is a solution subgraph, go to Step 3. Otherwise:

- Remove \( G' \) from \( OPEN \) and \( C(G') \) from \( COPEN \).

- Choose a node \( \eta \) from \( G' \), where, \( \eta \) has no successors and, \( \eta \cap \Gamma = \emptyset \).

- Generate \( SCS_\eta(G') \), where \( SCS_\eta(G') \) represents the set of basic subgraphs obtained by the expansion of the immediate successors of \( \eta \).

- Place each element of \( SCS_\eta(G') \) in \( OPEN \).
· Calculate $C(G'')$ for each $G'' \in SCS_{\eta}(G')$.

· Place each $C(G'')$ in $COPEN$.

· Go to Step 1-2.

**Step 3**

Discard $G'$ and $C(G')$ from $OPEN$ and $COPEN$, respectively. Place $G'$ in $PREF$ and $C(G')$ in $CPREF$. Go to Step 1-2.

In the application of $MPAO^*$ to POMDPs, the solution of the associated MDP is calculated by means of dynamic programming at the beginning of the procedure. The data are stored, and then generate $SCS(G_s)$.

In this algorithm, we note that the condition presented in Step 1-3:

$$\pi^* C(G') \leq \pi^* C(G) \forall G \in PREF$$

can be used to eliminate $G'$ from further consideration only if $G'$ is a solution subgraph. Thus, $MPAO^*$ may have to generate and evaluate a large fraction of the total number of solution subgraphs before any substantial reduction of $OPEN$ can take place and before the algorithm finds a solution.

### 4.6 LINEAR SUPPORT ALGORITHM

This algorithm was developed by Cheng [10]. The original motivation for the linear support algorithm was to develop an algorithm which does not require complicated constraint sets.
Besides having simpler constraint sets, the linear support algorithm also has a special property which makes it more attractive. If a large number of \( \alpha \)'s are required to accurately describe the cost function, then computing these \( \alpha \)'s and cost function is usually very time-consuming, regardless of which algorithm is used. In this case, an approximate solution might be tolerable if the maximal difference between the exact solution and the approximate solution is less than a given error. However, none of the algorithms discussed so far can be modified to find this kind of solution. The Linear Support algorithm can find an approximate cost that is less than a given error.

This algorithm can be summarized as follows:

**Step 0:** Set \( t = T \).

Define \( \Lambda^T = \gamma(T) \).

**Step 1:** If \( t = 0 \) go to Step 8. Otherwise set \( t = t - 1 \).

Initialize \( \Lambda^t \) as an empty set.

Put each of the extreme points of the state space \( \Pi \) into \( \omega \) and \( \upsilon \).

Find the supports (\( \alpha \)'s) and the cost \( C^t(\pi) \) for each \( \pi \in \omega \). Put the newly generated \( \alpha \)'s into the set \( \Lambda^t \).

Determine the regions for each \( \alpha \in \Lambda^t \) and find all vertices of these regions.

**Step 2:** Put the generated vertices into \( \zeta \).

If a vertex is in \( \zeta \) but not in \( \omega \), put it in \( \omega \).

Find \( C^t(\pi) \) for each \( \pi \in \zeta \).

Empty \( \zeta \).
Step 3: Compute:

\[ G(\pi) = \min_{\alpha \in \Lambda^t} \{ \pi \alpha \} - C^t(\pi) \quad \forall \pi \in \omega \backslash v \]

If \( G(\pi) \) value is less than a given error, then put \( \pi \) into \( v \).

Step 4: If all \( G(\pi) \) are less than the tolerable error, go to Step 1.

Step 5: Pick a vertex \( \pi \) with largest value \( G(\pi) \) from \( \omega \), and denote this vertex \( \bar{\pi} \).

Step 6: Find the \( \alpha' \)'s for \( \bar{\pi} \), and put them into \( \Lambda^t_{\bar{\pi}} \).

Put the \( \alpha' \)'s in \( \Lambda^t_{\bar{\pi}} \) which are not already in \( \Lambda^t \) into \( \Lambda^t \).

Find all the new vertices of these newly generated regions, and put them into \( \zeta \).

If \( \bar{\pi} \) is not a vertex in \( \zeta \), delete \( \bar{\pi} \) from \( \omega \); otherwise, put \( \bar{\pi} \) into \( v \).

Step 7 Compute:

\[ G'(\pi) = \min_{\alpha \in \Lambda^t_{\bar{\pi}}} \{ \pi \alpha \} - G(\pi) \quad \forall \pi \in \omega \backslash v \]

If \( G'(\pi) < G(\pi) \) and \( \pi \notin \zeta \), then delete \( \pi \) from \( \omega \).

Return to Step 2.

Step 8 Terminate. \( \Lambda^t \) possesses the \( \alpha \)-vectors for all the regions at time \( t \) (0 \( \leq t \leq T \)).

The computer code for this algorithm was developed in this study by using the technical computing environment MATLAB.
This chapter was developed to present a general idea about some well-known algorithms associated with the POMDP. The computer codes for these algorithms are used in chapter 6 to compare their performance with the performance of our proposed heuristic, which is developed in chapter 5.
Chapter 5

POMDP PROPOSED HEURISTIC

Currently, exact algorithms for general POMDPs are intractable for all but the smallest problems, so that any algorithmic solution will be an approximation. Although some of the approximate algorithms reviewed previously expand the class of problems that can be solved, none are capable of solving truly large problems [27]. The margin of feasibility will be found at problem sizes with 20 to 30 states of the core process, and problems with 50 states are currently infeasible. It is highly likely that many real world applications will be as large or larger, so that solving to a guaranteed accuracy is unrealistic. We remark that research by Papadimitriou and Tsitsiklis [31] (1987) concerning the computational complexity of POMDPs is not encouraging. They report that the general POMDP is a PSPACE-hard problem. The class NP is contained in PSPACE.

The fundamental problem in developing algorithmic solutions for POMDP's is that
\( \Pi \) is uncountably infinite. All feasible algorithms involve reducing \( \Pi \) to either a finite set of points or a finite set of partitions White [58](1994). Partitions of \( \Pi \) are computationally demanding, and suffer the problem called "the curse of dimensionality". That is, as we increase the number of actions, observations, and stages of the problem we have modelled as a POMDP, the number of partitions we must consider can grow exponentially rapidly. Fixed grid methods are natural and often use approximation methods in dynamic programming, but the challenge is to relate the approximate results to the original problem in some meaningful way. Adjusted grid methods provide flexibility which comes at the cost of increased algorithm complexity, White[58](1994).

Lark [24](1990) presents the following illustrative example of the curse of dimensionality problem by using Monahan's method:

Let:
\[
\#A_t = K \geq 2 \quad t = 0, \ldots, T - 1.
\]
\[
\#N_z(t) = M \geq 2 \quad t = 1, \ldots, T
\]
\( T \geq 1 \)
\( \Lambda^t = \{ \alpha^t : \alpha^t = \gamma^a + \sum_{z=1}^{N_z} P^n R^n \alpha^{t+1} \} \)

Monahan showed (see section 4.2) that the cardinality of \( \Lambda^0 \) is:
\[
\#\Lambda^0 = K^{\frac{(M^T - 1)}{(M - 1)}}
\]

Thus, if \( K = M = 2 \) and \( T = 4 \), the number of \( \alpha \)'s we must consider is \( 2^{15} \). If we can evaluate \( 10^6 \) \( \alpha \)'s per second, then we shall solve this problem in 0.033 seconds.

The consequences of increasing one parameter (\( K, M, \) or \( T \)) while keeping the other parameters fixed are: If \( K = 3, M = 2, \) and \( T = 4 \), the number of \( \alpha \)'s in-
increases to $3^{15}$; we can solve this problem in roughly 14.4 seconds. If $T = 5$, $K = 2$, and $M = 2$, the number of $\alpha$'s increases to $2^{31}$; the time needed to solve this problem increases to roughly 36 minutes. If $M = 3$, $K = 2$, and $T = 4$, the number of $\alpha$'s increases to $2^{40}$; the time needed to solve this problem increases to roughly 12.7 days.

Hence, the "curse" limits the applicability of current algorithms to problems with small number of states, observations, actions, and stages. Therefore, another approach is warranted.

Our interest in the finite-horizon case is due to the fact that many of the problems we wish to model as a POMDP have a clearly defined finite horizon. For example, machine maintenance problems (such the one we discuss in chapter 7) frequently incorporate specifications such as finite-length mandated replacement rules or warranties.

The algorithmic tool chosen in this research is dynamic programming ( introduced by Bellman (1957) and Howard (1960)). The advantages of this approach will be expanded on in chapter 6, where we compare the performance of various solution procedures for the POMDP. We select dynamic programming because it leads to a globally optimal solution breaking up a large problem into a series of smaller, more tractable problems. Dynamic programming is much more efficient than explicit enumeration of the total reward associated with each possible set of decisions that may be chosen during the $T$ stages. Unfortunately, however, many practical applications of dynamic programming involve very large state spaces, and in these situations considerable computational effort is required to determine optimal decisions. To avoid this problem, our algorithm makes some assumptions that are shown later, in the
Our heuristic algorithm looks for a good solution instead of an optimum one which may be found only after a considerable computational effort, or more probably, never found at all. This heuristic possesses the following advantages:

- The heuristic compares favourably for speed of computation and is competitive with other algorithms on expected reward. These advantages come from a simple parametric representation of the reward function, and an efficient update for such representation.

- There is at present a significant discrepancy between the potential and actual applied results of POMDPs. This is at least partially explained by the fact that, while the theoretical foundation is largely in place, algorithmic solutions for realistically sized problems are rare. This heuristic would give practitioners the confidence to embark upon POMDP modelling exercises with the knowledge that some solutions will be forthcoming, and this in turn would encourage data collection exercises appropriate for more applications.

- The heuristic is simple to understand and implement. This is translated into more efficient code development and debugging, and higher confidence in the results.

- The heuristic is capable of providing a stable and relatively accurate solution (see Chapter 6).
5. The heuristic performs better as the quality of the information improves (see Chapter 6).

The disadvantage comes from the solution technique, which is based on dynamic programming and inherits its computational difficulties.

5.1 PROBLEM DEFINITION

Consider the following situation facing a decision maker. The decision maker wishes to optimally control the operation of a process, which behaves as a Markov process, even though he cannot directly observe the state of the process.

At each stage \( t \), where \( t = 0, 1, \ldots, T \), and \( 1 \leq T < \infty \), the process under consideration can be in one of a finite number of states \( 1, 2, \ldots, N_s \). The set of states in which the process can be at stage \( t \) is called the state space at stage \( t \), and is denoted as \( X(t) \). We shall represent the state of the process at stage \( t \) by \( s(t) \).

At each stage \( t \), \( t = 0, \ldots, T - 1 \), the decision maker obtains imprecise information about the current state \( s(t) \) from an observation process. Specifically, assume that at stage \( t \), \( t = 1, \ldots, T \), the decision maker observes a signal \( z(t) \), and chooses an action \( a_t \) from a finite set of actions \( A_t \).

\( z(t) \) is related to the unknown \( s(t) \) by the probability:
\[ r^a_{iz} = pr(z(t) = z| s(t) = i, a_{t-1} = a) \]

The observation \( z(t) \) belongs to a given observation space \( Z(t) \).

Let action \( a_t \) be the action chosen by the decision maker at stage \( t \). If the process is in state \( i \) at stage \( t \), a reward \( \gamma^a_i(t) \) is incurred, and the state of the process changes from \( i \) to \( j \) at stage \( t + 1 \) with probability \( p^a_{ij}(t) \). If the process is in state \( i \) at stage \( T \), then terminal reward \( \gamma_i(T) \) is incurred.

The initial state \( s(0) \) is random and characterized by a given probability measure \( \pi(0) \), where:

\[
\pi(0) = [\pi_i(0)] = [pr(s(0) = i|\xi(0))] 
\]

We assume the problem is to choose the alternatives to maximize the expected reward of operating the process for \( T \) time periods with termination reward \( \gamma(T) \). This selection is a function of the history of the process \( \xi(t) \). As a result, \( a_t = f[\xi(t)] \).

This history consists of a record for each time period that the process has been in operation, of the observed output and the selected alternative, and of the initial probability distribution over the states of the core process.

For \( t = 0 \), let alternative \( a_0 \) be chosen on the basis of datum \( \xi(0) = [\pi(0)] \). That is, \( a_0 = f[\xi(0)] \)

For \( t = 1, ..., T \), define:

\[
\xi(t) = [\pi(0), z(0), a_0, z(1), a_1, ..., z(t-1), a_{t-1}, z(t)]
\]
For $t = 0, ..., T - 1$, let alternative $a_t$ be chosen on the basis of $\xi(t)$. That is, $a_t = f[\xi(t)]$.

A policy at stage $t$, $t = 0, ..., T - 1$, is a function $\delta_t : \{\xi(t)\} \rightarrow a_t$; a strategy is an ordered sequence of policies $\delta_0, ..., \delta_{T-1}$.

In order to find nondominated strategies, we must determine policies at each stage $t$. The domain of policy $\delta_t$ is $\xi(t)$. There are $\#N_s(0)\#A_0\#N_z(1)\#A_{t-1}\#N_z(t)$ elements in $\xi(t)$ for $t \geq 1$; For example, if $T = 10$ and $\#N_s(0) = \#A_0 = \#N_z(1) = \#A_{t-1} = \#N_z(t) = 2$, $\#\xi(10) = 2^{21}$. Hence, given a data sequence $\xi(t)$ we wish to find a way of condensing the information in $\xi(t)$ in a way that allows us to calculate and present the policies more easily. More formally, we wish to find for each $\xi(t)$ a way of representing $\xi(t)$ by a smaller set of numbers which contains all of the information necessary for calculating policies which comprise nondominated strategies. Such sets of numbers are called sufficient statistics for $\xi(t)$.

For each $t \in (0, 1, ..., T)$ and each $i \in N_s(t)$, define:

$$\pi(t) = [\pi_i(t)] = [pr(s(t) = i|\xi(t))]$$

The vector $\pi(t)$ is referred to as the information vector at stage $t$, and $(\pi(t), t = 0, 1, ..., T)$ is referred to as the information process.

In publications by Aoki [2], Astrom [4], Bertsekas [7], and Sondik [43], it is shown that the information process $\pi(t)$ is a sufficient statistic for the POMDP. Thus, we can reformulate the POMDP as a completely observed MDP with space $II$, which is uncountably infinite. The state process for this equivalent completely observed MDP is the information process $(\pi(t), t = 0, 1, ..., T)$. As a result, for a given POMDP, we shall refer to its associated MDP. The associated MDP for a POMDP is the MDP which shares the cost structure and probabilistic dynamics of the POMDP, and has
perfect state information at each stage. Following an action at time \( t \), the decision maker observes an outcome which imparts some imperfect knowledge regarding the underlying state of the system. Using Bayes’ rule, the decision maker can then revise his probabilities concerning the possible underlying state. His next decision at time \( t + 1 \) would then be based on his new updated beliefs \( (a_{t+1} = f[\pi(t + 1)]) \). This is the key to the transformation of a POMDP to a MDP. If these updated probabilities associated with each possible system state are used as the state descriptor, the resulting problem has the same form as an MDP and can, theoretically, be solved with the same algorithms.

To update probabilities, for each \( t + 1 \in (1, ..., T) \) and each \( j \in N_s(t + 1) \), define:

\[
pr\{s(t + 1) = j|\xi(t + 1)\} = T_j(\pi(t)|z,a)
\]

Then:

\[
T_j(\pi(t)|z,a) = \frac{\sum_{i=1}^{N_s} \pi_i(t)p_{ij}^a r_{ij}^a}{\sum_{i=1}^{N_s} \sum_{k=1}^{N_z} \pi_i(t)p_{ik}^a r_{ik}^a}
\] (5.1)

Proof

\[
T_j(\pi(t)|z,a) = \frac{pr(s(t + 1) = j, z(t + 1) = z|\xi(t), a_t = a)}{pr(z(t + 1) = z|\xi(t), a_t = a)}
\]

\[
T_j(\pi(t)|z,a) = [pr(z(t + 1) = z|\xi(t), a_t = a)]^{-1} \left[ \sum_{i=1}^{N_s} pr\{s(t) = i, s(t + 1) = j, z(t + 1) = z|\xi(t), a_t = a\} \right]
\]

\[
T_j(\pi(t)|z,a) = [pr(z(t + 1) = z|\xi(t), a_t = a)]^{-1} \left[ \sum_{i=1}^{N_s} pr\{s(t) = i|\xi(t), a_t = a\} \right]
\]
\begin{align*}
pr\{s(t + 1) = j|s(t) = i, \xi(t), a_t = a\} \\
pr\{z(t + 1) = z|s(t + 1) = j, s(t) = i, \xi(t), a_t = a\}
\end{align*}

\[T_j(\pi(t)|z, a) = \left[pr(z(t + 1) = z|\xi(t), a_t = a)\right]^{-1}\sum_{i=1}^{N_z} \pi_i(t)p_{ij}^a r_{jz}^a\]

\[T_j(\pi(t)|z, a) = \frac{\sum_{i=1}^{N_z} \pi_i(t)p_{ij}^a r_{jz}^a}{\sum_{i=1}^{N_z} \sum_{k=1}^{N_z} \pi_i(t)p_{ik}^a r_{kz}^a}\] (5.2)

The row vector of these probabilities is frequently represented as \(T(\pi(t)|z, a) = [T_j(\pi(t)|z, a)]\).

The use of \(T\) emphasizes that \(T(\pi(t)|z, a)\) is a transformation of one point of \(\pi(t)\) into another.

We can write the above probability in matrix form as follows:

\[T(\pi(t)|z, a) = \frac{\pi(t)p^a R_z^a}{\pi(t)p^a R_z^a 1}\] (5.3)

Where:

\(R_z^a\) is a diagonal matrix with \(r_{jz}^a\) as its diagonal elements. That is:

\[
R_z^a = \begin{bmatrix}
r_{1z}^a & \cdots & \cdots & \cdots \\
\vdots & r_{2z}^a & \cdots & \cdots \\
\vdots & \vdots & \ddots & \vdots \\
\vdots & \vdots & \cdots & a_{N_z}\n\end{bmatrix}
\]

\(1\) is an \(N_z\) dimensional column vector with all elements being 1.
$P^a$ is an $N_s \times N_s$ transition matrix. $P^a = [p^a_{ij}]$, where $p^a_{ij}$ is the probability that the state of the core process at time $t+1$ will be $j$ given that the state at time $t$ is $i$ and the decision made at time $t$ is $a$.

To formulate the problem in terms of $\pi$, we define $C^t(\pi)$ (maximization case) as follows:

$$C^T(\pi) = \pi(T)\gamma(T)$$

$$C^t(\pi) = \max_{a \in A} [\pi(t)\gamma^a + \sum_{i=1}^{N_s} \sum_{j=1}^{N_s} \sum_{z=1}^{N_s} \pi_i(t)p^a_{ij}r^a_{iz}C^{t+1}(T(\pi|z,a))] , t < T$$

This equation establishes that the expected reward from time $t$ to the end of the decision horizon is equal to the expected immediate reward of making one transition ($\pi(t)\gamma$) plus the expectation over $z$ of the expected reward from time $t+1$ to the end.

As we have already noted, in this representation, the state variable is known exactly; hence, the above equation is the recursive equation of a completely observable Markov Decision Process. In other words, with this equation, we convert the POMDP into an equivalent (completely observable) Markov Decision Process. The associated MDP for a POMDP is the MDP which shares the cost structure and probabilistic dynamics of the POMDP, and which has perfect state information at each stage.

### 5.2 DESCRIPTION OF THE HEURISTIC

This heuristic is composed of two main parts:
1. Actuator part.

2. Estimator part.

The actuator part allows us to generate an optimal control policy to the system as a function of the $\pi(t)$.

The estimator part is simply the generation of the probability distribution $\pi(t)$ at time $t$. $\forall t \leq T$. The value of $\pi(t)$, which is a function of the observations must obviously be calculated in real time as the outputs are observed. The estimator part uses the following information:

1. $z(t) = z$

2. $a_{t-1} = a$

3. $\pi(t - 1)$

The decomposition of the heuristic algorithm into estimator and actuator parts significantly reduces the amount of real time computation required in the implementation of the optimal controller. Functions used in both parts contain parameters which can be precomputed from the a priori data of the problem and stored in tables.

The Actuator Part

The procedures used in this part run off-line. As we said before, optimal controls based on $\pi$ eliminate the need for storing past histories of varying lengths. Using this fact and the standard "Principle of Optimality" of dynamic programming, we arrive at the following reward equation (see chapter
To avoid the "curse of dimensionality" problem described in the introduction of this chapter, this algorithm assumes that although the state of the core process is not known at time $t$, the state of the core process is known with certainty after time $t$, that is, at times $t+1, ..., T$. Under this assumption, the Actuator part uses Dynamic Programming to obtain the maximum expected reward that can be made during the time horizon, with the following recursion equation that relates the reward earned during stages $t, t+1, ..., T$ to the reward earned from stages $t+1, t+2, ..., T$, under the assumption that the core process state at these times is known.

Let:

$CT_i^T$ represents the optimal total expected reward obtained from time $t$ to the end of the decision process, when $s(t) = i$, and the process is observed from time $(t + 1)$ to the end, $\forall i, t$.

$CT^t$ represents a $N_s$-dimensional column vector $CT^t = [CT_i^t]$.

\[
CT_i^T = \gamma_i(T) \\
CT_i^t = \max_{a \in A} [\gamma_i^a + \sum_{j=1}^{N_s} p^a_{ij} CT_j^{t+1}] \quad t < T
\]
Equation [5.6] was obtained from equation [5.5] under the assumption that the \( s(t + 1) \) is known. It states that the total expected reward obtained by operating the process from time \( t \) to the end of the decision process equals the expected immediate reward of making one transition plus the optimal expected reward from time \( t + 1 \) to the end.

Dynamic programming is used to solve the problem represented by the equation 5.6. The tables obtained per each decision period are stored to be used after the row vector \( \pi(t) \) is calculated (Step 2 of the Estimator part).

The Actuator part of the heuristic has three main advantages:

1. Given that the procedures of this part are running off-line, the Actuator part significantly reduces the amount of real time computations required in the implementation of the optimal controller.

2. It assures an efficient On-Line implementation, involving polynomial computational time.

3. It guarantees that memory requirements do not exceed limits.

The Estimator Part

"The estimator portion of the optimal controller is an optimal solution of the problem of estimating the state \( s(t) \) assuming no control takes place" Bertsekas [7].

This heuristic can be defined as an On-Line heuristic because the essence of this solution procedure is developed in the part which is running in parallel with the process. The results from the Actuator Part are used in Step 2 to
select the policy with greatest expected reward of operating the process from time $t$ to the end.

The estimator part includes the following operations:

**Step 1**

Set $t = 0$.

Time $t = 0$ is always associated with an initial distribution of the state space $\pi(0) = [\pi_1(0), ..., \pi_N(0)]$, and an optimal policy $\delta_0$, which represents the policy used at the initialization of the process.

The expected immediate reward of the next transition is determined as:

$$\pi(0) \gamma^{\delta_0}$$

**Step 2**

$t = t + 1$

If $t = T$ go to Step 4, otherwise:

Observe $z(t) = z$ ( $z(t)$ is observed according to $s(t)$ and $R$).

Compute the row vector $\pi(t)$ from:

$$\pi(t) = \frac{\pi(t-1)P^\delta_{t-1}R^\delta_{t-1}}{\pi(t-1)P^\delta_{t-1}R^\delta_{t-1}z}$$

Where:

$\pi_i(t)$ represents the probability that the process is in state $i$ at time $t$ given all the information up to time $t$ about the process.

$z$ represents the output observed for the observation process at time $t$. 
\( \delta_t \) represents the policy with greatest expected reward of operating the process from time \( t \) to the end, that is:

\[
delta_t = \{a^* \in A_t | \pi(t)CT^t(a^*) \geq \pi(t)CT^t(a) \quad \forall a \in A_t \}
\]

Where:

\( CT^t_i(a) \) represents the total reward obtained from time \( t \) to the end of the decision process, when \( s(t) = i \), and \( a_t = a \). Its value is calculated in the Actuator part.

\( CT^t(a) \) represents a \( N_s \)-dimensional column vector \([CT^t_i(a)]\).

**Step 3**

The expected immediate reward of the next transition is determined as:

\[
\pi(t)\gamma^{\delta_t}
\]

Go to Step 2.

**Step 4**

Compute the row vector \( \pi(T) \) from:

\[
\pi(T) = \frac{\pi(T - 1)P^{\delta_T - 1}R^{\delta_T - 1}}{\pi(T - 1)P^{\delta_T - 1}R^{\delta_T - 1}1}
\]

The expected final reward is determined as:

\[
\pi(T)\gamma(T)
\]

Note: The change in the state of the core process from \( s(t - 1) \) to \( s(t) \) is assumed to occur instantaneously at the border between stage \( t - 1 \) and \( t \).
The computer code for this algorithm was developed in this study by using the technical computing environment MATLAB. The complete estimator part is presented in figure 5.1 in schematic form.

5.3 BOUNDS ON THE SOLUTION

These bounds on the solution are obtained by changing the amount of information which is available for the choice of the optimal policies. The following two cases are considered:

1. The information about the state of the core process \( s(t) \) \( \forall t \) is precise.

2. The information from the observation process \( z(t) \) does not provide information about the state of the core process \( s(t) \).

The results will allow us to provide a value to the information that is available at the time of the decisions.

5.3.1 PRECISE INFORMATION ABOUT THE STATE OF THE CORE PROCESS

In this case \( z(t) = s(t) \) \( \forall t \). Hence:

\[
R^a = I \quad \forall a
\]

Let \( z(t+1) = m \), then

\[
T(\pi(t)|m, a) = \frac{\pi(t)P_m^a}{\pi(t)P_m^a \cdot 1} \quad \text{(5.7)}
\]
CHAPTER 5. POMDP PROPOSED HEURISTIC

Initialization of the process
\[ t = 0 \]
\[ \pi(0) = [\pi_1(0), \ldots, \pi_{N_z}(0)] \]
\[ \delta_0 \]

Expected immediate reward of the next transition
\[ \pi(0) \gamma^{\delta_0} \]

\[ t = t + 1 \]

Observe \( z(t) = z \)

\[ t = T \]

YES

\[ \pi(t) = \frac{\pi(t+1) P^{\delta_t-1} C^{t-1}}{\pi(t-1) P^{\delta_t-1} C^{t-1}} \]

\[ \delta_t = \{ a \in A( t ) | \pi(t) C T^t(a) \geq \pi(t) C T^t(a) \} \]

\[ \forall a \in A( t ) \]

The expected reward of the next transition is:
\[ \pi(t) \gamma^{\delta_t} \]

NO

Compute the row vector:
\[ \pi(T) = \frac{\pi(T-1) P^{\delta_{T-1}} R^{\delta_{T-1}}}{\pi(T-1) P^{\delta_{T-1}} R_{Z}^{\delta_{T-1}}} \]

The expected reward is:
\[ \pi(T) \gamma(T) \]

Figure 5.1: THE ESTIMATOR PART (SCHEMATIC)
CHAPTER 5. POMDP PROPOSED HEURISTIC

Where:

\( \mathbf{1} \) is an \( N_s \) dimensional column vector with all elements being 1.

\( P_m^a \) is an \( N_s \times N_s \) matrix with all the elements equal to zero except column \( m \), which is equal to column \( m \) in \( P^a \).

This estimation of \( \pi(t+1) \) is a weighted average of the values in the column \( m \) of the matrix \( P^a \).

Astrom [4] shows that the solution of the equation 5.5 is bounded from above by the solution of equation 5.5 in which the estimation of \( \pi(t) \) is given by equation 5.7.

When the proposed algorithm is used, the solution of the equation 5.6 is bounded from above by the solution of the equation 5.6 in which the estimation of \( \pi(t) \) is given by equation 5.7.

5.3.2 NO INFORMATION ABOUT THE STATE OF THE CORE PROCESS

When the observation process does not provide information about the state \( s(t) \) \( \forall t \), then \( R^a \) is a \#\( N_s \times 1 \) matrix, each coordinate of which equals 1 \( \forall a \), then:

\[
T(\pi(t)|z,a) = \frac{\pi(t)P^a}{\pi(t)P^a \mathbf{1}} \tag{5.8}
\]

In this estimation of \( \pi(t+1) \), the information in \( R^a \) is not considered.

Astrom [4] shows that the solution of the equation 5.5 is bounded from below by the solution of equation 5.5 in which the estimation of \( \pi(t) \) is given by equation 5.8.

When the proposed algorithm is used, the solution of the equation 5.6 is bounded.
from below by the solution of equation 5.6 in which the estimation of \( \pi(t) \) is given by equation 5.8.
Chapter 6

EVALUATION OF SOME POMDP SOLUTION TECHNIQUES

In this chapter, we compare the performance of the following POMDP solution techniques:

1. Smallwood and Sondik’s algorithm (SSA).

2. Relaxed Region algorithm (RRA).

3. Linear Support algorithm (LSA).
4. S. Mukherjee and K. Seth's algorithm (MSA).

5. Proposed heuristic algorithm (PHA).

This comparison is based on two types of numerical evaluations:

**CPU Time** This evaluation is performed because the main concern of this research is improving the speed of solution procedures for the POMDP. To accomplish this evaluation, all algorithms were implemented as MATLAB programs, and CPU time was divided in CPU time for the off-line computation and CPU time for the on-line computation. CPU time for the off-line computation represents the CPU time required to compute $\alpha-$vectors at period $t$ by using $\alpha-$vectors at period $(t + 1)$ $\forall t < T$. CPU time for the on-line computation represents the time to find a solution when the initial $\pi$ value is provided, and the observations for all the decision periods are simulated.

**Expected Reward** This evaluation is performed to determine the differences in terms of the Total Expected Reward between these algorithms.

Restrictions $2 \leq N_s \leq 8$, $2 \leq N_a \leq 6$, $2 \leq N_z \leq 5$ were imposed to avoid problems due to memory limits, and the problems are assumed to be homogeneous.

We set the number of decision periods $N_p$ equal to 20, and ran the problems. If all the algorithms could not solve the problems within a fixed CPU time of 2000 seconds, or if all of them were unable to solve the problem due to memory limits, we set $N_p = N_p - 1$. 
CHAPTER 6. EVALUATION OF SOME POMDP SOLUTION TECHNIQUES

The state of the observation process is simulated for all the algorithms by using the following formula:

\[ pr\{z(t + 1) = z|\xi(t)\} = \{z|\pi(t)\} = \sum_{i=1}^{N_s} \sum_{j=1}^{N_a} \pi_i(t)p_{ij}^t r_{jz}^t \]

The total expected rewards, when the existing algorithms are used, are calculated as follows:

1. Each algorithm is run to obtain the finite set of values of \( \alpha' \) \( \forall t \).
2. The \( \pi(t) \) at time \( t \) is calculated.
3. The optimal control at time \( t \) is computed, by finding an index \( j \) maximizing \( \pi(t)\alpha_j^t \) \( \forall j \), say \( j^* \).
4. The optimal control associated with \( \alpha_{j^*}' \) is determined.
5. The expected reward at time \( t \) is calculated as: \( \pi(t)\alpha_{j^*}' \).

To obtain the total expected reward when our proposed algorithm is used, we simulate each problem, run the simulation one hundred times, and obtain the total expected reward by averaging the 100 trials. Confidence intervals on the expected reward using these 100 values are derived for each problem. The significance level (\( \alpha \)) is assumed to be 0.05.

This simulations were implemented with MATLAB.
6.1 CLASSIFICATION OF THE TEST PROBLEMS

We shall separate our test problems into four basic groups:

1. The first group contains several sets of data without any special assumption about problem structure.

2. The second group consists of POMDP problems for which special structure is imposed upon the observation process. Specifically, we assume that for each action \( a \in A \), the matrices \( R^a \) have the following properties:

   (a) For every row in \( R^a \), we require one and only one element of the row to be within a specified tolerance \( \epsilon \) of 1.

   (b) For every column in \( R^a \), we require one and only one element of the column to be within a specified tolerance \( \epsilon \) of 1.

If \( \epsilon \) is small, the matrices \( R^a \) provide the decision maker with good information concerning the state of the process. Hence, such POMDPs can be called "close-to-perfect information" POMDPs.
3. The third group considers the special case of the completely unobserved Markov decision process (CUMDP). We shall model the CUMDP as a POMDP for which \( z(t) = \text{constant} = z \) for each \( t, t = 1, \ldots, T \).

4. The fourth group considers the case when the measurements are exact, that is, the measured output \( z(t) \) will coincide with the state of the core process \( s(t) \) with probability one. Hence:

\[
R^a = I \quad \forall \ a
\]

These groups are obtained by modifying the amount of data which is available for the choice of the optimal policies.

6.1.1 THE GENERAL POMDP - GROUP 1

This section evaluates some POMDP solution techniques with test problems that have been generated without any special assumptions about problem structure. The test problems generated to compare the efficiency of the algorithms are listed in Appendix A.

The Expected Rewards are calculated under the following considerations:

1. Initial \( \pi \) value is equal to \([1 \ 0 \ 0]\) for Data Set 1, Data Set 2, Data Set 3.

2. Initial \( \pi \) value is equal to \([1 \ 0 \ 0 \ 0]\) for Data Set 4, Data Set 5.

3. Initial \( \pi \) value is equal to \([1 \ 0 \ 0 \ 0 \ 0]\) for Data Set 6.
Table 6.1: The General POMDP - Data Sets 1-3

4. Number of decision periods for all problems is equal to 20.

Tables 6.1 and 6.2 summarize the results obtained by applying some existing algorithms to each set of data.

Table 6.3 shows the performance of our proposed algorithm for data sets 1-6. The Table includes the CPU time for the Actuator part; the CPU time for the Estimator part; the average, the standard deviation, and the 95% intervals of confidence of the expected reward from the 100 simulations.
<table>
<thead>
<tr>
<th>ALGORITHM</th>
<th>NUMBER OF REGIONS</th>
<th>OFF-LINE CPU TIME</th>
<th>ON-LINE CPU TIME</th>
<th>EXPECTED REWARD</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATA SET 4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SSA</td>
<td>102</td>
<td>1342</td>
<td>23.45</td>
<td>42.72</td>
</tr>
<tr>
<td>RRA</td>
<td>36</td>
<td>74.43</td>
<td>17.52</td>
<td>42.95</td>
</tr>
<tr>
<td>LSA</td>
<td>77</td>
<td>682.62</td>
<td>17.51</td>
<td>42.80</td>
</tr>
<tr>
<td>MSA</td>
<td>-</td>
<td>OUT</td>
<td>OF</td>
<td>MEMORY</td>
</tr>
<tr>
<td>DATA SET 5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SSA</td>
<td>103</td>
<td>1462.4</td>
<td>20.26</td>
<td>74.06</td>
</tr>
<tr>
<td>RRA</td>
<td>40</td>
<td>89.80</td>
<td>19.23</td>
<td>75.16</td>
</tr>
<tr>
<td>LSA</td>
<td>58</td>
<td>338.28</td>
<td>18.51</td>
<td>74.25</td>
</tr>
<tr>
<td>MSA</td>
<td>-</td>
<td>OUT</td>
<td>OF</td>
<td>MEMORY</td>
</tr>
<tr>
<td>DATA SET 6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SSA</td>
<td>76</td>
<td>418.86</td>
<td>21.03</td>
<td>74.43</td>
</tr>
<tr>
<td>RRA</td>
<td>36</td>
<td>44.21</td>
<td>18.46</td>
<td>78.27</td>
</tr>
<tr>
<td>LSA</td>
<td>-</td>
<td>CANNOT COMPLETE</td>
<td>OF</td>
<td>MEMORY</td>
</tr>
<tr>
<td>MSA</td>
<td>-</td>
<td>OUT</td>
<td>OF</td>
<td>MEMORY</td>
</tr>
</tbody>
</table>

Table 6.2: The General POMDP - Data Sets 4-6
6.1.2 THE CLOSE-TO-PERFECT INFORMATION PROBLEM - GROUP 2

We have chosen this class of problems for testing because we wish to know whether performance of our proposed heuristic algorithm improves as the quality of the information about the core process improves.

Data set 7 to data set 10 in the Appendix A correspond to this type of problem. Table 6.4 summarize the results obtained by applying some existing algorithms to each set of data. The Expected Rewards for each algorithm are calculated under the following considerations:

1. Initial \( \pi \) value is equal to \([1 0]\) for Data Set 7 and Data Set 8.

2. Initial \( \pi \) value is equal to \([1 0 0]\) for Data Set 9, Data Set 10.
3. Number of decision periods for all problems is equal to 20.

Table 6.5 shows the performance of our proposed algorithm for data sets 7-10. The Table includes the CPU time for the Actuator part; the CPU time for the Estimator part; the average, the standard deviation, and the 95% intervals of confidence of the expected reward from the 100 simulations.

6.1.3 THE CUMDP PROBLEMS - GROUP 3

This class consists of CUMDP where the problem data are generated without any special assumptions about problem structure. For each action in $A_t$, $t = 0, ..., T - 1$, the observation matrix $R^a$ is an $\#S_t \times 1$ matrix, each coordinate of which equals 1. Data Set 11 to Data Set 14 in the Appendix A correspond to this type of problem. Table 6.6 summarize the results obtained by applying some existing algorithm to each set of data. The Expected Rewards for each algorithm are calculated under the following considerations:

1. Initial $\pi$ value is equal to $[1 \ 0]$ for Data Set 11 and Data Set 14.

2. Initial $\pi$ value is equal to $[1 \ 0 \ 0 \ 0 \ 0]$ for Data Set 12.

3. Initial $\pi$ value is equal to $[1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0]$ for Data Set 13.

4. Number of decision periods for all problems is equal to 20.
### Table 6.4: Close to Perfect information POMDP - Data Sets 7-10

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Number of Regions</th>
<th>Off-Line CPU Time</th>
<th>On-Line CPU Time</th>
<th>Expected Reward</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATA SET 7</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SSA</td>
<td>38</td>
<td>16.75</td>
<td>30.26</td>
<td>58.25</td>
</tr>
<tr>
<td>RRA</td>
<td>25</td>
<td>4.56</td>
<td>20.11</td>
<td>63.75</td>
</tr>
<tr>
<td>LSA</td>
<td>38</td>
<td>19.61</td>
<td>21.21</td>
<td>58.25</td>
</tr>
<tr>
<td>MSA</td>
<td>25</td>
<td>6.1</td>
<td>13.78</td>
<td>63.67</td>
</tr>
<tr>
<td>DATA SET 8</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SSA</td>
<td>38</td>
<td>16.65</td>
<td>19.56</td>
<td>94.06</td>
</tr>
<tr>
<td>RRA</td>
<td>33</td>
<td>11.20</td>
<td>21.58</td>
<td>96.62</td>
</tr>
<tr>
<td>LSA</td>
<td>38</td>
<td>19.23</td>
<td>17.58</td>
<td>94.06</td>
</tr>
<tr>
<td>MSA</td>
<td>33</td>
<td>9.88</td>
<td>14.45</td>
<td>96.62</td>
</tr>
<tr>
<td>DATA SET 9</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SSA</td>
<td>68</td>
<td>226.67</td>
<td>20.66</td>
<td>73.60</td>
</tr>
<tr>
<td>RRA</td>
<td>22</td>
<td>8.13</td>
<td>17.08</td>
<td>90.96</td>
</tr>
<tr>
<td>LSA</td>
<td>56</td>
<td>175.65</td>
<td>15.22</td>
<td>78.41</td>
</tr>
<tr>
<td>MSA</td>
<td>31</td>
<td>86.56</td>
<td>14.28</td>
<td>88.28</td>
</tr>
<tr>
<td>DATA SET 10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SSA</td>
<td>114</td>
<td>1973.2</td>
<td>23.68</td>
<td>51.98</td>
</tr>
<tr>
<td>RRA</td>
<td>33</td>
<td>52.07</td>
<td>19.22</td>
<td>63.84</td>
</tr>
<tr>
<td>LSA</td>
<td>57</td>
<td>405.08</td>
<td>18.46</td>
<td>61.61</td>
</tr>
<tr>
<td>MSA</td>
<td>32</td>
<td>94.69</td>
<td>13.51</td>
<td>61.62</td>
</tr>
</tbody>
</table>
Table 6.5: Performance of PHA- "close to perfect information" POMDPs

<table>
<thead>
<tr>
<th>DATA SET NUMBER</th>
<th>OFF LINE CPU TIME</th>
<th>ON LINE CPU TIME</th>
<th>MEAN</th>
<th>STANDARD DEVIATION</th>
<th>95.0% C.I.</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>0.15</td>
<td>12.58</td>
<td>63.53</td>
<td>5.76</td>
<td>(62.40, 64.66)</td>
</tr>
<tr>
<td>8</td>
<td>0.18</td>
<td>12.89</td>
<td>96.39</td>
<td>13.06</td>
<td>(93.83, 98.95)</td>
</tr>
<tr>
<td>9</td>
<td>0.22</td>
<td>12.60</td>
<td>86.99</td>
<td>9.28</td>
<td>(85.17, 88.81)</td>
</tr>
<tr>
<td>10</td>
<td>0.44</td>
<td>12.91</td>
<td>60.45</td>
<td>14.18</td>
<td>(57.67, 63.23)</td>
</tr>
</tbody>
</table>

Table 6.7 shows the performance of our proposed algorithm for data sets 11-14. The Table includes the CPU time for the Actuator part; the CPU time for the Estimator part; the average, the standard deviation, and the 95% intervals of confidence of the expected reward from the 100 simulations.

6.1.4 THE PERFECT INFORMATION PROBLEM - GROUP

This section considers the particular case when the measurements are exact, i.e., the measured output \( z(t) \) will coincide with the state \( s(t) \) with probability one. Hence:

\[
R^a = I
\]

Data Set 15 to Data Set 18 in the Appendix A correspond to this type of problems. Table 6.8 summarize the results obtained by applying some existing algorithm to each Set of Data. The Expected Rewards for each algorithm are calculated under the following considerations:
## Table 6.6: The CUMDP - Data Sets 11-14

<table>
<thead>
<tr>
<th>ALGORITHM</th>
<th>NUMBER OF REGIONS</th>
<th>OFF-LINE CPU TIME</th>
<th>ON-LINE CPU TIME</th>
<th>EXPECTED REWARD</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATA SET 11</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SSA</td>
<td>23</td>
<td>23.18</td>
<td>17.00</td>
<td>150.69</td>
</tr>
<tr>
<td>RRA</td>
<td>12</td>
<td>15.33</td>
<td>12.93</td>
<td>151.81</td>
</tr>
<tr>
<td>LSA</td>
<td>18</td>
<td>22.45</td>
<td>13.11</td>
<td>155.41</td>
</tr>
<tr>
<td>MSA</td>
<td>16</td>
<td>12.12</td>
<td>12.22</td>
<td>159.30</td>
</tr>
<tr>
<td>DATA SET 12</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SSA</td>
<td>40</td>
<td>61.73</td>
<td>21.75</td>
<td>170.50</td>
</tr>
<tr>
<td>RRA</td>
<td>22</td>
<td>35.20</td>
<td>15.83</td>
<td>170.33</td>
</tr>
<tr>
<td>LSA</td>
<td>28</td>
<td>54.12</td>
<td>16.24</td>
<td>173.45</td>
</tr>
<tr>
<td>MSA</td>
<td>26</td>
<td>30.18</td>
<td>15.60</td>
<td>176.24</td>
</tr>
<tr>
<td>DATA SET 13</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SSA</td>
<td>38</td>
<td>56.19</td>
<td>19.44</td>
<td>158.89</td>
</tr>
<tr>
<td>RRA</td>
<td>19</td>
<td>21.11</td>
<td>17.37</td>
<td>167.22</td>
</tr>
<tr>
<td>LSA</td>
<td>22</td>
<td>44.17</td>
<td>19.44</td>
<td>164.38</td>
</tr>
<tr>
<td>MSA</td>
<td>20</td>
<td>20.33</td>
<td>17.37</td>
<td>169.43</td>
</tr>
<tr>
<td>DATA SET 14</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SSA</td>
<td>23</td>
<td>19.84</td>
<td>14.55</td>
<td>247.16</td>
</tr>
<tr>
<td>RRA</td>
<td>15</td>
<td>12.22</td>
<td>12.11</td>
<td>252.41</td>
</tr>
<tr>
<td>LSA</td>
<td>18</td>
<td>15.67</td>
<td>13.24</td>
<td>252.00</td>
</tr>
<tr>
<td>MSA</td>
<td>16</td>
<td>12.44</td>
<td>11.21</td>
<td>256.22</td>
</tr>
<tr>
<td>PHA</td>
<td>-</td>
<td>0.19</td>
<td>11.37</td>
<td>252.99</td>
</tr>
</tbody>
</table>
CHAPTER 6. EVALUATION OF SOME POMDP SOLUTION TECHNIQUES

Table 6.7: Performance of PHA- "CUMDP"

<table>
<thead>
<tr>
<th>DATA SET NUMBER</th>
<th>OFF LINE CPU TIME</th>
<th>ON LINE CPU TIME</th>
<th>MEAN</th>
<th>STANDARD DEVIATION</th>
<th>95.0 %C.I.</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>0.16</td>
<td>11.61</td>
<td>156.43</td>
<td>7.29</td>
<td>( 155 , 157.86 )</td>
</tr>
<tr>
<td>12</td>
<td>0.25</td>
<td>14.92</td>
<td>174.16</td>
<td>11.94</td>
<td>( 171.82 , 176.5 )</td>
</tr>
<tr>
<td>13</td>
<td>0.60</td>
<td>15.22</td>
<td>164.40</td>
<td>19.54</td>
<td>( 160.57 , 168.23 )</td>
</tr>
<tr>
<td>14</td>
<td>0.19</td>
<td>11.37</td>
<td>253.70</td>
<td>11.22</td>
<td>( 251.5 , 255.90 )</td>
</tr>
</tbody>
</table>

1. Initial $\pi$ value is equal to [1 0 0] for Data Set 15 and Data Set 16.

2. Initial $\pi$ value is equal to [1 0 0 0] for Data Set 17.

3. Initial $\pi$ value is equal to [1 0 0 0 0] for Data Set 18.

4. Number of decision periods for all problems is equal to 20.

Table 6.9 shows the performance of our proposed algorithm for data sets 15-18. The Table includes the CPU time for the Actuator part; the CPU time for the Estimator part; the average, the standard deviation, and the 95% intervals of confidence of the expected reward from the 100 simulations.

To determine whether the 100-values from the simulation of each set of data could have mean equal to the expected reward obtained with MSA, we perform hypotheses testing for the mean.

Table 6.10 shows the results of the hypotheses. $h$, the result, is 1 if we can reject the null hypothesis at the significance level $\alpha$ and 0 otherwise. $\alpha = 0.05$ by default. $p$, the $p$-value associated with the T-statistic, is the probability that the observed
### Table 6.8: The Perfect Information Problem - Data Sets 15-18

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Data Set 15</th>
<th></th>
<th>Data Set 16</th>
<th></th>
<th>Data Set 17</th>
<th></th>
<th>Data Set 18</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Number of</td>
<td>Off-Line</td>
<td>On-Line</td>
<td>Expected</td>
<td>Number of</td>
<td>Off-Line</td>
<td>On-Line</td>
</tr>
<tr>
<td>SSA</td>
<td>62</td>
<td>210.40</td>
<td>16.33</td>
<td>59.99</td>
<td>57</td>
<td>149.99</td>
<td>19.28</td>
</tr>
<tr>
<td>RRA</td>
<td>16</td>
<td>5.02</td>
<td>14.11</td>
<td>61.32</td>
<td>24</td>
<td>12.56</td>
<td>16.88</td>
</tr>
<tr>
<td>LSA</td>
<td>50</td>
<td>144.23</td>
<td>13.20</td>
<td>59.99</td>
<td>38</td>
<td>44.23</td>
<td>14.32</td>
</tr>
<tr>
<td>MSA</td>
<td>18</td>
<td>63.65</td>
<td>12.97</td>
<td>61.32</td>
<td>20</td>
<td>7.35</td>
<td>13.25</td>
</tr>
</tbody>
</table>
Chapter 6. Evaluation of Some POMDP Solution Techniques

<table>
<thead>
<tr>
<th>DATA SET NUMBER</th>
<th>OFF LINE CPU TIME</th>
<th>ON LINE CPU TIME</th>
<th>MEAN</th>
<th>STANDARD DEVIATION</th>
<th>95.0 %C.I.</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>0.17</td>
<td>12.30</td>
<td>61.3</td>
<td>2.04</td>
<td>(60.9, 61.7)</td>
</tr>
<tr>
<td>16</td>
<td>0.19</td>
<td>13.12</td>
<td>44.8</td>
<td>2.34</td>
<td>(44.34, 45.26)</td>
</tr>
<tr>
<td>17</td>
<td>0.44</td>
<td>15.22</td>
<td>54.01</td>
<td>5.61</td>
<td>(52.91, 55.11)</td>
</tr>
<tr>
<td>18</td>
<td>0.23</td>
<td>14.34</td>
<td>76.88</td>
<td>4.89</td>
<td>(75.92, 77.84)</td>
</tr>
</tbody>
</table>

Table 6.9: Performance of PHA- perfect information POMDPs

value of T-statistic, could be as large or larger by chance under the null hypothesis that the mean of the reward from PHA is equal to the mean of the reward from MSA.

6.2 Interpretation of the Results

In this section, we discuss and interpret the results from the Tables presented in section 6.1. Where appropriate, we make comments concerning specific groups in which the test problems were separated.

In examining the runtime data for the algorithms, several important points are observed.

1. In all the Tables which list runtimes for the algorithms, we note that the total CPU time required using SSA is longer than that for other algorithms.

Sondik shows that for any finite horizon POMDP, the optimal value function is piecewise linear and concave, implying that II can be partitioned into a finite number of concave regions within which the value function is linear.
# Chapter 6. Evaluation of Some POMDP Solution Techniques

<table>
<thead>
<tr>
<th>DATA SET NUMBER</th>
<th>T-statistic</th>
<th>h</th>
<th>p</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-12.32</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>-1.12</td>
<td>0</td>
<td>0.26</td>
</tr>
<tr>
<td>3</td>
<td>-0.25</td>
<td>0</td>
<td>0.8</td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>5</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>6</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>7</td>
<td>-0.19</td>
<td>0</td>
<td>0.85</td>
</tr>
<tr>
<td>8</td>
<td>-0.18</td>
<td>0</td>
<td>0.86</td>
</tr>
<tr>
<td>9</td>
<td>-1.39</td>
<td>0</td>
<td>0.16</td>
</tr>
<tr>
<td>10</td>
<td>-0.83</td>
<td>0</td>
<td>0.41</td>
</tr>
<tr>
<td>11</td>
<td>-3.94</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>-1.74</td>
<td>0</td>
<td>0.08</td>
</tr>
<tr>
<td>13</td>
<td>-2.57</td>
<td>1</td>
<td>0.01</td>
</tr>
<tr>
<td>14</td>
<td>-2.25</td>
<td>1</td>
<td>0.02</td>
</tr>
<tr>
<td>15</td>
<td>-0.098</td>
<td>0</td>
<td>0.9</td>
</tr>
<tr>
<td>16</td>
<td>-1.7</td>
<td>0</td>
<td>0.09</td>
</tr>
<tr>
<td>17</td>
<td>-0.34</td>
<td>0</td>
<td>0.73</td>
</tr>
<tr>
<td>18</td>
<td>-0.86</td>
<td>0</td>
<td>0.39</td>
</tr>
</tbody>
</table>

Table 6.10: Performance of PHA- hypotheses testing
The regions, however, must be reconstructed for each iteration at considerable computational expense, and the number of regions necessary for an exact solution can grow exponentially in the time horizon.

We can see in all the Tables that SSA breaks up the space II into more smaller subregions than other algorithms. The number of these regions can be very large even if the number of nondominated vectors is small. SSA finds nondominated $\alpha$ vectors by examining a potentially huge number of them. It is important to note that there is not a relation between dominated vectors and nondominated vectors. As a result, SSA can expend a considerable length of time using linear programs with large number of constraints to examine candidate $\alpha$-vectors which are dominated.

2. SSA performs quite badly with large problems; however, it does well in the case of CUMDP test problems. In this case, a region is defined by a relatively few constraints. Thus, the linear programs which must be run tend to be small and can be run quickly, relative to the linear programs run in SSA with other type of problems.

3. In all the Tables, the number of regions produced by RRA is much smaller than that of SSA. Since fewer regions are produced, this method requires less CPU time than SSA.

In terms of CPU time, RRA performs consistently well against all competitors for virtually all problems tested. RRA tends to perform better versus MSA on large problems; MSA tends to perform better than RRA in small problems.
From our evaluation, it appears that the essential advantage for RRA stems from its ability to maintain a relatively small set of $\alpha-$vectors at each stage.

4. The CPU time for both RRA and LSA are close with small problems (Data Sets: 1, 8, 11, 12, 14): However, with large problems CPU time for LSA increases considerably, due to the great quantity of comparisons that this algorithm processes (Data Sets: 2-6, 7, 9, 10, 13, 15-18).

5. In Data Set 6, LSA (Error = 0) cannot complete the calculations, that is, the algorithm could not solve the problem within 2000 seconds of CPU time. This difficulty is caused by the generation of too many $\alpha-$vectors. The procedure of finding all vertices of regions is deficient under this condition. From our results, we can conclude that LSA might not be as efficient as the SSA if the number of system states or $\alpha-$vectors is large.

6. The CPU time of MSA is quite short with small problems; however, as with LSA, it increases considerably with larger problems. This is due to the high number of $\alpha-$vectors generated at the beginning of each decision period, and the lack of an effective reduction technique to reduce the $\alpha-$vectors for active consideration while constructing the boundaries. From our analysis, we believe this ineffective reduction technique is the reason for the out-of-memory results.

7. In terms of runtime, the algorithms have substantially better performance with CUMDP than with other type of problems. This is quite predictable because as the number of observations increases from one (CUMDP) to two or greater, the number of operations increase considerably.
8. PHA presents the shortest CPU time. This large reduction of CPU time is important to be able to solve large problems in polynomial time.

In examining the Expected Reward for the algorithms, the following important points are observed:

1. The performance of RRA and LSA with these sets of data agree with Cheng's comments about his algorithms. He says that the constraint set for the LSA gives more stable results for finding all vertices of a region than RRA. This could be the reason for having Expected Rewards for RRA (Data Sets 4, 5, and 9) that do not correspond to the values obtained by the application of other algorithms. In addition, the constraint set which defines a region is simple to set up for the LSA.

2. Tables 6.3, 6.5, 6.7, 6.9 show that PHA is competitive with the other algorithms on expected reward.

3. Table 6.10 shows that for many of the test problems, the hypotheses testing are not rejected. that is, the 100-values from the simulation of each set of data could have mean equal to the expected reward obtained with MSA.

The results obtained in this section confirm the restricted application of the algorithms studied previously and serve as a motivation to continue the evaluation of PHA with large problems.

The steel processing problem will be used to evaluate the performance of PHA with large problems. In this case, the results from PHA will be compared to the results obtained by using simulation.
Chapter 7

ROLL SURFACE CONTROL AT HOT STRIP MILL

The objective of this chapter is to show how the control of finishing roll surface deterioration at the hot strip mill can be formulated as a POMDP.

The chapter begins with a description of the steel making and finishing process, the steel metallurgy, the current situation at the Hot Mill, and the problem on which this research is focused. Subsequently, we define the elements of the finite horizon POMDP with discrete and countable sets: the stages (decision periods over the time required to produce a Product Block), the state space (the surface deterioration level of the rolls), the observation space (the steel strip surface quality), and the action space (roll inspection and maintenance).
CHAPTER 7. ROLL SURFACE CONTROL AT HOT STRIP MILL

7.1 PROCESS DESCRIPTION

The production of steel is a complex process involving several intricate and interrelated, yet separate, processes. These processes can be divided into the following two sections (ref. fig. 7.1):

1. Primary Operations

2. Finishing Operations

PRIMARY OPERATIONS

These operations produce the steel products from raw material, and handle liquid or hot solidified semifinished material. These operations can be divided into six areas:

Coke Ovens change coal to its purest hydrocarbon form: coke.

Blast Furnaces produce liquid iron by an exothermic reaction between coke, iron ore, and limestone.

Melt Shop converts liquid iron, scrap steel, some purifying fluxes and alloying elements into 330-ton batches of liquid steel called heats. The customization process begins in this operation, because each heat is made to
Figure 7.1: THE STEEL PRODUCTION PROCESS
CHAPTER 7. ROLL SURFACE CONTROL AT HOT STRIP MILL

<table>
<thead>
<tr>
<th>Dimension</th>
<th>units</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thickness</td>
<td>203 mm to 254 mm (8” to 10” )</td>
</tr>
<tr>
<td>Width</td>
<td>610 mm to 1625 mm (24” to 64” )</td>
</tr>
<tr>
<td>Length</td>
<td>4000 mm to 10363 mm (13” to 34” )</td>
</tr>
<tr>
<td>Weight</td>
<td>Full size slab ≈ 31.78 tons</td>
</tr>
<tr>
<td></td>
<td>Average 13.7 tons</td>
</tr>
</tbody>
</table>

Table 7.1: Range of slab Dimensions

satisfy customer requirements.

**Ladle Metallurgy** each heat is sent to the Ladle Metallurgy area via a refractory, brick-lined steel vessel called a ladle. This centre purifies the product, refines the alloy compositions, and ensures homogeneous chemistry and temperature.

**Continuous Casting** produces slabs that meet customers’ specifications, that is, they possess the required size (width, thickness and length), and quality attributes. The dimensions of each slab are shown in Table 7.1 (Range of slab dimensions).

Identified slabs from Casting can then be handled in one of the following three ways:

1. Transported directly to a Reheat Furnace in the Hot Strip Mill.
2. Taken to the Slab Yard and charged into the furnaces within the next 4 hours.

3. Taken to the Slab Yard and allowed to cool.

**Hot Strip Rolling Mill** The objective of the Hot Strip Rolling mill is to provide large scale reductions in gauge, from 10" to 0.08", while controlling the shape and profile throughout the approximately 4000 feet of produced strip.

Conventional Hot Strip Mills consist of roughing and finishing stands. The configuration of the roughing mills varies widely. A mill with one reversing stand is called a semi-continuous mill. A mill with one reversing stand and one or two continuous roughing stands is called a three quarters \( \frac{3}{4} \) continuous mill, and a mill with 4 to 6 continuous roughing stands is called a continuous mill. Finishing mills have a minimum of 4 stands. Normally they have 6 or 7 stands.

Each rolling stand consists of two pairs of rolls: *work rolls*, which exert force directly to the rolled steel, and *backup rolls*, which help distribute the appropriate stress profile to the work rolls.

In our case, this operation is performed by a main gauge reduction Roughing station, which consists of one reversing stand (semi-continuous mill); and by a Finishing Mill with seven synchronized roll stands.
FINISHING OPERATIONS

The finishing operations control the final specifications of the orders. These operations can be divided into five areas:

Pickling is an acid bath operation to eliminate any surface oxidation.

Cold Rolling produces customer ordered gauges, using a combination of rolling pressure, tension and lubrication.

Annealing consists of heating the steel in an oxygen-free atmosphere for a specific period of time. This operation allows the recrystallisation of the cold rolled sheet. The possibility of changing flatness depends on the type of annealing.

Tempering permits the restoration of the hardness required, and it provides the necessary finish and flatness.

Coating is the application of the specific customer requested surface coating. There are three basic coatings:

1. Tin plating used for food and beverage containers, prevents the steel from altering the taste of food and beverages.

2. Zinc coating provides extra protection against corrosion.
CHAPTER 7. ROLL SURFACE CONTROL AT HOT STRIP MILL

3. Paint coating adds beauty and corrosion protection.

There are two product categories:

1. Cold Roll Steel Strip
2. Hot Roll Steel Strip

Cold Roll Steel Strip represents 50% of total production. The products included in this category are shipped to the customer after they pass all the primary operations. Hot Roll Steel Strip represents 50% of total production. The products included in this category are shipped to the customer after they pass all the primary and finishing operations. The market for the Hot Roll products can be categorized into:

1. Automotive Parts Markets (35%)
2. Pipe and Tube Markets (28%)
3. Service Centres (27%)
4. Other (10%)

7.2 STEEL METALLURGY

When steel is observed under a microscope, it is possible to see numerous crystals or grains. Within the grains, atoms of carbon, manganese, silicon, aluminum and other alloying elements perform specific functions, which control various characteristics of the steel:
Hardness: The resistance to penetration, scratching, abrasion or cutting. It is measured on a relative scale called a sliding Rockwell scale.

Strength: The stress that the material withstands without deforming. It is measured in force per unit area.

Elasticity: Recoverable, non-permanent deformation. Most materials deform somewhat when force is applied to them and return to their original shape when the stress is removed. This ability is described quantitatively using the Modulus of Elasticity.

Plasticity: Non-recoverable permanent deformation.

Steel is alloyed in order to produce varying physical properties. Each variation in steel composition is classified in grades. Each grade requires specific conditions for deformation of the grain, based upon the alloying elements in that grade. Normally, as the amounts of alloying elements are increased, the resistance to deformation increases, and the rate at which the cells return to their normal size and shape slows down. The resistance to forming and the re-crystallizing rate determine the force, the power applied to the roll, and the temperature that varies for different types of steel. (ref.fig. 7.2)

7.3 CURRENT HOT STRIP MILL OPERATING POLICY

To describe the actual roll operating policy at the Finishing Mill, it is important to understand the meaning of Product-Block and Line-Up.

Product-Block is a set of slabs rolled between two consecutive Finishing Mill roll changes.
Figure 7.2: CHANGES IN GRAIN STRUCTURE
Line-Up is a set of slabs rolled between two consecutive Roughing Mill rolls changes.

Both Line-Ups and Product-Blocks should possess a coffin shaped profile with respect to successive slab lengths. The reason is that as each product is rolled, stress concentrations occur on the work rolls at the edges of the product-roll contact area. If too many slabs of the same width are rolled, gouging of the work rolls will result at the contact area. If wider slabs are subsequently rolled, the gouges will cause blemishes on the surface of these products. In extreme cases, these gouges will be transferred to the backup rolls. In order to avoid these problems, an attempt is made to sequence all slabs in a Product-Block such that each slab is no wider than the slab that preceded it in the sequence, and no narrower than the one to follow. This production sequencing strategy is referred to as a coffin profile (ref. fig. 7.3). (The first few slabs rolled during each Product-Block turn actually increase in width due to the need to "warm up" the mill in a systematic manner.) Generally, a number of slabs of the same width have to be rolled sequentially. However, good mill practice usually defines the maximum number of such slabs that may be rolled without a width change so as to avoid defective strip edges.

Currently, all roll sets at the Finishing Mill are replaced after each Product-Block is rolled. This is known as Total Replacement Policy. Furthermore, to meet unforeseen conditions, the current practice is to monitor the progress of deterioration over the period during which the Product-Block is rolled. To do this, a sample is taken from the coil produced from every 20th slab. The deterioration is tracked using these product observations and, if necessary, a decision is made to replace one or more roll sets. This is called Partial Replacement Policy.

The Total Replacement Policy for operating the Finishing Mill is an age replacement policy, which is easy to implement, because it is only necessary to count the number
of slabs rolled. Although the information requirements are low under this policy, it is very restrictive and can be far from optimal in terms of efficient use of roll resources.

The Partial Replacement Policy does not obey any optimization-based strategy, for example, to balance costs associated with decreased roll performance and increased capital investments over the time required to produce a Product-Block.
7.4 ACTUAL CONTROL OF THE IMPORTANT FACTORS WHEN ROLLING

This section describes not only the factors that determine the quality obtained from the operation of slabs through the Hot Strip Mill but also their current control.

Factor 1: Temperature There are two critical temperatures used when rolling, AR3 and AR1. AR3 represents the lower bound temperature at which steel has to be deformed. To heat slabs to a state of plasticity suitable for hot rolling, it is important to have uniform temperature. Heating increases plasticity, which reduces the electrical energy required to push the steel through the rolls. Steel temperature in the roll bite affects the resistance to deformation. As a result, as lower temperatures are encountered, more rolling torque is required to overcome resistance, greater friction is needed to enter the roll bite, and the rate of deformation is reduced. In addition, when the temperature is too low, the grains that reform after rolling are not the same size. Some are large and some are small. This mixture of grain size leads to non-uniform mechanical properties in the steel which affects further processing in the Cold Mill or in the customer's operations.

The function of the reheat furnace is to send steel to the mill at a temperature that produces rolling above AR3. The top surface temperature of the slabs is measured by pyrometers. No bottom surface temperature or in-slab temperature is known. These are predicted using models which allow slab heating control. A number of factors that affect the temperature of the slabs are:

1. Slab dimensions.
2. Walking rate (meters/hour) i.e. rate at which slabs travel through the furnace.

3. Delays.

4. Grade.

5. Hot/Warm/Cold Charge.

AR1 represents the upper bound temperature needed for the steel to attain the required gauge. Pyrometers are used for measuring the temperature of the strip and for process control. They are located in pairs throughout the mill, in specific locations where they can measure temperature accurately without influences like water, steam, and scaling.

If AR3 and AR1 are met, the steel will have a uniform grain size and will therefore exhibit uniform mechanical properties.

**Factor 2: Scale Formation** Scale is a layer of oxidation (corrosion) products formed upon metal at high temperatures. A layer of scale first forms on the slab surface as it exits the Caster. This layer thickens as the slab passes through the Reheat Furnace; it is removed before rolling by high-velocity water sprays. In addition, scale is formed on the steel sheet during rolling and is continuously removed by water sprays. The following factors affect scale formation:

1. Furnace Atmosphere.

2. Temperature.

3. Residence Time.
Factor 3: Width  When width deviates from the desired width target, either of
two costs can be applied: reapply or repair.
A coil is reapplied when it is assigned to another order. The original order is
rolled again.
The repair cost for slab damage due to defective width includes the loss of
between 5 % and 10 % of the coil weight.
There are three width gauges in use at the Hot Strip Mill. Two are located at
the Roughing Mill, the third is located at the exit of the Finishing Mill. All of
them make possible the on-line width control.

Factor 4: Gauge  The purpose of the Automatic Gauge Control (AGC) is to ad-
just the roll gaps to achieve the target gauge. The AGC at the Finishing mill
has two control configurations:

1. Gaugemeter System.
2. X-Ray Monitor System.

The first system maintains gauge on an individual stand by taking into con-
sideration the known roll force and roll gap. The second system measures the
gauge out of stand 7, compares the rolled gauge with the target gauge and
sends a corrective signal back to all stands. This procedure is repeated until
the gauge being produced is equal to the target gauge.

Factor 5: Condition of the rolls  Each roll stand at the Finishing Mill includes
a series of heavy rolls, capable of applying thousands of pounds of force to
squeeze the steel into successively smaller and smaller gauges from one stand
to the next. This stress can cause roll surface deterioration, which can be
observed on rolls as they are removed from service.
To control this factor, Total and Partial Replacement Policies are used.

**Factor 6: Product types** Several different product types are produced in the mill. Each product type may require a specific type of rolling equipment and may have specific quality requirements. Furthermore, the physical process of reducing the dimensions of slabs of different types may affect the quality of the products differently.

A three day Hot Mill Scheduling Plan is employed to expedite the completion of customer orders and to coordinate the various milling operations. To accommodate the fact that different types of orders require different types of milling equipment and/or conditions, the orders coming into the Plan are classified into smaller groups known as Product-Blocks. A Product-Block is a group of customer orders of the same product category, with the restriction that it follows a coffin pattern and its length not exceed some maximum value.

### 7.5 PROBLEM DESCRIPTION

It is a common practice to inspect periodically a system which deteriorates over time, as part of a program to keep it operating efficiently. Actions which are intended, in some sense, to alter optimally the system’s performance are typically determined on the basis of these inspections. Often such inspections do not provide perfect state information, due, for example, to the inaccessibility of important system components or to the prohibitive expense of a thorough inspection. This research is concerned with characterizing strategies which optimally alter a system’s future performance for the case where inspection of the system’s state is imperfect and where the only way the system’s performance can be altered is by partial or total replacement of the system’s components.
The application at the Finishing Mill may be viewed as a discrete-time dynamic control problem. The rolls gradually deteriorate over time, and their degree of deterioration can only be observed by inspections. An imperfection in the quality of the product can provide information about which roll stands are the cause of the decrease in quality and as a consequence are the candidates to be replaced. The objectives are:

1. To make periodic decisions about which roll stands should be replaced to maximize the total expected reward, based on knowledge about the quality of the strip.

2. To show improvement in the quality of the following strips, in an environment with underlying stochastic fluctuations in accumulated surface deterioration, and product quality.

This control is important because roll performance plays a significant part in determining product quality, operating costs, and productive capacity. Since the first four finishing stands constitute the major influence on quality, mill availability, and total roll costs, the scope of this work will be restricted to these four stands. We believe that control of the roll surface deterioration in these stands can contribute greatly to the efficiency of the integrated manufacturing process.

The decision process is based on the following considerations:

1. Tin-Plate is the product to be considered because, the Tin-Plate Product-Blocks are the most uniform Product-Blocks produced by the company.
2. There is a finite number of alternatives to control the process.

3. The deterioration levels of the roll are classified into a finite number of states. It is assumed that the states are identifiable and can be ordered to reflect the degree of deterioration.

4. The quality of the strips takes on only discrete values in a finite set. It is assumed that these values are identifiable and can be ordered to reflect the quality level.

5. The probabilistic relationship between the deterioration level of a roll or group of rolls and the observed product quality is assumed to be known.

6. The decision making periods coincide with the sampling periods.

7.6 FORMULATION OF THE HOT STRIP MILL CONTROL PROBLEM AS A POMDP

Let us consider the first four stands at the Finishing Mill as a system having the following characteristics:

1. The stages of the decision process are represented by \( t \{t = 0, 1, ..., T\} \), i.e., the time periods for which the decision maker must decide on the next period’s
policy to control the rolls deterioration. Each period represents the time required to roll 20 slabs (time at which the quality of the product is determined by taking a sample).

2. Strip quality is observed by end tests taken at 20-coil intervals. Each test involves uncoiling approximately the last 100 feet of the coil, at which point a sample, 8" wide, is obtained. At the same time, the entire length is inspected for defects, such as scale, which may not be present on the sample itself. The sample is then pickled and classified for surface quality (surface texture and residual scale) by comparison with a set of 11 standards (ref. fig. 7.4). This classification allows the determination of two basic conditions:

**Depth of Scale** This is the evaluation of the depth to which scale has been pressed into the steel strip. Iron oxide is much harder than the hot steel and consequently the colder oxide is readily impressed into the surface. The particle size of the scale will determine the depth of the impression.

The depth is rated using a number from 0 to 4.

- **0** ________ smooth-no texture.
- **1** ________ no scale-some roll texture.
- **2** ________ light.
- **3** ________ medium.
- **4** ________ heavy.

**Proximity of Pits and Valleys** This condition refers to how close the pits and valleys, formed by scale, are. Widely spaced valleys are classified as open flecked (OF). Closely spaced valleys are classified as (CF). As more scale is impressed into the strip, the pits and valleys come closer together. Under close examination, the tight proximity resembles a "Wave" shape.
3. The deterioration of the rolls can be classified into a finite number of levels. The roll deterioration levels are ordered to reflect the degree of the deterioration. Level 1 denotes the best condition, when the roll is new. The last level represents the most deteriorated condition.

In this research, the surface condition of the rolls is classified in one of the following four levels: (ref. fig 7.5)

(a) **level 1** Smooth grey black oxide layer.

(b) **level 2** Grey black oxide with some evidence of light peeling.

(c) **level 3** Heavy oxide peeling with light roughness to touch.

(d) **level 4** Very heavy peeling, rough to touch.

According to this classification: New rolls (level 1) in the early finishing stands of a hot strip mill develop a smooth, black oxide that aids in maintaining good strip-surface quality. As a result of the propagation of firecracks, the surface
CHAPTER 7. ROLL SURFACE CONTROL AT HOT STRIP MILL

Figure 7.4: QUALITY CLASSIFICATION OF STRIP SURFACE

0 - Mirror Finish
1 - No Scale: Roll Texture
2 - Light
3 - Medium
4 - Heavy
OF - Open Spaced Fleck
CF - Close Spaced Fleck
W - Wave Pattern
Figure 7.5: **QUALITY CLASSIFICATION OF ROLL SURFACE DE-TERIORATION**
oxide cannot be supported and microspalls of roll material are discarded (level 2). When microspalling progresses, the roll surface roughs (level 3), and this roughness eventually leads to very heavy peeling (level 4). Later in this work, section 8.1.5, it is shown that stand 4 does not significantly affect the quality of the product when Tin-Plate is produced. As a result, we do not consider this stand from now on, and the core process state at time $t$, $s(t)$, is represented by a three-dimensional random variable. Let the random variable $s'(t)$, an element of the vector $s(t)$, represent the actual (unobserved) deterioration level of stand $i$ at period $t$, $i = 1, ..., 3$. The stochastic process \{s(t) = (s^1(t), s^2(t), s^3(t)) \; t = 0, 1, ..., T\} describes dynamics of roll surface deterioration at stands 1, 2 and 3 over the production of a Product-Block.

4. At each sample time, the rolls are monitored indirectly by the information provided for the sample in terms of the quality of the strips. It is noted that this information involves uncertainty and relates probabilistically to the exact state of the rolls. It is assumed that the probabilistic relation between the state of the rolls and the quality of the strips is prescribed by the following known conditional probability:

$$r_{iz} = \Pr\{ \text{the quality of the strip inspected is } z \mid \text{the roll set is in state } i \text{ and the decision made in the last period is } a \}.$$ 

5. The following actions can be selected during the production of each Product-Block:
$a_R$ : Stop the mill and replace all the rolls without inspecting them.

$a_C$ : Continue the finishing mill operation with no examination of the rolls.

$a_{1234}$ : Stop the mill, inspect the rolls, and replace all that are in states 2, 3 or 4.

$a_{134}$ : Stop the mill, inspect the rolls, and replace all that are in states 3 or 4.

$a_{14}$ : Stop the mill, inspect the rolls, and replace all that are in state 4.

6. Each period has an associated reward. This reward consists of the revenue from coils minus any resources used in the rolling process.

The terms included in the total expected reward equation are the following:

(a) **THE OPERATING COST PER DECISION PERIOD** This cost is represented by $C^O$ ($/period) and is roll state independent. This cost includes all the costs incurred in primary operations, and the cost of the material used in each slab.

(b) **THE REPLACEMENT COST** This cost is independent of the number of rolls that are replaced. A replacement takes a fixed number of $T^R$ time units, and involves a fixed cost of $CR$ ($/replacement). After a replacement the roll(s) has level of deterioration 1. The work roll change is performed automatically. The system used to replace rolls employs hydraulic extraction cylinders in front of each stand, and a side-shifting floor. Rolls are pulled from the mill onto the moving floor, which shifts to align the new sets of work rolls so that they can be pushed into the mill by the extracting cylinders.
(c) **ROLLS INSPECTION COST** This cost is represented by $C^{I-R}$ (\$/inspection). If $a_{I234}$, $a_{I34}$ or $a_{I4}$ are selected, the inspection is performed on all rolls.

(d) **REVENUE RECEIVED FROM EACH COIL** The price of a coil depends on its quality, and is represented by $C^Z$. After the sample taken at 20-coil intervals, the quality of these twenty coils is estimated, and the revenue is calculated.

(e) **GRINDING COST** A new roll begins its operation in stand number 4, in which it operates under specific environmental conditions; during its operation, an irreversible diameter reduction occurs due to removal of deterioration effects. This reduction accumulates until the roll can no longer perform satisfactorily in this stand, and it is sent to stand 3, then to stand 2, and finally to stand 1. When a roll can no longer perform satisfactorily in stand 1, the roll is said to have failed; the time at which the roll ceased to perform satisfactorily determines the lifetime of the roll. Since each stand works under very particular conditions, we should study the diameter reduction in each stand. Let us consider, as an example, the diameter reduction of a roll in stand $i$, $i = 1, 2, 3$. Assume the roll is initially new. After a Product-Block is produced or after an unscheduled replacement is performed the roll is removed from the mill, and the grinding operation is performed. This operation can be divided into the following three stages:

**Initial Grind** The roll is ground throughout its length. This operation continues until the effect on the center region is removed.
CHAPTER 7. ROLL SURFACE CONTROL AT HOT STRIP MILL

Rough Grind The roll is ground until all the impression marks are removed.

Finish Grind The finish grind is basically a polishing and equalizing stage.

The procedure for grinding the roll set generates a loss in diameter but allows the roll to be used for the production of another Product-Block. To estimate the average diameter reduction in our example, stand i, we record the loss in diameter for various rolls removed from this stand. Figure 7.6 shows the roll wear profile and grinding stages of a roll removed from stand 4.

The arithmetical average values of diameter reduction as a function of

Figure 7.6: ROLL WEAR PROFILE AND GRINDING STAGES
the surface condition of the rolls are represented as a vector $LD$ ($4 \times 1$), with the following elements:

$$LD = \begin{pmatrix}
LD_1 \\
LD_2 \\
LD_3 \\
LD_4
\end{pmatrix}$$

$LD_j$ represents the average loss in diameter, when rolls are in state $j$ ($j = 1, 2, 3, 4$).

The grinding cost per decision period is calculated by taking into consideration the grind charge rate ($GCR$) (\$/unit time), the stock removal rate ($SRR$) (unit time/inch), and the loss diameter ($LD$) (inches).

$$\text{grinding cost/decision period} = (GCR)(SRR)(LD)(\text{number of rolls})$$

(f) THE PURCHASE COST The purchase cost per decision period is calculated by taking into consideration the following variables: the roll's price ($RP$); the maximum loss in diameter per roll ($MLD$) (inches); and the total loss diameter per period per stand.

Let the average values of loss diameter per period per stand (1, 2 and 3) be represented by the following vectors $LDP1$, $LDP2$ and $LDP3$ respectively. Let us consider, as an example, the diameter reduction per period at stand $i$, $i = 1, 2, 3$ represented as a vector ($LDP_i$) ($4 \times 1$), with the following elements:
CHAPTER 7. ROLL SURFACE CONTROL AT HOT STRIP MILL

\[
LDP_i = \begin{pmatrix}
LDP_{i1} \\
LDP_{i2} \\
LDP_{i3} \\
LDP_{i4}
\end{pmatrix}
\]

$LDP_i_j$ represents the average loss in diameter per period, when rolls at stand $i$ are in state $j$ ($j = 1, 2, 3, 4$).

\[
\text{purchase cost/decision period} = \frac{(2 \times RP)(LDP_{1k} + LDP_{2l} + LDP_{3m})}{(MLD)}
\]

where $k$, $l$, and $m$ represents the state of rolls at stand 1, 2, and 3 respectively.

(g) STRIP INSPECTION COSTS This inspection can be divided into two parts, as follows:

i. The sample taken from the coil produced from every $20^{th}$ slab is inspected. This inspection involves a fixed cost of $C_{1}^{I-P}$ (\$/coil).

ii. If the inspection of the sample reveals that the surface quality of the strip is lower than the surface quality required by the customer, a subsequent inspection of the previously produced coils is arranged. This inspection involves a fixed cost of $C_{2}^{I-P}$ (\$/coil).

All the coils that do not meet the customer's specifications are reapplied. A coil is reapplied when it is assigned to another order. The original order has to be rolled again.

A weekly report is published by the Manufacturing Controls Department, in which coil reapplications throughout the plant are tabulated.
by defect. This report provides an excellent description of quality problems, and it allows us to determine the average number of reapplications ($ANR|z$) as a function of the surface quality observed from the end sample taken at 20-coil intervals.

The expected immediate rewards of making one transition and producing some output $z$, given $s(t) = i$ and $a_t$ are presented in Appendix B.

The formulation of the hot strip mill control problem as a POMDP can be summarized as following:
During the production of a Product-Block, the decision maker is faced with the problem of determining the sequence of optimal actions that maximize the total expected reward obtained.
Since the information about the condition of the rolls is incomplete, it is necessary to infer the state of the rolls from the past data.

The problem is to maximize the expected total reward over the time required to produce a Product-Block. By taking the standard results of the theory of POMDP’s, this problem can be reduced to an equivalent Markov Decision Process (MDP) having a probability distribution $\pi(t)$ as the state of the decision process.

$$\pi(t) = [\pi_i(t)] = [pr(s(t) = i|\xi(t))]$$

Where:

$$\pi_i(t) \geq 0 \ \forall i$$

$$\sum_{i=1}^{N_s} \pi_i(t) = 1$$
The transitions between the states are specified as follows:

1. When the action $a_C$ is selected, $\pi(t) \implies T(\pi|z, a_C)$, where:

$$T(\pi|z, a_C) = \frac{\pi(t)P_a^c R_z^a}{\pi(t)P_a^c R_z^a \cdot 1}$$

2. When the action $a_R$ is selected, $\pi(t) \implies [1, 0, 0, \ldots, 0]$ with probability 1.

3. When the action $a_{1234}$ is selected, $\pi(t) \implies T(\pi|z, a_{1234})$, where:

$$T(\pi|z, a_{1234}) = \frac{\pi(t)P_a^1R_z^a}{\pi(t)P_a^1 R_z^a \cdot 1}$$

4. When the action $a_{134}$ is selected, $\pi(t) \implies T(\pi|z, a_{134})$, where:

$$T(\pi|z, a_{134}) = \frac{\pi(t)P_a^1R_z^a}{\pi(t)P_a^1 R_z^a \cdot 1}$$

5. When the action $a_{14}$ is selected, $\pi(t) \implies T(\pi|z, a_{14})$, where:

$$T(\pi|z, a_{14}) = \frac{\pi(t)P_a^1R_z^a}{\pi(t)P_a^1 R_z^a \cdot 1}$$

Now, determination of an optimal control policy using our proposed heuristic algorithm requires the calculation of the optimal total reward obtained by operating the process from time $t$ to the end of the decision process, under the assumption that the core process is in a known state. The basic functional equations needed for this purpose can be presented as follows:

$$CT_i^T = \gamma_i^T \equiv T$$
We shall assume that the increment of surface deterioration at time inspection \((t)\) depends in a probabilistic manner only on the amount of deterioration present in the last previous inspection \((t - 1)\) and on deterioration accumulated in the period \((t - 1, t)\), and that it is independent of how much damage was accumulated before \((t - 1)\). This is the familiar Markov assumption. Thus, we are viewing deterioration accumulation in the setting of an embedded Markov process, where surface deterioration is only considered at the end of each inspection period.

\[
CT_i^t = \max_{a \in A} \{ \eta_i^a + \sum_{j=1}^{N_s} p_{ij}^a CT_j^{t+1} \} \quad t < T
\]  

(7.1)

7.7 MODEL INPUTS AND OUTPUTS

This section presents the results of the Hot Strip Mill Control Problem corresponding to POMDP formulation. The inputs and outputs of the model are described.

7.7.1 MODEL INPUTS

The Hot Strip Mill Control Problem as a POMDP uses the following sets of input data:

1. The State-Transition Matrices \(P^a\).

2. The State-Observation Matrices \(R^a\).
3. The $N_z$-dimensional column vector of rewards $\gamma^a$.

These input data parameters are described below in more detail.

**State Transition Matrices** These are the period-to-period probability transition matrices $P^a$, whose elements are defined as:

$$p^a_{ij} = Pr\{s(t) = j | s(t-1) = i, a(t-1) = a\}$$

$i$ and $j$ are elements of the following set:

$$\zeta = \{(1,1,1), (2,1,1), ..., (4,1,2), ..., (4,4,4)\}$$

The cardinality in the finite set $\zeta$ is:

$$|\zeta| = (\text{deterioration levels of the rolls})^{\text{number of stands considered}}$$

$$|\zeta| = (4)^3 = 64$$

The process of roll surface deterioration is described by the $(\zeta \times \zeta)$ probability transition matrix $P^a = (p^a_{ij})$.

To check the assumption that the deterioration process for each stand is a homogeneous first-order Markov chain, it is necessary to use statistical tests. For a first-order Markov chain with transition probability matrix $P = (p_{ij})$ the maximum likelihood estimates of the constant transition probabilities $p_{ij}$ are:

$$\hat{p}_{ij} = \frac{n_{ij}}{n_i}$$
If the chain is not homogeneous, then the corresponding estimates of $p_{ij}(t)$ are:

$$\hat{p}_{ij}(t) = \frac{n_{ij}(t)}{n_i(t)}$$

In this case, we denote $n_{ij}(t)$ as the number of sequences passing through the states $i$ and $j$ at the moments of time $(t - 1)$ and $t$, where $t$ varies from 1 to $T$. The following relations can be established:

$$n_i(t - 1) = \sum_j n_{ij}(t)$$

$$n_{ij} = \sum_{t=1}^{T} n_{ij}(t)$$

$$n_i = \sum_{t=0}^{T-1} n_i(t)$$

(7.2)

Appendix C gives 75, 81 and 80 observed sequences of states for stand 1, 2 and 3 respectively.

It is shown by Anderson and Goodman [1] that the assumption that the chain is a homogeneous first-order chain can be tested by using the following $\chi^2$ test with $(N_s)T$ degrees of freedom.

$$\chi_i^2 = \sum_j n_i(t - 1)[\hat{p}_{ij}(t) - \hat{p}_{ij}]^2/\hat{p}_{ij}$$

Where:

$\chi_i^2$ has an asymptotic $\chi^2$ distribution with $(N_s)T$ degrees of freedom.

If the null hypothesis ($p_{ij}$ are independent of $t$) is true, then the second step is to check the hypothesis that the order of the chain is less than 2. The maximum likelihood estimates of the transition probabilities $\hat{p}_{ijk}$ of the second-order chain can be obtained in a similar way to equations 7.2. Under the null hypothesis, the asymptotic distribution of:
\[ \chi_j^2 = \sum_{i,k} n_{ij}^* \left( \hat{p}_{ijk} - \hat{p}_{jk} \right)^2 / \hat{p}_{jk} \]

will be a \( \chi^2 \) distribution with \( (N_2^2) \) degrees of freedom where:

\[ n_{ij}^* = \sum_{t=1}^{T-1} n_{ij}(t) \]

Using the maximum likelihood ratio and \( \chi^2 \) tests, it has been calculated that the null hypotheses about the order and homogeneity can not be rejected with a significance level \( \alpha \) equal to 0.05. Thus, we may assume that the deterioration process of rolls is a homogeneous Markov chain of the first order. The calculated \( \chi^2 \) values are given in the floppy disk. The estimated transition matrices are given in the floppy disk.

**State-Observation Matrices** These are period-to-period probability matrices \( R^a \) which define the probabilistic relationship between the actual deterioration of the rolls (the state of the core process) and the quality of the strip surface (the state of the observation process).

The estimated state-observation matrices are given in the floppy disk.

**Reward vectors** These are the vectors \( \gamma^a \) which make up the values \( CT' \). The estimated \( \gamma^a \) vectors are given in the floppy disk, and its calculation is in Appendix B.

### 7.7.2 MODEL OUTPUTS

The results of the Hot Strip Mill Control corresponding to POMDP formulation are generated as a file that reports the following:
• The Total Expected Reward obtained during the production of each Product Block.

• The Optimal Strategy that was used during the production of each Product Block.

• CPU Time

The analysis of results, implications, and perspectives based on the results of this model are discussed in chapter 9.
Chapter 8

A SIMULATOR FOR THE HOT STRIP MILL

The simulator for the Hot Strip Mill imitates the deterioration process of the rolls during the production of a Product Block. There are three objectives in developing the simulator:

1. TO EVALUATE THE PERFORMANCE OF OUR PROPOSED HEURISTIC ALGORITHM WITH LARGE PROBLEMS

To accomplish this, the results from our proposed heuristic algorithm will be compared to the results obtained using Simulation, evaluating actual data from Tin-Plate Product Blocks with both procedures. The comparison is based on the strategies obtained, the rewards, and the processing time.
2. TO DETERMINE THE SIZE OF THE PRODUCT BLOCK

Perhaps the greatest benefit of this simulator is that it allows a system-wide view of the effect of changes in the size of the Product Block on the manufacturing system.

If a Product Block has too many slabs, the rolls will wear, causing production of poor quality steel coils. If a Product Block is too short, the rolls will be changed before it is necessary, which creates more frequent set ups and higher costs of roll maintenance. By using the simulator for the Hot Strip Mill, and by taking into consideration that a Product Block is the set of slabs processed between two consecutive changes of Finishing Mill work rolls, we determine the optimal length of a Tin Plate Product Block. In order to do that, the department in the company that is in charge of the generation of schedules for the Hot Strip Mill will provide Tin Plates Product Blocks with sizes that go from a minimum to maximum size.

3. TO DETERMINE THE FREQUENCY OF SAMPLING

The objective is to find a monitoring procedure by drawing random samples of size 1 at times that maximizes the expected reward per Product Block.

In addition to the above general objectives of the simulator, there are a number of specific potential benefits from using simulation for the manufacturing
analyses, including:

1. Increased coils produced per unit of time.

2. Increased utilization of rolls or workers.

3. Increased on-time deliveries of coils to customers.

4. Insurance that a proposed size of a Product Block will, in fact, operate as expected.

5. Information gathered to build the simulation model will promote a greater understanding of process, which often produces other benefits.

The company supplied us with a large amount of data that already existed in its computer databases and reports; however, a significant effort was required to get the data conforming to model requirements. The information provided per slab is the following: Serial number of the Product Block that contains the slab, serial number of the slab, serial number of the rolls, date in which the slab is sent to the hot mill, slab width, slab grade, slab gauge to be obtained in each stand, rolls diameter, stands speed, slab speed in each stand, stands force, surface deterioration of the rolls that are removed from the mill.

The Minitab statistical software package was used to analyze the data and to determine the appropriate probability distribution for each source of randomness.

The simulation model was coded in the MATLAB simulation language, although other languages could have been used as well. MATLAB was selected
because of its flexibility and simplicity.

To verify that the simulation program was working as intended, the following actions were taken:

1. The model was coded and debugged in steps.

2. An interactive debugger was used to verify that each program path was correct.

3. Model output results were checked for reasonableness.

4. All fitted probability distributions were tested for correctness.

To simulate roll surface deterioration, it is necessary to generate a stochastic cumulative damage model that not only possesses the major sources of variability in life prediction as an inherent part of its structure, but also provides a description of fatigue and surface deterioration that is consistent with the complexity of the physical phenomena.

The following section describes this model.
8.1 CUMULATIVE DAMAGE MODEL FOR THE FINISHING ROLL SURFACE TO BE USED IN THE SIMULATION

The rolling conditions are directly related to the hot strip mill configuration and the rolling practices.

1. The mill configuration consists of:

   (a) a sequence of stands with a specific type of rolls.

   (b) maximum separation force (for each stand).

   (c) maximum torque (for each stand).

   (d) speed ranges (for each stand).

   (e) diameters of the rolls.

   (f) cooling system used in the finishing mill.

2. The rolling practices consist of:

   (a) strip grade.

   (b) slab and strip temperatures.
(c) slab and strip dimensions.

(d) load distribution.

However, only the actual pass design and the real rolling schedules show what happens in the bite of each pass, providing basic information of the rolling conditions. The schedules for the finishing mill may change from strip to strip. Nevertheless, in this research, the schedules correspond to Tin-Plate, the most uniform Product-Blocks generated.

Following, we will try to establish a relation between surface deterioration of the rolls at each stand and other important variables.

It has been generally accepted ([21], [12], [9]) that for the work rolls at the front finishing stands ($F1 - F4$), thermal fatigue is the primary cause of roll surface deterioration. The thermal fatigue effect can be explained as follows: The work rolls are in contact with the steel strip at a temperature of about $1000^\circ C$. Heat flows from the strip to the roll over the area of contact between strip and roll. The mean temperature of the roll is maintained in the interval $[50^\circ - 100^\circ C]$ by water cooling. If the difference between the roll surface temperature and its body temperature is sufficiently great, then compressional forces acting at the surface appear. These forces are cyclically imposed on the roll surface with the same frequency as the roll rotation. The roll surface is thus subjected to a thermal fatigue action which causes cracking at the roll surface.

To understand the mechanism of the roll surface deterioration in the first
four finishing stands, which are closely related with the surface quality of the strips. The following variables are considered (ref. fig. 8.1):

1. Strip thickness $H_1$ before and $H_2$ after the pass (mm).

2. Speed of strip $SV_1$ before and $SV_2$ after the pass.

3. Separation force $P_i$ ($kg/mm^2$).

4. Strip temperature $STemp$ ($^\circ C$).
5. Strip width SWidth.

6. Roll diameter $D_i$ (mm).

7. Roll speed $RV_i$ (rpm).

8. Strip reduction factor $\frac{\Delta H}{H_i} = \frac{H_1 - H_2}{H_i}$

9. Bite angle $\cos \alpha = 1 - \frac{\Delta H}{D_i}$

10. Contact length between strip and work roll.
    $$L_i = \frac{(\alpha)(\pi)(D_i)}{360}$$

11. Contact time between strip and work roll.
    $$t_i = \frac{L_i}{RV_i}$$

12. Coefficient of heat penetration from strip to work roll.
    $$\bar{W}_i = (\alpha)(t_i)$$

13. The fatigue ductility exponent $c$.

The thermal fatigue damage ($TFD$) is described by the Manson-Coffin equation [21] as a function of the temperature difference during one revolution $\Delta T$, the heat penetration depth at the exit of a roll bite $\bar{W}_i$, the number of revolutions during rolling $RV_i$, the fatigue ductility exponent $c$ (0.788 for cast iron and 0.9 for high-speed tool steel), and $n$ (exponent that represents the
dependency of the plastic strain range on the heat penetration depth). That is:

\[ TFD = F(\Delta T, n, c, \overline{W_i}, RV_i) \]  

(8.1)

The initial roll temperature is assumed to be \( T_0 = 30^\circ C \).

\[ TFD = \sum_{j=1}^{m} \Delta T_{ij} \overline{W}_{ij}^{-n} RV_{ij}^c \]  

(8.2)

Where:

\( m \) represents the number of slabs in a Product-Block.

subscript \( j \) represents the \( j^{th} \) strip.

subscript \( i \) represents the \( i^{th} \) stand.

The thermal fatigue damage per decision period \( t \), can be expressed as:

\[ TFD_t = \begin{cases} 0 & t = 0 \\ TFD_{t-1} + \sum_{k=1}^{m_t} \Delta T_{ik} \overline{W}_{ik}^{-n} RV_{ik}^c & t > 0 \end{cases} \]  

(8.3)

(8.4)

Where:

\( m_t \) represents the number of slabs in period \( t \).

Formula 8.3 is used when the alternative selected at time \( t \) represents the replacement of the roll.

Formula 8.4 is used when the alternative selected at time \( t \) does not represent the replacement of the roll.
8.1.1 TEMPERATURE DIFFERENCE $\Delta T_i$ DURING ONE REVOLUTION

To predict the temperature distribution inside the rolls F1-F4 as a function of the operating parameters, including the spray cooling configuration, a thermal model of hot-strip mill work rolls developed by X. Ye and I.V. Samarasekera at the Department of Metal and Materials Engineering, Centre for Metallurgical Process Engineering, University of British Columbia, [63](1994), is used.

8.1.2 FORMULATION OF THE THERMAL MODEL

Referring to figure 8.2, it is seen that heat enters a work roll to raise its temperature above ambient as a result of the following factors:

1. Radiation from the workpiece entering and leaving the roll bite.

2. Conduction from the workpiece through a layer of oxide.

3. Frictional effects along the arc of contact in the roll bite.

4. Frictional effects in the roll-neck bearings.

5. Rolling friction at the work roll/backup roll contact.

As illustrated in figure 8.3, heat leaves the work rolls in a number of ways:
Figure 8.2: FLOW OF HEAT INTO A WORK ROLL
Figure 8.3: HEAT LOSSES FROM A WORK ROLL
1. The action of the water sprays.

2. Convection in air.

3. Radiation to surroundings.

4. Contact with the backup rolls.

5. Flow in the axial direction toward the roll necks.

To understand the mathematical expressions presented in Ye and I.V. Samarasekera’s model [63](1994) related to the thermal aspects of hot rolling, heat conduction equations are developed in this section.

The basic law of thermal conduction was postulated by Fourier in 1812 and may be expressed as:

\[ dQ = -K\frac{\partial T}{\partial x}dx \]

(8.5)

Where:

- \( dA \) : Represents a small planar area inside the body.
- \( \frac{\partial T}{\partial x} \) : Represents the temperature gradient perpendicular to the plane.
- \( dQ \) : Represents the quantity of heat flowing through the plane in time \( dt \).
- \( K \) : Represents the thermal conductivity of the material.

The negative sign shows that heat flows down the temperature gradient.

If we consider a rectilinear volume as illustrated in figure 8.4, the rate of heat flowing into the volume in a direction parallel to the \( x \) axis is denoted by \( q_{x(\text{in})} \), where:

\[ q_{x(\text{in})} = -(dz\cdot dy)K\frac{\partial T}{\partial x} \]

(8.6)
Figure 8.4: COOLING OF MILL ROLLS

Where:

$K_x$ represents the thermal conductivity in the $x$ direction.

The difference between the rates of heat entering and leaving along the $x$ axis can be obtained by differentiating the expression for $q_{x\,(in)}$ with respect to $x$. That is,

$$\Delta q_x^{(in-out)} = dy.dz.[\delta(K_x\delta T/\delta x)/\delta x].dx$$

(8.7)

Similarly, for the other two directions, the net heat entering or leaving the volume can be written as:

$$\Delta q_y^{(in-out)} = dx.dz.[\delta(K_y\delta T/\delta y)/\delta y].dy$$

(8.8)

and:

$$\Delta q_z^{(in-out)} = dx.dy.[\delta(K_z\delta T/\delta z)/\delta z].dz$$

(8.9)

The net rate at which heat is stored in the volume, $\sum q_{\text{stored}}$, is equal to:

$$\sum q_{\text{stored}} = \rho.C.dx.dy.dz.\delta T/\delta t$$

(8.10)
Since the difference between the rates of heat entering and leaving along the three axes equals the net rate of heat storage, then, from the last four equations we have the following:

\[
\rho C \frac{dx \cdot dy \cdot dz \cdot \delta T}{\delta t} = dy \cdot dz \cdot [\delta(K_x \delta T/\delta x)/\delta x] \cdot dx \\
+ dx \cdot dz \cdot [\delta(K_y \delta T/\delta y)/\delta y] \cdot dy \\
+ dx \cdot dy \cdot [\delta(K_z \delta T/\delta z)/\delta z] \cdot dz
\]

(8.11)

or

\[
\rho C \frac{\delta T}{\delta t} = \delta(K_x \delta T/\delta x)/\delta x + \delta(K_y \delta T/\delta y)/\delta y + \delta(K_z \delta T/\delta z)/\delta z
\]

(8.12)

This equation is known as the Fourier-Poisson equation for unsteady heat conduction.

If the thermal conductivity is assumed to be independent of variation in temperature, time, and position; the general heat equation becomes:

\[
\frac{\delta T}{\delta t} = (K/\rho C)(\delta^2T/\delta x^2 + \delta^2T/\delta y^2 + \delta^2T/\delta z^2)
\]

\[
\frac{\delta T}{\delta t} = \alpha(\delta^2T/\delta x^2 + \delta^2T/\delta y^2 + \delta^2T/\delta z^2)
\]

(8.13)

The parameter \(\alpha\) represents the thermal diffusivity of the material and is expressed in terms of units of length squared divided by units of time.

For a unidirectional heat flow, the general heat conduction may be written as:

\[
\frac{\delta T}{\delta t} = \alpha(\delta^2T/\delta x^2)
\]

(8.14)

Instead of using orthogonal axes, it is often more convenient to express heat conduction equations in terms of cylindrical coordinates. If we consider a solid
element (ref. Fig. 8.5) of length $dz$, radius $r$ for the inner surface, radius $r + dr$ for the outer surface, and the sides of the element subtending an angle $d\theta$ at the $z$ axis. The net heat flow radially into the solid element (indicated by $A$ in the figure 8.5) is:

$$dz \cdot r \cdot d\theta \cdot \left[ \frac{\delta (K \delta T)}{\delta r} / \delta r \right] dr + dz \cdot dr \cdot d\theta \cdot K \delta T / \delta r \tag{8.15}$$

In the tangential direction, the net heat flow, indicated by $B$, is:

$$dz \cdot dr \cdot (1/r) \cdot \left[ \frac{\delta (K \delta T)}{\delta \theta} / r \right] \cdot r \cdot d\theta \tag{8.16}$$
and parallel to the $z$ axis, indicated by $C$

$$r \, d\theta \, dr \left[ \delta \left( K \frac{\delta T}{\delta z} \right) / \delta z \right] dz$$

Since the net rate of heat storage is $\rho C \, r \, d\theta \, dr \, dz \, \delta T / \delta t$, the heat flow balance equation can be written as:

$$\rho C \, r \, d\theta \, dr \, dz \, \delta T / \delta t = K \left( \delta^2 T / \delta r^2 + \delta T / \delta r / r + \delta^2 T / \delta \theta^2 / r^2 + \delta^3 T / \delta z^2 \right) dz \, dr \, d\theta$$

so that:

$$\delta T / \delta t = K / \rho C \left( \delta^2 T / \delta r^2 + \delta T / \delta r / r + \delta^2 T / \delta \theta^2 / r^2 + \delta^2 T / \delta z^2 \right)$$

This equation represents the heat conduction equation in terms of cylindrical coordinates.

The heat conduction equation is solved subject to the following assumptions:

1. Taking into account that roll speed is high, heat conduction in the circumferential direction can be ignored relative to the heat transfer by bulk rotation.

2. The thermal field approaches a steady state.

The heat conduction equation under these assumptions is:

$$\delta T / \delta t = K / \rho C \left[ \delta^2 T / \delta r^2 + r \delta T / \delta r + \delta^2 T / \delta z^2 \right]$$
The following symbols are used to establish the thermal boundary conditions:

\( h \) : Coefficient of heat transfer between the roll surface and its surroundings.

\( T_s \) : Temperature of work rolls surface.

\( T_A \) : Temperature of medium around a roll.

\( \delta T/\delta r \) : Represents the temperature gradient perpendicular to the roll surface.

\( K \) : represents the thermal conductivity of the material.

\[
h(T_s - T_A) = -K\delta T/\delta r \quad (8.21)
\]

The value of \( h \) is determined [31] for conditions at each of the following locations:

1. In the roll bite.

2. Beyond the roll gap.

3. In the region of spray water.

4. In the region where a film of water streams down from the spray zone.

8.1.3 NUMERICAL SOLUTION AND VALIDATION OF THE THERMAL MODEL

In the two dimensional heat-transfer equation (8.21 ), the temperature is regarded as exhibiting no angular variation; the concern is with temperature
variations in the radial and axial directions. The equation has been solved by the Alternating Direction Implicit Finite Difference Method [63](1994). To calculate the accuracy of the thermal model of the work roll, the prediction was compared with the results of an analytical solution for heating a cylinder. The agreement between the two models, calculated at the Centre for Metallurgical Process Engineering was found to be excellent.

Roll surface temperature variation during a single revolution of the roll, under steady-state conditions, is shown in Figure 8.6. These figures were obtained at the Centre for Metallurgical Process Engineering.

### 8.1.4 DETERMINATION OF HEAT PENETRATION DEPTH $\overline{W_i}$

The penetration depth, $\overline{W_i}$ in equation 8.4, can be obtained from the data of strip reduction $\Delta H$, contact time between strip and work roll, and roll speed. In addition, the exponent $n$ in equation 8.4 was assumed to be the value which best describes the deterioration of the front finishing stands (F1-F4).

### 8.1.5 CORRELATION BETWEEN ROLL SURFACE BREAKDOWN AND THERMAL FATIGUE DAMAGE

The deterioration of the roll surface was evaluated by observing the surface condition of the rolls after each roll change. The degree of the roll surface deterioration is classified into four levels, as mentioned previously.

Figure 8.7 shows the relationship between the observed surface of the rolls and the correspondent thermal fatigue damage. The data were collected from 130 Tin-Plate Product-Blocks for four front-work rolls. The thermal fatigue
Figure 8.6: WORK ROLLS SURFACE TEMPERATURE
CHAPTER 8. A SIMULATOR FOR THE HOT STRIP MILL

<table>
<thead>
<tr>
<th>STAND</th>
<th>CORRELATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>F1</td>
<td>0.946</td>
</tr>
<tr>
<td>F2</td>
<td>0.885</td>
</tr>
<tr>
<td>F3</td>
<td>0.841</td>
</tr>
</tbody>
</table>

Table 8.1: Correlation between Roll Surface Breakdown and Thermal Fatigue Damage

damage of the front work rolls at the hot strip mill was described by using formula 8.4 determined from the temperature fluctuation and the heat penetration depth. The formula was found to have a good correlation with the observed roll surface breakdown behavior. As a result, it was assumed that the thermal fatigue damage can be used in designing roll deterioration control. The correlations are recorded in Table 8.1.

As we can see in fig. 8.7, the rolls at stand 4 do not present significant surface deterioration during the production of a product block. The reason for this is that the total force to squeeze the steel into successively smaller and smaller gauges is often applied for the Tin Plate product in the first three stands. Therefore, these rolls do not affect the quality of the coils, and they do not have to be included in our analysis.

8.1.6 PROBABILITY DISTRIBUTIONS FOR THE MODEL

As we said before, Minitab was used to analyze the data and to determine the appropriate probability distribution for each one of the sources of randomness. Given that we simply cannot find a theoretical distribution that fits the deterioration of the roll data adequately, we use the observed data themselves to specify directly a distribution, called an empirical distribution, for which
Figure 8.7: SURFACE RATING OF THE ROLLS vs THERMAL FATIGUE DAMAGE
random values are generated during the simulation.

To specify the empirical conditional distribution of roll surface deterioration given the thermal fatigue, we group the thermal fatigue into 4 adjacent intervals \([0,800), [800,1250), [1250,1700), [1700,\infty)\], and specify the empirical conditional distribution function of roll surface deterioration given the thermal fatigue (see Table 8.2).

Figure 8.8 shows the relationship between the thermal fatigue damage at each stand and the number of slabs produced.

### Table 8.2: Conditional Distribution Function of roll surface deterioration given the Thermal Fatigue

<table>
<thead>
<tr>
<th>ROLL STATE</th>
<th>THERMAL FATIGUE INTERVALS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>([0,800))</td>
</tr>
<tr>
<td>1</td>
<td>0.97</td>
</tr>
<tr>
<td>2</td>
<td>0.03</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
</tr>
</tbody>
</table>

8.2 **THE SIMULATOR PROGRAM**

This simulator evaluates at each decision time every given action in terms of costs. The simulator is based on the concept of the next event. The simulator considers the following 5 events:
Figure 8.8: CUMULATIVE PRODUCTION vs CUMULATIVE THERMAL FATIGUE
<table>
<thead>
<tr>
<th>Number of the Event</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EVENT 1</td>
<td>First Slab in the Product Block</td>
</tr>
<tr>
<td>EVENT 2</td>
<td>Slab Processed Between Two Consecutive Decision Times</td>
</tr>
<tr>
<td>EVENT 3</td>
<td>Slab Processed at Decision Time</td>
</tr>
<tr>
<td>EVENT 4</td>
<td>Last Slab in the Product Block</td>
</tr>
<tr>
<td>EVENT 5</td>
<td>End of the Simulation</td>
</tr>
</tbody>
</table>

Each event performs some of the following procedures:

- **PROCEDURE 1: ESTIMATION OF THE THERMAL FATIGUE DAMAGE PRODUCED ON EACH STAND**

This estimation is made by using formulas 8.3 and 8.4.
Formula 8.3 is used when the alternative selected at time $t$ represents the replacement of the roll, and formula 8.4 is used when the alternative selected at time $t$ does not represent the replacement of the roll.

- **PROCEDURE 2: ESTIMATION OF THE SURFACE CONDITION STATE OF EACH STAND**

To obtain the roll surface deterioration condition given the accumulated thermal fatigue and the distribution presented in table 8.2, we use the inverse-transform method.
When the variate $X$ is discrete, the distribution function is:

$$F(x) = (X \leq x) = \sum_{x_i \leq x} p(x_i)$$
Where \( p(x_i) \) is the probability mass function.

If we assume that \( X \) can take on only the values \( x_1, x_2, \ldots \) where \( x_1 < x_2 < \ldots \) then the algorithm is as follows:

1. Generate \( U \sim U(0, 1) \).
2. Determine the smallest positive \( I \) such that \( U \leq F(x_I) \), and return \( X = x_I \)

**PROCEDURE 3: ESTIMATION OF THE STRIP QUALITY**

A regression equation for a particular response \( Y \) (Strip Quality) in terms of the following independent or predictor variables is determined.

\[
X_1 = \text{Thermal Fatigue Damage at Stand 1.}
\]
\[
X_2 = \text{Thermal Fatigue Damage at Stand 2.}
\]
\[
X_3 = \text{Thermal Fatigue Damage at Stand 3.}
\]

The regression equation is:

\[
Y = 1.81 - 0.0812X_1^{0.5} + 0.0130X_2^{0.8} + 0.0112X_3^{0.33}
\]

This model is intrinsically linear, because it can be expressed, by suitable transformation of the variables, in the standard linear model.

\( R^2 = 81.1\% \), and all estimated coefficients are statistically significant with \( \alpha = 0.05 \).

**PROCEDURE 4: ESTIMATION OF THE REWARD**

This reward consists of the revenue from coils minus any resources used in the rolling process. The terms included in the total expected reward
equation are shown in appendix B.

The simulator checks which is the next event. If the next event is EVENT 1, the program updates the number of slabs that passes through each stand and the thermal fatigue damage in each stand with the zero values. The simulator passes the first slab through the Hot Mill, updates the number of slabs that passes through each stand, and performs procedures 1, 2, 3, and 4.

If the next event is EVENT 2, the simulator passes the slab through the Hot Mill, updates the number of slabs that passes through each stand, and performs procedures 1, 2, 3, and 4.

If the next event is EVENT 3, the simulator passes the slab through the Hot Mill, updates the number of slabs that passes through each stand, performs procedures 1, 2, 3, and 4, evaluates the actions that can be taken at this decision time to determine the one that maximizes the Total Expected Reward, and updates the number of slabs that passes through each stand and the thermal fatigue damage on each stand according to the decision taken.

If the next event is EVENT 4, the simulator passes the slab through the Hot Mill, updates the number of slabs that passes through each stand, performs procedures 1, 2, 3, and 4, and generates a file that reports the total expected reward obtained during the production of that Product Block, and the optimal strategy that was used.

If the next event is EVENT 5, the simulation ends.
Chapter 9

ANALYSIS OF RESULTS

The simulator for the Hot Strip Mill, developed in chapter 8, is used in this chapter to do the following:

1. Compare the results from our proposed heuristic algorithm to the results obtained by using Simulation. The comparison is based on the strategies obtained, and the rewards. The sensitivity of the results to changes in the input data is explored.

2. Obtain a system-wide view of the effect of changes in the size of the Product Block on the manufacturing system. The comparison between different Product Blocks' sizes is based on the reward.

3. Find a monitoring procedure by drawing random samples of size 1 at times that maximizes the expected reward per Product Block.
4. Compare the performance between proposed and current control procedure.

9.1 EVALUATION OF THE PERFORMANCE OF OUR PROPOSED HEURISTIC ALGORITHM WITH THE MILL PROBLEM

To see how precise our proposed heuristic algorithm for the Hot Strip Mill is, it is necessary to compare the results of it against the results of the simulator. The simulator evaluates at each decision time every given action in terms of costs. To do this, the simulator uses actual data from 10 Product-Blocks. Each Product-Block has 120 slabs, and each slab has the information required to estimate the thermal fatigue produced when it passes through the Hot Mill. The Proposed Heuristic uses the following sets of input data: the state-transition matrices, the state-observation matrices, and the column vectors of reward. The results allow us to compare the optimal strategy used and the Total Expected Reward obtained per Product Block. Results are presented in Tables 9.1 and 9.2. The stages of the decision process are represented in these tables by \( t \ {t = 0, 1, ..., T} \), i.e., the time periods for which the decision maker must decide on the next period’s policy to control the rolls deterioration. Each period represents the time required to roll 20 slabs (time at which the quality of the product is determined by taking a sample).

With respect to the sequence of actions selected (strategies), it can be seen in tables 9.1 and 9.2 that both procedures generates the same optimal strategies when the size of the Product-Blocks is 120. In addition, the only actions
## CHAPTER 9. ANALYSIS OF RESULTS

<table>
<thead>
<tr>
<th>PB</th>
<th># Slabs</th>
<th>Exp.Reward</th>
<th>$t = 0$</th>
<th>$t = 1$</th>
<th>$t = 2$</th>
<th>$t = 3$</th>
<th>$t = 4$</th>
<th>$t = 5$</th>
<th>$t = 6$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>120</td>
<td>308697</td>
<td>$a_C$</td>
<td>$a_C$</td>
<td>$a_C$</td>
<td>$a_C$</td>
<td>$a_C$</td>
<td>$a_C$</td>
<td>$a_R$</td>
</tr>
<tr>
<td>2</td>
<td>120</td>
<td>308723</td>
<td>$a_C$</td>
<td>$a_C$</td>
<td>$a_C$</td>
<td>$a_C$</td>
<td>$a_C$</td>
<td>$a_C$</td>
<td>$a_R$</td>
</tr>
<tr>
<td>3</td>
<td>120</td>
<td>306655</td>
<td>$a_C$</td>
<td>$a_C$</td>
<td>$a_C$</td>
<td>$a_C$</td>
<td>$a_C$</td>
<td>$a_C$</td>
<td>$a_R$</td>
</tr>
<tr>
<td>4</td>
<td>120</td>
<td>307921</td>
<td>$a_C$</td>
<td>$a_C$</td>
<td>$a_C$</td>
<td>$a_C$</td>
<td>$a_C$</td>
<td>$a_C$</td>
<td>$a_R$</td>
</tr>
<tr>
<td>5</td>
<td>120</td>
<td>308565</td>
<td>$a_C$</td>
<td>$a_C$</td>
<td>$a_C$</td>
<td>$a_C$</td>
<td>$a_C$</td>
<td>$a_C$</td>
<td>$a_R$</td>
</tr>
<tr>
<td>6</td>
<td>120</td>
<td>310023</td>
<td>$a_C$</td>
<td>$a_C$</td>
<td>$a_C$</td>
<td>$a_C$</td>
<td>$a_C$</td>
<td>$a_C$</td>
<td>$a_R$</td>
</tr>
<tr>
<td>7</td>
<td>120</td>
<td>309044</td>
<td>$a_C$</td>
<td>$a_C$</td>
<td>$a_C$</td>
<td>$a_C$</td>
<td>$a_C$</td>
<td>$a_C$</td>
<td>$a_R$</td>
</tr>
<tr>
<td>8</td>
<td>120</td>
<td>308579</td>
<td>$a_C$</td>
<td>$a_C$</td>
<td>$a_C$</td>
<td>$a_C$</td>
<td>$a_C$</td>
<td>$a_C$</td>
<td>$a_R$</td>
</tr>
<tr>
<td>9</td>
<td>120</td>
<td>308996</td>
<td>$a_C$</td>
<td>$a_C$</td>
<td>$a_C$</td>
<td>$a_C$</td>
<td>$a_C$</td>
<td>$a_C$</td>
<td>$a_R$</td>
</tr>
<tr>
<td>10</td>
<td>120</td>
<td>311222</td>
<td>$a_C$</td>
<td>$a_C$</td>
<td>$a_C$</td>
<td>$a_C$</td>
<td>$a_C$</td>
<td>$a_C$</td>
<td>$a_R$</td>
</tr>
</tbody>
</table>

Table 9.1: Results using Simulation

<table>
<thead>
<tr>
<th># Slabs</th>
<th>Exp.Reward</th>
<th>$t = 0$</th>
<th>$t = 1$</th>
<th>$t = 2$</th>
<th>$t = 3$</th>
<th>$t = 4$</th>
<th>$t = 5$</th>
<th>$t = 6$</th>
</tr>
</thead>
<tbody>
<tr>
<td>120</td>
<td>296788</td>
<td>$a_C$</td>
<td>$a_C$</td>
<td>$a_C$</td>
<td>$a_C$</td>
<td>$a_C$</td>
<td>$a_C$</td>
<td>$a_R$</td>
</tr>
</tbody>
</table>

Table 9.2: Results using Proposed Heuristic Algorithm
selected during the production of 120 slabs were: Continue the finishing mill operation with no examination of the rolls $a_C$, and stop the mill and replace all the rolls without inspecting them $a_R$. The reason for this situation is the high cost involved in the inspection of the rolls.

The strategies to control the deterioration of the rolls with different sizes of Product-Blocks were determined by using both procedures, finding that they generate the same optimal strategies. In addition, actions related to inspection of rolls were never selected.

The differences between the results generated by the Simulation and the Proposed Heuristic are in the Expected Reward. We notice that the Expected Rewards obtained with our proposed heuristic algorithm are systematically lower than the ones obtained with the Simulation Model. These differences exist because Simulation estimates the thermal fatigue damage produced on each stand, the surface condition state of each stand, and the strip quality after each slab passes through the Hot Mill, while our proposed heuristic algorithm estimates it at 20-strips intervals.

The sensitivity of the POMDP resulting in changes to the input data is explored by defining different scenarios affecting specific data items. The Scenarios are defined specifically for rolls inspection cost, replacement cost, operating cost per decision period, strip inspection cost, grinding cost, and price of the rolls.

Scenarios are defined by specifying the following data items:

**Rolls Inspection Cost**

The analysis of the surface deterioration control begins with a specification of different percentages by which the rolls inspection cost is reduced.

The corresponding inputs are used in the computer model to generate the
CHAPTER 9. ANALYSIS OF RESULTS

Table 9.3: Sensitivity Analysis - Inspection Cost of the Rolls

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Roll Insp. Cost</th>
<th>t = 0</th>
<th>t = 1</th>
<th>t = 2</th>
<th>t = 3</th>
<th>t = 4</th>
<th>t = 5</th>
<th>t = 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>20%</td>
<td>aC</td>
<td>aC</td>
<td>aC</td>
<td>aC</td>
<td>aC</td>
<td>aC</td>
<td>aR</td>
<td></td>
</tr>
<tr>
<td>30%</td>
<td>aC</td>
<td>aC</td>
<td>aC</td>
<td>aC</td>
<td>aC</td>
<td>aC</td>
<td>aR</td>
<td></td>
</tr>
<tr>
<td>40%</td>
<td>aC</td>
<td>aC</td>
<td>aC</td>
<td>aC</td>
<td>aC</td>
<td>aC</td>
<td>aR</td>
<td></td>
</tr>
<tr>
<td>50%</td>
<td>aC</td>
<td>aC</td>
<td>aC</td>
<td>aC</td>
<td>a1/3a</td>
<td>aC</td>
<td>aR</td>
<td></td>
</tr>
</tbody>
</table>

Table 9.3: Sensitivity Analysis - Inspection Cost of the Rolls

POMDP strategies. In this analysis four percentages of reduction to the rolls inspection cost were selected. These are: 20%, 30%, 40%, 50%.

During each period of time the core process evolves according to the state transition matrices of the scenario. The policy selected at each period depends on the random quality observed and the updated state probabilities. Table 9.3 illustrates the differences in strategies to changes in the inspection cost of the rolls.

Replacement Cost

In this case, we specify five levels or percentages in which the replacement cost is reduced. These are: 10%, 20%, 30%, 40%, 45%.

Table 9.4 illustrates the differences in strategies to changes in the replacement cost.

Price of the Roll

To understand the sensitivity of the results to changes in roll cost, we specify some levels or percentages in which the price of the roll is changed. Table 9.5 presents the differences in strategies to changes in roll cost.
### Table 9.4: Sensitivity Analysis - Replacement Cost of the Rolls

<table>
<thead>
<tr>
<th>Scenario % of Reduction</th>
<th>Roll Repl. Cost</th>
<th>t = 0</th>
<th>t = 1</th>
<th>t = 2</th>
<th>t = 3</th>
<th>t = 4</th>
<th>t = 5</th>
<th>t = 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>10%</td>
<td>aC</td>
<td>aC</td>
<td>aC</td>
<td>aC</td>
<td>aC</td>
<td>aC</td>
<td>aR</td>
<td></td>
</tr>
<tr>
<td>20%</td>
<td>aC</td>
<td>aC</td>
<td>aC</td>
<td>aC</td>
<td>aC</td>
<td>aC</td>
<td>aR</td>
<td></td>
</tr>
<tr>
<td>30%</td>
<td>aC</td>
<td>aC</td>
<td>aC</td>
<td>aC</td>
<td>aR</td>
<td>aC</td>
<td>aR</td>
<td></td>
</tr>
<tr>
<td>40%</td>
<td>aC</td>
<td>aC</td>
<td>aC</td>
<td>aR</td>
<td>aC</td>
<td>aC</td>
<td>aR</td>
<td></td>
</tr>
<tr>
<td>45%</td>
<td>aC</td>
<td>aC</td>
<td>aC</td>
<td>aR</td>
<td>aC</td>
<td>aR</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 9.4: Sensitivity Analysis - Replacement Cost of the Rolls

### Table 9.5: Sensitivity Analysis - Roll Cost

<table>
<thead>
<tr>
<th>Scenario % of Increase</th>
<th>Roll Cost</th>
<th>t = 0</th>
<th>t = 1</th>
<th>t = 2</th>
<th>t = 3</th>
<th>t = 4</th>
<th>t = 5</th>
<th>t = 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>5%</td>
<td>aC</td>
<td>aC</td>
<td>aC</td>
<td>aC</td>
<td>aC</td>
<td>aC</td>
<td>aR</td>
<td></td>
</tr>
<tr>
<td>10%</td>
<td>aC</td>
<td>aC</td>
<td>aC</td>
<td>aC</td>
<td>aC</td>
<td>aC</td>
<td>aR</td>
<td></td>
</tr>
<tr>
<td>15%</td>
<td>aC</td>
<td>aC</td>
<td>aC</td>
<td>aC</td>
<td>aC</td>
<td>aC</td>
<td>aR</td>
<td></td>
</tr>
<tr>
<td>20%</td>
<td>aC</td>
<td>aC</td>
<td>aC</td>
<td>aC</td>
<td>aC</td>
<td>aC</td>
<td>aR</td>
<td></td>
</tr>
<tr>
<td>25%</td>
<td>aC</td>
<td>aC</td>
<td>aC</td>
<td>aR</td>
<td>aC</td>
<td>aC</td>
<td>aR</td>
<td></td>
</tr>
</tbody>
</table>

Table 9.5: Sensitivity Analysis - Roll Cost
As we can see, the results are relatively sensitive to changes in the rolls inspection cost, rolls replacement cost, and price of the rolls.

Following the same procedure, we found that the results are insensitive to changes in the operating cost, strip inspection cost, and grinding cost.

### 9.2 DETERMINATION OF THE SIZE OF THE PRODUCT BLOCK

The simulation program for the Hot Strip Mill, described in chapter 8, can be used to determine the optimal length of a Product Block. In order to do that, the size of the Product Block was changed with values that come from a given interval (minimum size, maximum size). 10 Product Blocks per each size were evaluated, and the decision time at which the maximum reward per slab is obtained is used to define the Product Block optimal size. The results are presented in table 9.6.

It can be seen in table 9.6 that the first decision time at which replacement of all stands is the optimal policy (RT) occurs at time 8, that is, after 160 slabs are produced. In addition, Product-Blocks with 140 slabs will allow the Company to have the maximum reward per slab.

The simulation can be used to obtain a precise result about the estimated best size of a Product Block. To do this, we evaluate Product Blocks with sizes between (141,160), where the decision time is represented for the time required to roll 1 slab. The decision time at which the reward per slab is maximum is $T = 155$, that is, after 155 slabs are produced. As a result, the
### 9.3 DETERMINATION OF THE FREQUENCY OF SAMPLING

The simulation can be used to propose a manner by which the frequency of sampling can be regulated, while taking into account the underlying probability distribution of rolls deterioration. The criterion for choosing the sampling plan is that the expected reward per product block should be maximized. We attempt to obtain a near optimal solution by imposing the following sort of restrictions on the lengths of sampling intervals:
1. The first sample of size 1 will be taken at time $t_{\text{min}}$ at which the Mill is not capable of obtaining better quality for the strips than the minimum accepted quality ($L$).

2. The following $L$ values are considered:

   (a) 3OF (strip surface with medium open spaced fleck)

   (b) 3CF (strip surface with medium close spaced fleck)

   (c) 3W (strip surface with wave pattern)

3. After the process reaches to the minimum accepted quality, a sample of size 1 is taken from the output of the process at a regular interval of every $h$ coils. The value $h$ represents the minimum possible time between two consecutives samples, and it is totally determined from the availability of the crane used to transport the coils from the mill to the inspection area. The company estimates $h$ as 15 coils.

We now discuss the procedures used to estimate $t_{\text{min}}$ from empirical data. These estimations are obtained for each value of $L$.

The data in Tables 9.7 - 9.9 represents the $t_{\text{min}}$ time on 233 Product Blocks, when the minimum accepted qualities are 3OF, 3CF, and 3W respectively. In this case, $t_{\text{min}}$ is represented by the number of produced coils at the time the process reaches the minimum accepted quality.

With the 233 samples of $t_{\text{min}}$ values per each minimum accepted quality,
### Table 9.7: $t_{\text{min}}$ Data - Minimum Accepted Quality 3OF

<table>
<thead>
<tr>
<th>$t_{\text{min}}$ INTERVAL (produced coils)</th>
<th>NUMBER OF PRODUCT BLOCKS</th>
</tr>
</thead>
<tbody>
<tr>
<td>70 ≤ $t_{\text{min}}$ ≤ 80</td>
<td>3</td>
</tr>
<tr>
<td>80 ≤ $t_{\text{min}}$ ≤ 90</td>
<td>9</td>
</tr>
<tr>
<td>90 ≤ $t_{\text{min}}$ ≤ 100</td>
<td>26</td>
</tr>
<tr>
<td>100 ≤ $t_{\text{min}}$ ≤ 110</td>
<td>141</td>
</tr>
<tr>
<td>110 ≤ $t_{\text{min}}$ ≤ 120</td>
<td>1</td>
</tr>
<tr>
<td>120 ≤ $t_{\text{min}}$</td>
<td>53</td>
</tr>
</tbody>
</table>

### Table 9.8: $t_{\text{min}}$ Data - Minimum Accepted Quality 3CF

<table>
<thead>
<tr>
<th>$t_{\text{min}}$ INTERVAL (produced coils)</th>
<th>NUMBER OF PRODUCT BLOCKS</th>
</tr>
</thead>
<tbody>
<tr>
<td>100 ≤ $t_{\text{min}}$ ≤ 105</td>
<td>4</td>
</tr>
<tr>
<td>105 ≤ $t_{\text{min}}$ ≤ 110</td>
<td>2</td>
</tr>
<tr>
<td>110 ≤ $t_{\text{min}}$ ≤ 115</td>
<td>11</td>
</tr>
<tr>
<td>115 ≤ $t_{\text{min}}$ ≤ 120</td>
<td>5</td>
</tr>
<tr>
<td>120 ≤ $t_{\text{min}}$</td>
<td>211</td>
</tr>
</tbody>
</table>

### Table 9.9: $t_{\text{min}}$ Data - Minimum Accepted Quality 3W

<table>
<thead>
<tr>
<th>$t_{\text{min}}$ INTERVAL (produced coils)</th>
<th>NUMBER OF PRODUCT BLOCKS</th>
</tr>
</thead>
<tbody>
<tr>
<td>105 ≤ $t_{\text{min}}$ ≤ 110</td>
<td>1</td>
</tr>
<tr>
<td>110 ≤ $t_{\text{min}}$ ≤ 115</td>
<td>4</td>
</tr>
<tr>
<td>115 ≤ $t_{\text{min}}$ ≤ 120</td>
<td>3</td>
</tr>
<tr>
<td>120 ≤ $t_{\text{min}}$</td>
<td>225</td>
</tr>
</tbody>
</table>
we performed a Chi-Square test to determine the probability distribution that best fits these sample values. Using Minitab, we found that the best fit for each sample was a Weibull distribution with the parameters shown in Table 9.10.

The parameters were estimated using the maximum likelihood method. The Chi-Square test results are in Table 9.11:

For \( \alpha \) values less than \( p \) value, we do not reject the fact that the samples have Weibull distributions with the parameters described in Table 9.10. We can see the threshold parameters as point estimators of the \( t_{\text{min}} \) per each minimum accepted quality. From these results we conclude that our sampling plans are the following:

1. **Sampling Plan when the Minimum Accepted Quality is 3OF (3OF-SP)**

   The first sample of size 1 will be taken at time \( t_{\text{min}} = 74 \) at which the Mill
is not able to obtain better quality than the minimum accepted quality 3OF. After that, a sample of size 1 is taken from the output of the process at a regular interval of every 15 coils.

2. Sampling Plan when the Minimum Accepted Quality is 3CF (3CF-SP)
The first sample of size 1 will be taken at time \( t_{min} = 102 \) at which the Mill is not able to obtain better quality than the minimum accepted quality 3CF. After that, a sample of size 1 is taken from the output of the process at a regular interval of every 15 coils.

3. Sampling Plan when the Minimum Accepted Quality is 3W (3W-SP)
The first sample of size 1 will be taken at time \( t_{min} = 107 \) at which the Mill is not able to obtain better quality than the minimum accepted quality 3W. After that, a sample of size 1 is taken from the output of the process at a regular interval of every 15 coils.

By using simulation, the results obtained with each sampling plan are compared against the results obtained by the current hot strip mill sampling plan (CURRENT-SP). Results over 233 Products-Blocks are shown in Table 9.12.
### Table 9.12: Comparison of performance between proposed and current Sampling Plans

<table>
<thead>
<tr>
<th>SAMPLING PLAN</th>
<th>REWARD PER SLAB</th>
</tr>
</thead>
<tbody>
<tr>
<td>3OF-SP</td>
<td>2578.75</td>
</tr>
<tr>
<td>3CF-SP</td>
<td>2592.75</td>
</tr>
<tr>
<td>3W-SP</td>
<td>2594.15</td>
</tr>
<tr>
<td>CURRENT-SP</td>
<td>2563.25</td>
</tr>
</tbody>
</table>

9.4 EVALUATION OF THE CURRENT HOT STRIP MILL OPERATING PROCEDURE

The simulation can be used to compare the hot mill operation by our proposed procedure against the current procedure.

Our proposed procedure includes the sampling plan 3W-SP, and an evaluation at each decision time of each given action in terms of costs. The current procedure tracks the deterioration of the rolls using the product observations every 20 slabs. If the quality of the product is 3W or worse, a decision is made to replace all the rolls.

To perform this evaluation, we generate a program that follows their current procedure, and uses Product Blocks sizes in the interval [100,180]. Results are shown in table 9.13.

The proposed procedure represents an average increment of 15.3% in the reward per slab produced.
Table 9.13: Comparison of performance between proposed and current control procedure
Chapter 10

TOPICS FOR FUTURE RESEARCH

In this chapter, we discuss some of our ideas for future investigation into solution techniques for POMDP and POSMDPs, with particular emphasis on the use of the methods developed in this dissertation.

This chapter contains two sections:

Section 1 Presents a new heuristic algorithm to determine good approximate solutions to the partially observed semi-Markov optimization problem. The mathematical formulation of POSMDP and two algorithms to solve the partially observed semi-Markov optimization problem are discussed. The first algorithm was developed by C.C. White [51](1974). He has modified the One Pass algorithm so that it can be applied to partially observed semi-Markov processes. The second algorithm is a generalization of the heuristic algorithm proposed for the POMDP to solve the
POSMDPs. The new heuristic is under this chapter because we do not compare its performance with other solution procedures.

Section 2 Presents other recommendations for future research.

10.1 POSMDP PROPOSED HEURISTIC ALGORITHM

In chapter 3, it was assumed that the core process transition, observation, and time of control occurred at each time interval. Of course, many problems for which the transition dynamics are Markovian feature state transitions at random intervals. In these cases, the problems are more validly modeled as a semi-Markov process than as a Markov process.

10.1.1 PROBLEM DEFINITION

This section is concerned with the presentation of fundamental concepts and results associated with the partially observed semi-Markov optimization problem and its related cost evaluation problem.

In this section, the assumption made in Chapter 3 is detached; that is, the time between core process transition and time between observations are considered to be random variables, and the ratio of the number of transitions to the number of observations is not necessarily one to one.

In general, we can say that the model being studied in this chapter has the following characteristics:
1. The core process is a controlled semi-Markov process.

2. Observations can be made to provide partial information about the core process.

3. Times of observations and times of core process transition do not necessarily occur simultaneously.

4. Control occurs at the time of observations.

COMPONENTS OF THE POSMDP's

A partially observable semi-Markov decision control process possesses the following components:

THE CORE PROCESS

The core process is an underlying semi-Markov process. It is characterized by the states labeled $1, 2, ..., N_s$.

The process makes a transition from one state to the next at time $t_q$, where $t_q$ is the $q^{th}$ time of transition.

The sequence of integer-valued random variables $(t_q, q = 0, 1, ...)$ is called the transition time process, which has the following properties:

\[
\begin{align*}
t_0 & \leq 0 \\
t_1 & > 0
\end{align*}
\]
The last assumption is included for convenience, but it can be omitted with a resulting increase in complexity. $K$ represents the largest possible time of transition. The controller is assumed to know the times of core process transition; however, the state of the core process can not be observed directly. The controller sees instead one of the $N_z$ states of the observation process.

**THE OBSERVATION PROCESS**

This process is observed at observation times $\tau_r$, where $\tau_r$ is the time of the $r^{th}$ observation. The sequence of integer valued random variables $(\tau_r, r = 0, 1, \ldots)$ is called the observation time process, which has the following properties:

\[
\begin{align*}
\tau_0 & = 0 \\
\tau_r & > 0 \\
\tau_r & < \tau_{r+1} \quad \forall r \quad r = 1, 2, \ldots
\end{align*}
\]

We assume that the observation process has $N_z$ states labeled $1, 2, \ldots, N_z$, and that $z(\tau_r)$ represents the state of the observation process at time $\tau_r$. The core process and the transition time process can be described as follows:

Let:

\[\sigma(t) = \max(\tau_r : \tau_r \leq t)\]
Then:

1. If core process transition has occurred after the last time of observation \((\sigma(t_{q+1}) \leq t_q)\)

\[
\begin{align*}
    s(t_{q+1}) &= f_s[s(t_q), a(\sigma(t_{q+1})), \mu_s(t_q)] \\
    t_{q+1} - t_q &= f_q[s(t_q), s(t_{q+1}), a(\sigma(t_{q+1})), \mu_q(t_q)]
\end{align*}
\]

(10.3)

where, processes \(\mu_s(t), \mu_q(t)\) are sequences of independent in time, identically distributed random variables with known distributions.

2. If observation time has occurred after the last time of core process transition \((\sigma(t_{q+1}) > t_q)\)

\[
\begin{align*}
    s(t_{q+1}) &= f_s[s(t_q), \sigma(t_{q+1}) - t_q, a(\sigma(t_{q+1})), \mu_s(\sigma(t_{q+1}))] \\
    t_{q+1} - \sigma(t_{q+1}) &= f_q[s(t_q), s(t_{q+1}), \sigma(t_{q+1}) - t_q, a(\sigma(t_{q+1})), \mu_q(\sigma(t_{q+1}))]
\end{align*}
\]

(10.4)

In POMDP \(f_q = 1\).

At each decision time and core process transition, equations 10.3 and 10.4 have to be reevaluated.

The observation time process and the observation process are described by the following stochastic difference equations:

\[
\begin{align*}
    \tau_{r+1} &= \tau_r + f_r[s(\tau_r), a(\tau_r), \mu_r(\tau_r)] \\
    z(\tau_{r+1}) &= f_z[s(\tau_{r+1}), a(\tau_r), \mu_z(\tau_r)]
\end{align*}
\]

(10.5)
It is assumed that, at most, one transition and one observation can be made per unit interval of time (\( f_q \) and \( f_r \) are strictly positive for all arguments, with probability one).

The processes \( \mu_r(t) \), \( \mu_z(t) \) are sequences of independent in time, identically distributed random variables with known distributions.

In the POMDP \( f_r = 1 \).

If:

1. for each \( i \) and \( a \), \( f_q[i, j, a, \mu_q(t)] = f_r[i, a, \mu_r] \) for all \( j \).
2. for each \( j \) and \( a \), \( f_z[i, a, \mu_z] = j \)

we have the completely observed semi-Markov optimization problem.

THE STATE OF KNOWLEDGE

The probability that the core process is in a particular state at time \( t \), where \( t \in \{ \tau_r : t_q \leq \tau_r \leq t_{q+1} \} \), can be calculated using the semi-Markov property and basic probability relations.

If we assume that \( \xi(t) \) contains all the information concerning the state of the process up to time \( t \), then a portion of this information can be encoded in the form of a \( K \times N_s \) matrix \( \pi(t) \).

If we define the function \( \omega \) as:

\[
\omega(t) = t - \max(t_q : t_q \leq t)
\]

Then, the matrix \( \pi(t) \) is defined as:

\[
\pi(t) = [\pi_{ki}(t)] = [pr(\omega(t) = k, s(t) = i|\xi(t), t = \tau_r)] \forall i \in S_s, k \in(0, ..., K-1)
\]
Where:

\[
\pi_{ki} \geq 0 \\
\sum_k \sum_i \pi_{ki} = 1
\]

(10.6)

Let \( \Pi \) be the space of all \( K \times N_s \) matrices \( \pi \), that is:

\[
\Pi = \text{set}\{ \pi = [\pi_{ki}] : \pi_{ki} \geq 0, \sum_k \sum_i \pi_{ki} = 1 \}
\]

Let \( \pi(0) = [\pi_{ki}(0)] \), the probability matrix of the initial state of the system, be known by the decision maker. As a result, \( \xi(t) \) is a function of the given initial density matrix \( \pi(0) \), the sequence of previously determined control vectors up to time \( t \), the sequence of times of observation up through time \( t \), and the sequence of observation outcomes up through time \( t \).

We shall refer to \( \pi(t) \) as the state of knowledge of the partially observable semi-Markov process. Note that \( \xi(t) \) contains all past information, and that \( \pi(t) \) is only a partial representation of that information. However, \( \pi(t) \) serves as a sufficient statistic (see [51]).

To determine the dynamics of \( \pi(t) \), let us define the following conditional distributions:

**The transition probability** \( p_{ij}^k(a) \)

\[
p_{ij}^k(a) = \Pr( s(t_{q+1}) = j | s(t_q) = i, t - t_q = k, a(\sigma(t)) = a ) \\
\sum_j p_{ij}^k(a) = 1 \quad \forall i, k, a
\]

(10.7)

**The holding time probability** \( h_{ij}^k(m, a) \)

\[
h_{ij}^k(m, a) = \Pr( t_{q+1} - t = m | s(t_q) = i, s(t_{q+1}) = j, \\
\]

...
\[ t - t_q = k, a(\sigma(t)) = a \]

\[ \forall t \in [t_q, (\tau_r : t_q \leq \tau_r < t_{q+1})] \quad m, k \in (0, 1, \ldots, K) \]

\[ \sum_j \sum_m h_{ij}^k(m, a) = 1 \quad \forall i, k, a \]

\[ (10.8) \]

The observation probability \( q_jz(a) \)

\[ q_jz(a) = pr(z(\tau_{r+1}) = z|s(\tau_{r+1}) = j, a(\tau_r) = a) \]

\[ \sum_z q_jz(a) = 1 \quad \forall j, a \]

\[ (10.9) \]

The observation time probability \( b_i(m, a) \)

\[ b_i(m, a) = pr(\tau_{r+1} - \tau_r = m|s(\tau_r) = i, a(\tau_r) = a) \]

\[ \sum_m b_i(m, a) = 1 \quad \forall i, j, a \]

\[ (10.10) \]

The relationships between the probabilities \( p_{ij}(a) \) and \( h_{ij}^k(m, a) \) are provided by C.C. White [51] pages 54-56.

In the absence of other information, \( \xi(\tau_{r+1}) \) as a function of \( \xi(\tau_r), \tau_{r+1}, a(\tau_r), z(\tau_{r+1}) \) can be encoded as the vector \( \pi(\tau_{r+1}) \).

Let:

\[ \gamma_{ij}^m(k, l, a) = pr[s(\tau_{r+1}) = j, \omega(\tau_{r+1}) = l, \tau_{r+1} - \tau_r = m|s(\tau_r) = i, \omega(\tau_r) = k, a(\tau_r) = a] \]

\[ \tilde{\gamma}_{ij}^m(k, l, a) = pr[s(\tau_{r+1}) = j, \omega(\tau_{r+1}) = l|s(\tau_r) = i, \tau_{r+1} - \tau_r = m, \omega(\tau_r) = k, a(\tau_r) = a] \]

\[ (10.11) \]
which are related by the expression:

\[ \gamma_{ij}^m(k, l, a) = b_i(m, a) \hat{\gamma}_{ij}^m(k, l, a) \]  

(10.12)

where:

\[ \hat{\gamma}_{ij}^1(k, k + 1, a) = \sum_{j=1}^{N_x} p_{ij}^k(a) \sum_{m=2}^{K-1} h_{ij}^k(m, a) \]

\[ \hat{\gamma}_{ij}^1(k, 0, a) = p_{ij}^k(a) h_{ij}^k(1, a) \]  

(10.13)

\[ \forall i, j, a, 0 \leq k \leq K - 1, \text{ where for all other possibilities } \hat{\gamma}_{ij}^1(k, l, a) = 0 \]

\[ \hat{\gamma}_{ij}^{m+1}(k, 0, a) = \sum_{\alpha=1}^{N_x} \sum_{\beta=1}^{k-1} \hat{\gamma}_{i\alpha}^m(k, \beta, a)p_{\alpha j}^\beta(a)h_{\alpha j}^\beta(1, a) \]

\[ \hat{\gamma}_{ij}^{m+1}(k, l + 1, a) = \hat{\gamma}_{ij}^m(k, l, a) \sum_{\alpha=1}^{N_x} p_{j\alpha}^l(a) \sum_{\beta=2}^{k-1} h_{j\alpha}^\beta(\beta, a) \]  

(10.14)

When either \( l > k + m \) for \( i = j \) or \( l \geq m \) for \( i \neq j \) then \( \hat{\gamma}_{ij}^m(k, l, a) = 0. \)

The probability that \( [s(\tau_{r+1}) = j, \omega(\tau_{r+1}) = l] \) given \( \xi(\tau_{r+1}) \), where \( \xi(\tau_{r+1}) \) is a function of \( \xi(\tau_r), \tau_{r+1} - \tau_r, z(\tau_{r+1}) \), can be obtained as follows by using Bayes' rule:

If we assume that for a specific \( a(\tau_r) = a, \omega(\tau_r) = k \)

\[ pr(s(\tau_{r+1}) = j, \omega(\tau_{r+1}) = l|\xi(\tau_{r+1}) = f(\xi(\tau_r), \tau_{r+1} - \tau_r = m, z(\tau_{r+1}) = z) = T_{ij}(\pi|z, m, a, k) \]  

(10.15)
Then:

\[
T_{ij}(\pi|z, m, a, k) = \frac{\sum_k \sum_i q_{ij}(a) \gamma_{ij}^m(k, l, a) \pi_{ki}}{\sum_i \sum_u \sum_k \sum_v q_{uv}(a) \gamma_{iu}^m(k, v, a) \pi_{ki}}
\]  
(10.16)

Proof:

\[
T_{ij}(\pi|z, m, a, k) =
\]

\[
pr(s(\tau_{r+1}) = j, \omega(\tau_{r+1}) = l, \tau_r + 1 = \tau_r = m, z(\tau_{r+1}) = z|\xi(\tau_r))
\]

\[
pr(z(\tau_{r+1}) = z, \tau_r + 1 = \tau_r = m|\xi(\tau_r))
\]

\[
T_{ij}(\pi|z, m, a, k) = \frac{\sum_k \sum_i q_{ij}(a) \gamma_{ij}^m(k, l, a) \pi_{ki}}{\sum_i \sum_u \sum_k \sum_v q_{uv}(a) \gamma_{iu}^m(k, v, a) \pi_{ki}}
\]  
(10.17)

(10.18)

The matrix of these probabilities can be represented by:

\[
T(\pi|z, m, a, k) = [T_{ij}(\pi|z, m, a, k)]
\]

The use of \(T\) emphasizes that \(T(\pi|z, m, a, k)\) is a transformation of one point of \(\Pi\) into another.

\(T(\pi|z, m, a, k)\) will generally be written as follows in terms of matrices:

Let:

1. the matrix \(\Gamma^m_i(a)\) be defined so that the \((k, i)\) element of this matrix is \(\gamma_{ij}^m(k, l, a)\).

2. the matrix \(\Gamma^m(a)\) be defined as having the matrix \(\Gamma^m_i(a)\) as its \((l, j)\) element.

3. the matrix \([\pi, \Gamma^m(a)]\) be defined so that the \((l, j)\) element of this matrix is the scalar \(\sum_k \sum_i \gamma_{ij}^m(k, l, a) \pi_{ki}\).

As a result, the matrix \([\pi, \Gamma^m(a)]R_z(a)\) has as its \((l, j)\) element the \((l, j)\) term of the numerator of \(T(\pi|z, m, a, k)\).
CHAPTER 10. TOPICS FOR FUTURE RESEARCH

Let \([1]\) be the \(K \times N_s\) matrix with 1 in all entries. Then:

\[
T(\pi \mid z, m, a, k) = \frac{[\pi, \Gamma^m(a)]R_z(a)}{[1, \Gamma^m(a)]R_z(a)} \tag{10.19}
\]

Where \([...]\) represents the inner product. This product is defined as:

\[
[C, D] = \sum_i \sum_j c_{ij}d_{ij}
\]

Where: \(C = (c_{ij})\) and \(D = (d_{ij})\) are matrices with the same dimension.

THE REWARD STRUCTURE

Suppose we are interested in the expected reward of running the process for \(T\) time periods, when there is an initial state of knowledge \(\pi(0)\) and the process satisfies equations 10.7 to 10.14. Suppose, furthermore, that the reward includes the following terms:

1. The reward of the core process during the interval \((t_q + m, t_q + m + 1)\)
   \(0 \leq m < t_{q+1} - t_q\).
   This reward is represented by \(y[s(t_q), s(t_{q+1}), m, a(\sigma(t_q + m))]\).

2. The reward originated in the core process at the time of transition from state \(s(t_q)\) to state \(s(t_{q+1})\).
   This reward is represented by \(d[s(t_q), s(t_{q+1}), m, a(\sigma(t_{q+1} + 1))]\), where \(m = t_{q+1} - t_q\).

3. The terminal reward incurred at time \(T\), which is represented by \(g^T_{s(T)}\).
4. The reward incurred at time \( r_r \), which is represented by \( c(a) \). Where,
\[
a = a(r_r).
\]

We now present a recursive relationship for the expected reward over time that was developed by Chelsea C. White [51].

Let:

\[
C^t_a(\pi(t)) \text{ represents the expected reward incurred from time } t \text{ to the end of the decision process.}
\]

\( g^{ki}_t(a) \) represents the expected reward incurred in the interval \((0,t]\), neglecting observation and terminal reward, given \( s(0) = i \), \( \omega(0) = k \), and that alternative \( a \) is selected over the entire interval.

\( g^{ki}_t(a) \) represents the expected reward incurred from observation time \( t \) up to either the next time of observation or up through the terminal time if \( t = r_r \); where: \( \bar{r} = \max(r : r_r \leq T) \); \( s(t) = i \); \( \omega(t) = k \); \( a(t) = a \).

Let:

\[
\begin{align*}
\tilde{g}^{ki}_t(a) &= 0 \\
\tilde{g}^{ki}_t(a) &= \sum_{m=1}^{t} \sum_{j} p_{ij}^k(a)h_{ij}^k(m,a) \\
&\quad \cdot \left[ \sum_{\beta=k}^{k+m-1} y(i,j,\beta,a) + d(i,j,k+m,a) + \tilde{g}^{0j}_{i-m}(a) \right] \\
&\quad + \sum_{m=t+1}^{K} \sum_{j} p_{ij}^k(a)h_{ij}^k(m,a) \sum_{\beta=k}^{k+t-1} y(i,j,\beta,a)
\end{align*}
\]
Let:

\[ g_{t}^{ki}(a) = c(a) + \sum_{m=1}^{t-1} b_i(m,a)\hat{g}_{m}^{ki}(a) \]
\[ + \sum_{m=t}^{\infty} b_i(m,a)[\hat{g}_{t}^{ki}(a) + \sum_{j} \sum_{l} \gamma_{ij}^t(k,l,a)g_{j}^{l}] \]

(10.21)

If we define the \( K \times N \) matrix \( G_t(a) = (g_{t}^{ki}(a)) \), a recursive equation for \( C^t_a(\pi(t)) \) can be expressed as:

\[ C^t_a(\pi(t)) = [\pi, G_t(a)] + \sum_{m=1}^{T-t} \sum_{z} \sum_{m} V(z,m,\pi,a)C^{t+m}_a(T(\pi|z,m,a,k)) \]
\[ C^T_a(\pi) = [\pi, G_T] \]

(10.22)

Where:

\[ V(z,m,\pi,a) = \text{pr}(z(\tau_{r+1}) = z, \tau_{r+1} - \tau_r = m|\pi(\tau_r)) \]
\[ = \sum_{i} \sum_{j} \sum_{k} \sum_{l} q_{jz}(a)\gamma_{ij}^m(k,l,a)\pi_{ki} \]

(10.23)

**THE CONTROL PROBLEM** The control problem can be formulated as follows, so that the optimal control can be found as an explicit function of the state of knowledge \( \pi \).

\[ C^t(\pi(t)) = \max_{a}[[\pi, G_t(a)] + \]
10.1.2 POSMDP PROPERTIES

To develop a generalization of Sondik's computational procedure to solve a semi-Markov decision process with continuous state space, the following properties of the reward function $C^t(\pi(t))$ are used (see C.C. White [51]):

1. The optimal reward function is concave.

2. The optimal reward function is piecewise linear.

10.1.3 COMPUTATIONAL EXPRESSIONS FOR $C^t(\pi(t))$

These properties allow us to represent the optimal reward function in the form:

$$C^t(\pi(t)) = [\pi, \alpha^t(\pi(t))]$$  \hspace{1cm} (10.25)

Where $\alpha^t$ is a piecewise constant matrix over $\pi$.

The piecewise constancy of $\alpha^t$ allows us to partition the state space $\Pi$ into sets, and deal with a state space consisting of a set of points rather than a continuum.

Assume that $\Lambda^t = (\alpha^t_1, \alpha^t_2, \ldots)$ is the finite set of all constant matrix values of $\alpha^t$. 

\[
C^T(\pi) = \left[ \pi, G_T \right]
\]  

(10.24)
Let:

$\tilde{R}_k^t$ be the region in $\Pi$ where $\alpha^t$ has the matrix value $\alpha_k^t$.

$\tilde{R}^t$ be the associated partition of $\Pi$.

Then:

$$\tilde{R}_k^t = \{ \pi | C^t(\pi) = (\pi)\alpha_k^t \}$$

$$\tilde{R}^t = \bigcup_k \tilde{R}_k^t$$

$C^t(\pi)$ is completely determined by the set $\Lambda^t$.

Associated with each $\alpha_k^t$ in $\Lambda^t$ is a region of $\Pi$, $\tilde{R}_k^t \subset \tilde{R}^t$.

For each $t$, there exists a finite set $\Lambda^t$ of vectors such that:

$$C^t(\pi) = \max\{[\pi, \alpha^t(\pi) : \alpha^t(\pi) \in \Lambda^t]\}$$

Assume that $\Lambda^{t+m}$ has been determined for $m = 1, \ldots, T - t$. Thus for $m = 1, \ldots, T - t$.

$$C^t(\pi) = \max_a \{ [\pi, G_t] + \sum_m \sum_z \max_j \{ [\pi, \Gamma^m(a)] R_z(a), \alpha_{j}^{t+m} \} \}$$

Section 10.1.4 presents some techniques for determining $\alpha^t$ from $\alpha_{j}^{t+m}$, and consequently for determining the optimal control for time $t$. 
10.1.4 POSMDP SOLUTION TECHNIQUES

Two algorithms to solve the partially observed semi-Markov optimization problem are presented in this section. The first algorithm, developed by C.C. White [51], generalizes Sondik’s algorithm for the partially observed Markov optimization problem. The second algorithm uses our approach to alleviate the computational difficulties associated with this problem.

GENERALIZATION OF THE ONE-PASS ALGORITHM

C.C. White developed this algorithm for solving POSMDP problem as a generalization of Sondik’s One-Pass algorithm.

Taking into consideration that $C^t$ is piecewise linear and concave, $C^t$ is written as:

$$C^t(\pi(t)) = [\pi, \alpha^t(\pi(t))]$$

(10.26)

Where $\alpha^t$ is a piecewise constant matrix over $\pi$.

The piecewise constancy of $\alpha^t$ allows us to partition the state space $\Pi$ into sets, and deal with a state space consisting of a set of points rather than a continuum.

Assume that $\Lambda_{t+m}^{t+m}$ has been determined for $m = 1, ..., T - t$, where $\Lambda_{t+m}^{t+m}$ represents the set of all constant matrix values ($\alpha_{1}^{t+m}, \alpha_{2}^{t+m}, ...$). It follows that the set $\Lambda^t$ can be determined from $\Lambda_{t+m}^{t+m}$ in the following manner:

$$C^t(\pi(t)) = [\pi, G_t(a)] +$$

$$\sum_{m=1}^{T-t} \sum_{z} V(z, m, \pi, a) C_{a}^{t+m}(T|z, m, a, k)$$
\[ C^t(\pi(t)) = [\pi, G_t(a)] + \sum_{m=1}^{T-t} \sum_z V(z, m, \pi, a) \max_j \{ [T(\pi|z, m, a, k), \alpha_j^{t+m}] \} \]

\[ C^t(\pi(t)) = [\pi, G_t] + \sum_m \sum_z \max_j \{ [\pi, \Gamma^m(a)]R_z(a), \alpha_j^{t+m}] \} \]

Then, the recursive equation used to calculate the function \( C^t(\pi) \) is reduced to:

\[ C^t(\pi(t)) = \max_a \{ [\pi, G_t(a)] + \sum_{m=1}^{T-t} \sum_z [\pi, \Gamma^m(a)]R_z(a), \alpha_j^{t+m} \} \]

\[ C^T(\pi) = [\pi, G_T] \]

(10.28)

Where:

\( l(\pi, a, z, m) \) denotes the subscript \( k \) of the set \( \Lambda_k \) that maximizes:

\[ T(\pi|z, m, \pi, a)\alpha_k^{t+m} \]

C.C. White generalized Sondik's algorithm for finding the optimum \( a \)-vectors and corresponding control alternatives with the following modifications:

1. \( l(\pi, a, z, m) \) is calculated from the following equation:

\[ l(\pi, a, z, m) = \{ k : \max_k \{ [\pi, \Gamma^m(a)]R_z(a) \alpha_k^{t+m} \} \} \]

2. The optimal \( a \)-matrices, \( a^* \), and corresponding alternatives are calculated by using \( l(\pi, a, z, m) \) values and equation:

\[ C^t(\pi(t)) = \max_a \{ [\pi, G_t(a)] + \]
3. To identify the region of the information space over which $\alpha^*$ is optimal and to identify the $\alpha$-matrices of the neighboring regions, the algorithm consider the following constraints:

$$[\pi, (\alpha^{t+m}_{l(p,a,z)} - \alpha^{t+m}_l) R_z(a^*) \Gamma^m(a^*)] \geq 0 \quad \forall z, l, m$$

$$[\pi, a_l^* - a_l^a] \geq 0 \quad \forall a$$

This procedure and the One-Pass Algorithm differ in that for the Markov case $l(p,a,z)$ need to be determined only for $m = 1$, whereas for the semi-Markov case $l(p,a,z,m)$ must be determined for $m = 1, ..., T$

POSMDO PROPOSED HEURISTIC ALGORITHM

This algorithm is a generalization of the heuristic algorithm proposed for the partially observed Markov process, and only differs in that for the Markov case strategies need only be determined at each observation time, whereas for the semi-Markov case strategies must be determined not only at each observation time, but also at each possible $m$ value.

Determination of an optimal control policy can be performed in two basic parts:

1. Actuator part.

2. Estimator part.
The Actuator Part

The procedures of this part are running out off-line. As we said in chapter 3, optimal controls based on $\pi$ eliminate the need for storing past histories of varying lengths. Using this fact and the standard "Principle of Optimality" of dynamic programming, Bellman [5]1957, we arrive at the following reward equation:

$$C^t(\pi(t)) = \max_a \{[\pi, G_t(a)] + \sum_{m=1}^{T-t} \sum_{z} V(z, m, \pi, a) C_{a}^{t+m}(T(\pi|z, m, a, k))\}$$

$$C^T(\pi) = [\pi, G_T]$$

(10.30)

As we said before, in this representation, the state variable is assumed to be known exactly; hence, the above equation is the recursive equation of a completely observable semi-Markov Decision Process. In other words, with this equation, we convert the POSMDP into an equivalent (completely observable) semi-Markov Decision Process. The associated SMDP for a POSMDP is the SMDP which shares the reward structure and probabilistic dynamics of the POSMDP, and which has perfect state information at each stage.

To avoid the computational intractability of the problem described above, this algorithm assumes that although the state of the core process is not known at time $t$, the state of the core process is known with certainty at any discrete time after time $t$. Under this assumption, the Actuator part uses Dynamic Programming to obtain the maximum expected reward that can be made during the time horizon, with the following recursion equation, under the assumption that the core process state after time $t+1$
is known.

Let:

\( CT^t_{ki} \) represents the optimal total reward obtained from time \( t \) to the end of the decision process, when \( s(t) = i, \omega(t) = k \)

\( CT^t \) represents a \( K \times N_s \) matrix \( CT^t = [CT^t_{ki}] \).

\[
CT^t_{ki} = \max_{a \in A} \left[ g^t_{ki} + \sum_{m=1}^{T-t} \sum_{z_i} \sum_{j} \sum_{l} q_{ji}(a) \gamma^m_{ij}(k, l, a) CT^t_{kj} \right]
\]

\[
CT^T_{ki} = g^T_{ki}
\]

Equation [10.30] was obtained from equation [10.29] under the assumption that the state of the core process is known at any discrete time after time \( t \).

Dynamic programming is used to solve the problem represented by the equation [10.30]. The tables obtained are stored to be used after the matrix \( \pi \) is calculated (Step 2 of the Estimator part). This part of the algorithm has the following advantages:

1. Given that the procedures of this part are running out of line, the Actuator part significantly reduces the amount of real time computations required in the implementation of the optimal controlled.

2. It assures an efficient On-Line implementation, involving polynomial computational time.
3. It guarantees that memory requirements do not exceed limits.

**The Estimator Part**

This part runs in parallel with the process. The results from the Actuator part are used in Step 2 to select the policy with greatest expected reward. On-Line part includes the following operations:

**Step 1**
Set $t = 0$.
Time $t = 0$ has associated a probability matrix of the initial state of the process $\pi(0) = [\pi_{ki}(0)]$, and an optimal policy $\delta_0$, which represents the policy used at the initialization of the process.
The expected immediate reward of the next transition is calculated as:

$$[\pi_0, G_0(\delta_0)]$$

Where:

$[..]$ represents the inner product. This product is defined as:

Let $C = (c_{ij})$ and $D = (d_{ij})$ be matrices with the same dimension.

$$[C, D] = \sum_i \sum_j c_{ij}d_{ij}$$

**Step 2**

$t = t + 1$
If $t = T$ go to Step 4, otherwise:
Observe $z(t) = z$.
Compute the matrix $\pi(t)$ from:
CHAPTER 10. TOPICS FOR FUTURE RESEARCH

\[ T(\pi(t)\mid z, m, \delta_{t-1}, k) = \frac{[\pi(t - 1), \Gamma^m(\delta_{t-1})]R_z(\delta_{t-1})}{[[1], [\pi(t - 1), \Gamma^m(\delta_{t-1})]R_z(\delta_{t-1})]} \]  

(10.32)

Where:

\[ ... \] represents the inner product.

\[ [1] \] represents the \( K \times N \), matrix with 1 in all entries.

\( z \) represents the output produced for the observation process at time \( t \).

\( \delta_t \) for an specific value of \( k \), represents the action with biggest expected reward of operating the process from time \( t \) to the end.

That is:

\[ \delta_t = \{ a^* \in A(t) \mid \pi_k(t)CT_k(a^*) \geq \pi_k(t)CT_k(a) \quad \forall \ a \in A(t) \ and \ k \} \]

**Step 3**

The expected immediate reward of the next transition is determined as:

\[ [\pi_t, G_t(\delta_t)] \]

Go to Step 2.

**Step 4**

Compute the matrix \( \pi(T) \) from:

\[ T(\pi(T)\mid z, m, \delta_{T-1}, k) = \frac{[\pi(T - 1), \Gamma^m(\delta_{T-1})]R_z(\delta_{T-1})}{[[1], [\pi(T - 1), \Gamma^m(\delta_{T-1})]R_z(\delta_{T-1})]} \]  

(10.33)
The expected final reward is determined as:

$$[\pi_T, G_T(\delta_T)]$$

10.2 OTHER RECOMMENDATIONS FOR FUTURE RESEARCH

1. In our definition of POMDP, we assumed that problem data such as transition probabilities, observation probabilities, and costs of actions were known. However, in many situations, the phenomena we wish to model as POMDPs involve probabilistic behavior which we do not know perfectly, or which may change over time. Therefore, it would be useful to have techniques which can accommodate parameter imprecision.

2. The extension of these results to the infinite horizon POMDP, and the infinite horizon POSMDP could be extremely useful in many applications.

3. A very important class of problems modelled as POMDP is the machine maintenance / quality control / optimal replacement problem. Much of the research into these problems deals with the issue of specially structured optimal strategies. The article by Monahan [28](1982) examines some of the results of this research. In [55], White proved conditions which guaranteed the existence of strategies known as "control-limit" for a replacement problem modelled as a POMDP. It is possible we can modify the POMDP solution techniques of this dissertation in order to take advantage of the special structure of optimal strategies for this problem.
4. Specific algorithms appropriate for problems with special structure are likely to perform significantly better than generic methods. It would be a significant contribution to develop solution methods that exploit the special structure in a specific problem class.

5. It will be interesting to provide an exploratory comparison between the performance of our proposed heuristic algorithm for POSMDP and other solution procedures.
Chapter 11

CONTRIBUTIONS AND CONCLUSIONS

This chapter concludes the present study. It discusses the most important contributions of the dissertation.

The major contributions are the following:

1. A new heuristic to solve the partially observed Markov problems has been developed. The most important feature of this heuristic is that it can find solutions which are close to optimal ones in a minimum time, and it can solve problems with large number of state in the core process, the observation process, the time horizon, and the alternatives considered.

2. Solution procedures for the POMDP’s are compared. This comparison is based on two types of numerical evaluations: CPU Time and Expected
Reward. Test problems were separated into four basic groups:

(a) The first group contains several sets of data without any special assumption about problem structure.

(b) The second group consists of "close-to-perfect information" POMDPs.

(c) The third group considers the special case of the completely unobserved Markov decision process (CUMDP).

(d) The fourth group considers the case when the measured output $z(t)$ will coincide with the state of the core process $s(t)$ with probability one.

It is clear from the numerical examples that our proposed heuristic algorithm always performs faster than others to find an approximate expected cost efficiently, and its performance takes advantage of the available information.

3. We have described a POMDP that arises in a steel company, and formulated it as a control problem. Since this problem is hard to solve to optimality, we have used our proposed heuristic to approach this problem.

4. We determine the optimal size of a Product Block. The new size of the Tin Plate's Product Block is 155 slabs.

5. We propose a method by which the frequency of sampling can be regulated, while taking into account the underlying probability distribution
of roll deterioration. The criterion for choosing the sampling plan is that the expected reward per product block should be maximized.

6. We compare the performance between the proposed and the current procedure. The proposed procedure represents an average increment of 15.3% in the reward per slab produced.

7. We have also formulated and analyzed the more general partially observed semi-Markov optimization problem. The computational procedure generated by C.C. White [51], which generalizes a numerical algorithm for the partially observed Markov optimization problem due to Sondik, is presented. Several preliminary results were proved which justify the basic approach of the computational procedure. A new heuristic for the POMDP was developed. This heuristic generalizes our proposed heuristic to solve the POMDP.

From the results and analysis presented, we can conclude that although the heuristic method proposed for POMDP does not find the optimal solution, the results produced represent an impressive improvement over the current procedures.
REFERENCES


REFERENCES


REFERENCES


REFERENCES


Appendix A

TEST DATA FOR FINITE HORIZON ALGORITHMS

The following sets of data are generated to compare the efficiency of the algorithms discussed in previous chapters. As we said before, these problems are separated into four groups:

1. Group 1 consists of POMDPs where the problem data have been generated without any special assumptions about problem structure (Data sets 1-6).

2. Group 2 consists of "close-to-perfect" information POMDPs (Data sets 7-10).
3. Group 3 consists of CUMDP problems (Data sets 11-14).

4. Group 4 consists of perfect information POMDPs (Data sets 15-18).

1. DATA SET 1 (Group 1)

   - Number of states of the Core Process = 3

   - Number of Actions = 2

   - Number of states of the Observation Process = 2

**ALTERNATIVE 1:**

\[ P^1 = \begin{pmatrix} 0.1 & 0.1 & 0.8 \\ 0.2 & 0.5 & 0.3 \\ 0.7 & 0.1 & 0.2 \end{pmatrix} \]

\[ R^1 = \begin{pmatrix} 0.7 & 0.3 \\ 0.1 & 0.9 \\ 0.4 & 0.6 \end{pmatrix} \]

\[ \gamma^1 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \]

**ALTERNATIVE 2:**

\[ P^2 = \begin{pmatrix} 0.1 & 0.8 & 0.1 \\ 0.7 & 0.1 & 0.2 \\ 0.1 & 0.9 & 0.0 \end{pmatrix} \]
APPENDIX A. TEST DATA FOR FINITE HORIZON ALGORITHMS

\[ R^2 = \begin{pmatrix} 0.2 & 0.8 \\ 0.4 & 0.6 \\ 0.3 & 0.7 \end{pmatrix} \]

\[ \gamma^2 = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \]

\[ \gamma(T) = \begin{pmatrix} 0 \\ 3.5 \\ 0 \end{pmatrix} \]
2. DATA SET 2 (Group 1)

- Number of states of the Core Process = 3

- Number of Actions = 3

- Number of states of the Observation Process = 3

**ALTERNATIVE 1:**

\[
P^1 = \begin{pmatrix}
0.573 & 0.346 & 0.081 \\
0.416 & 0.441 & 0.143 \\
0.103 & 0.390 & 0.507
\end{pmatrix}
\]

\[
R^1 = \begin{pmatrix}
0.646 & 0.199 & 0.155 \\
0.106 & 0.643 & 0.251 \\
0.120 & 0.220 & 0.660
\end{pmatrix}
\]

\[
\gamma^1 = \begin{pmatrix}
2.500 \\
7.400 \\
1.300
\end{pmatrix}
\]

**ALTERNATIVE 2:**

\[
P^2 = \begin{pmatrix}
0.357 & 0.345 & 0.298 \\
0.370 & 0.460 & 0.170 \\
0.169 & 0.192 & 0.639
\end{pmatrix}
\]

\[
R^2 = \begin{pmatrix}
0.704 & 0.116 & 0.179 \\
0.0 & 0.840 & 0.160 \\
0.188 & 0.135 & 0.677
\end{pmatrix}
\]
APPENDIX A. TEST DATA FOR FINITE HORIZON ALGORITHMS

\[ \gamma^2 = \begin{pmatrix} 5.800 \\ 3.600 \\ 6.600 \end{pmatrix} \]

**ALTERNATIVE 3:**

\[ P^3 = \begin{pmatrix} 0.305 & 0.239 & 0.456 \\ 0.388 & 0.356 & 0.256 \\ 0.139 & 0.271 & 0.590 \end{pmatrix} \]

\[ R^3 = \begin{pmatrix} 0.702 & 0.211 & 0.087 \\ 0.304 & 0.581 & 0.115 \\ 0.145 & 0.238 & 0.617 \end{pmatrix} \]

\[ \gamma^3 = \begin{pmatrix} 7.200 \\ 7.000 \\ 4.900 \end{pmatrix} \]
3. DATA SET 3 (Group 1)

- Number of states of the Core Process = 3
- Number of Actions = 3
- Number of states of the Observation Process = 3

ALTERNATIVE 1:

\[
P^1 = \begin{pmatrix} 0.256 & 0.693 & 0.051 \\ 0.002 & 0.273 & 0.725 \\ 0.573 & 0.063 & 0.359 \end{pmatrix}
\]

\[
R^1 = \begin{pmatrix} 0.712 & 0.066 & 0.222 \\ 0.089 & 0.790 & 0.121 \\ 0.014 & 0.126 & 0.860 \end{pmatrix}
\]

\[
\gamma^1 = \begin{pmatrix} 7.100 \\ 9.300 \\ 5.300 \end{pmatrix}
\]

ALTERNATIVE 2:

\[
P^2 = \begin{pmatrix} 0.350 & 0.239 & 0.411 \\ 0.244 & 0.237 & 0.519 \\ 0.227 & 0.323 & 0.450 \end{pmatrix}
\]

\[
R^2 = \begin{pmatrix} 0.763 & 0.041 & 0.196 \\ 0.081 & 0.566 & 0.353 \\ 0.092 & 0.031 & 0.877 \end{pmatrix}
\]
APPENDIX A. TEST DATA FOR FINITE HORIZON ALGORITHMS

\[ \gamma^2 = \begin{pmatrix} 8.300 \\ 8.600 \\ 7.800 \end{pmatrix} \]

**ALTERNATIVE 3:**

\[ P^3 = \begin{pmatrix} 0.482 & 0.045 & 0.473 \\ 0.108 & 0.485 & 0.407 \\ 0.445 & 0.158 & 0.397 \end{pmatrix} \]

\[ R^3 = \begin{pmatrix} 0.504 & 0.061 & 0.435 \\ 0.080 & 0.705 & 0.215 \\ 0.194 & 0.015 & 0.791 \end{pmatrix} \]

\[ \gamma^3 = \begin{pmatrix} 0.900 \\ 3.900 \\ 8.800 \end{pmatrix} \]
4. DATA SET 4 (Group 1)

- Number of states of the Core Process = 4

- Number of Actions = 4

- Number of states of the Observation Process = 4

ALTERNATIVE 1:

\[ P^1 = \begin{pmatrix} 0.355 & 0.321 & 0.100 & 0.224 \\ 0.430 & 0.181 & 0.342 & 0.044 \\ 0.254 & 0.155 & 0.065 & 0.526 \\ 0.066 & 0.420 & 0.172 & 0.342 \end{pmatrix} \]

\[ R^1 = \begin{pmatrix} 0.519 & 0.192 & 0.154 & 0.135 \\ 0.161 & 0.551 & 0.093 & 0.195 \\ 0.112 & 0.158 & 0.662 & 0.068 \\ 0.157 & 0.126 & 0.055 & 0.662 \end{pmatrix} \]

\[ \gamma^1 = \begin{pmatrix} 0.2 \\ 8.2 \\ 6.8 \\ 8.1 \end{pmatrix} \]

ALTERNATIVE 2:

\[ P^2 = \begin{pmatrix} 0.094 & 0.313 & 0.262 & 0.333 \\ 0.241 & 0.173 & 0.233 & 0.353 \\ 0.351 & 0.259 & 0.273 & 0.117 \\ 0.064 & 0.174 & 0.454 & 0.308 \end{pmatrix} \]
\[ R^2 = \begin{pmatrix} 0.571 & 0.013 & 0.162 & 0.254 \\ 0.158 & 0.547 & 0.105 & 0.190 \\ 0.089 & 0.180 & 0.629 & 0.102 \\ 0.109 & 0.198 & 0.149 & 0.544 \end{pmatrix} \]

\[ \gamma^2 = \begin{pmatrix} 0.8 \\ 1.0 \\ 5.4 \\ 3.9 \end{pmatrix} \]

**ALTERNATIVE 3:**

\[ P^3 = \begin{pmatrix} 0.455 & 0.182 & 0.135 & 0.228 \\ 0.370 & 0.280 & 0.005 & 0.345 \\ 0.220 & 0.276 & 0.270 & 0.234 \\ 0.403 & 0.499 & 0.071 & 0.027 \end{pmatrix} \]

\[ R^3 = \begin{pmatrix} 0.650 & 0.121 & 0.126 & 0.103 \\ 0.175 & 0.538 & 0.134 & 0.153 \\ 0.064 & 0.086 & 0.676 & 0.174 \\ 0.204 & 0.232 & 0.010 & 0.554 \end{pmatrix} \]

\[ \gamma^3 = \begin{pmatrix} 2.2 \\ 6.7 \\ 2.6 \\ 1.7 \end{pmatrix} \]

**ALTERNATIVE 4:**

\[ P^4 = \begin{pmatrix} 0.255 & 0.255 & 0.230 & 0.260 \\ 0.294 & 0.108 & 0.417 & 0.181 \\ 0.025 & 0.143 & 0.352 & 0.480 \\ 0.267 & 0.298 & 0.194 & 0.241 \end{pmatrix} \]
APPENDIX A. TEST DATA FOR FINITE HORIZON ALGORITHMS

\[ R^4 = \begin{pmatrix}
0.672 & 0.103 & 0.134 & 0.091 \\
0.004 & 0.694 & 0.109 & 0.193 \\
0.159 & 0.166 & 0.621 & 0.054 \\
0.070 & 0.172 & 0.123 & 0.635
\end{pmatrix} \]

\[ \gamma^4 = \begin{pmatrix}
7.9 \\
1.5 \\
7.4 \\
1.7
\end{pmatrix} \]
5. DATA SET 5 (Group 1)

- Number of states of the Core Process = 4

- Number of Actions = 4

\[ \text{Number of states of the Observation Process} = 4 \]

**ALTERNATIVE 1:**

\[
P^1 = \begin{pmatrix}
0.252 & 0.210 & 0.240 & 0.298 \\
0.291 & 0.123 & 0.149 & 0.437 \\
0.171 & 0.075 & 0.475 & 0.279 \\
0.294 & 0.010 & 0.396 & 0.300
\end{pmatrix}
\]

\[
R^1 = \begin{pmatrix}
0.700 & 0.088 & 0.131 & 0.081 \\
0.075 & 0.699 & 0.149 & 0.077 \\
0.222 & 0.016 & 0.710 & 0.052 \\
0.088 & 0.252 & 0.149 & 0.511
\end{pmatrix}
\]

\[
\gamma^1 = \begin{pmatrix}
1.1 \\
9.2 \\
4.2 \\
9.9
\end{pmatrix}
\]

**ALTERNATIVE 2:**

\[
P^2 = \begin{pmatrix}
0.375 & 0.192 & 0.150 & 0.283 \\
0.462 & 0.121 & 0.077 & 0.340 \\
0.120 & 0.640 & 0.092 & 0.148 \\
0.222 & 0.283 & 0.129 & 0.366
\end{pmatrix}
\]
\[ R^2 = \begin{pmatrix} 0.644 & 0.156 & 0.163 & 0.037 \\ 0.040 & 0.750 & 0.088 & 0.122 \\ 0.050 & 0.204 & 0.655 & 0.091 \\ 0.020 & 0.038 & 0.154 & 0.788 \end{pmatrix} \]

\[ \gamma^2 = \begin{pmatrix} 0.8 \\ 8.1 \\ 9.5 \\ 7.1 \end{pmatrix} \]

**ALTERNATIVE 3:**

\[ P^3 = \begin{pmatrix} 0.255 & 0.140 & 0.309 & 0.296 \\ 0.180 & 0.395 & 0.032 & 0.393 \\ 0.286 & 0.229 & 0.273 & 0.212 \\ 0.157 & 0.012 & 0.517 & 0.314 \end{pmatrix} \]

\[ R^3 = \begin{pmatrix} 0.713 & 0.002 & 0.139 & 0.146 \\ 0.116 & 0.667 & 0.020 & 0.197 \\ 0.076 & 0.159 & 0.678 & 0.087 \\ 0.176 & 0.083 & 0.096 & 0.645 \end{pmatrix} \]

\[ \gamma^3 = \begin{pmatrix} 5.5 \\ 7.3 \\ 9.3 \\ 1.3 \end{pmatrix} \]

**ALTERNATIVE 4:**

\[ P^4 = \begin{pmatrix} 0.316 & 0.188 & 0.378 & 0.118 \\ 0.333 & 0.320 & 0.228 & 0.119 \\ 0.280 & 0.346 & 0.069 & 0.305 \\ 0.149 & 0.185 & 0.392 & 0.274 \end{pmatrix} \]
$R^4 = \begin{pmatrix}
0.728 & 0.080 & 0.072 & 0.120 \\
0.102 & 0.612 & 0.208 & 0.078 \\
0.198 & 0.123 & 0.540 & 0.139 \\
0.037 & 0.170 & 0.260 & 0.533 \\
\end{pmatrix}$

$\gamma^4 = \begin{pmatrix}
9.7 \\
8.0 \\
8.4 \\
4.0 \\
\end{pmatrix}$
6. DATA SET 6 (Group 1)

- Number of states of the Core Process = 5

- Number of Actions = 3

- Number of states of the Observation Process = 3

**ALTERNATIVE 1:**

\[
P^1 = \begin{pmatrix}
0.064 & 0.343 & 0.303 & 0.106 & 0.184 \\
0.340 & 0.087 & 0.214 & 0.069 & 0.290 \\
0.189 & 0.200 & 0.402 & 0.097 & 0.112 \\
0.322 & 0.003 & 0.053 & 0.322 & 0.300 \\
0.137 & 0.292 & 0.108 & 0.100 & 0.363
\end{pmatrix}
\]

\[
R^1 = \begin{pmatrix}
0.092 & 0.403 & 0.505 \\
0.310 & 0.099 & 0.591 \\
0.431 & 0.289 & 0.280 \\
0.287 & 0.304 & 0.400 \\
0.382 & 0.213 & 0.405
\end{pmatrix}
\]

\[
\gamma^1 = \begin{pmatrix}
10 \\
6.0 \\
4.9 \\
3.2 \\
7.4
\end{pmatrix}
\]
ALTERNATIVE 2:

\[ P^2 = \begin{pmatrix}
0.425 & 0.117 & 0.261 & 0.077 & 0.120 \\
0.108 & 0.298 & 0.092 & 0.320 & 0.182 \\
0.470 & 0.033 & 0.155 & 0.194 & 0.148 \\
0.173 & 0.184 & 0.284 & 0.215 & 0.144 \\
0.154 & 0.136 & 0.236 & 0.128 & 0.346
\end{pmatrix} \]

\[ R^2 = \begin{pmatrix}
0.081 & 0.495 & 0.424 \\
0.736 & 0.024 & 0.240 \\
0.521 & 0.467 & 0.012 \\
0.360 & 0.199 & 0.441 \\
0.040 & 0.447 & 0.513
\end{pmatrix} \]

\[ \gamma^2 = \begin{pmatrix}
4.5 \\
0.2 \\
2.5 \\
6.5 \\
5.5
\end{pmatrix} \]

ALTERNATIVE 3:

\[ P^3 = \begin{pmatrix}
0.005 & 0.226 & 0.292 & 0.264 & 0.213 \\
0.239 & 0.126 & 0.266 & 0.220 & 0.149 \\
0.228 & 0.016 & 0.259 & 0.040 & 0.457 \\
0.092 & 0.019 & 0.518 & 0.034 & 0.337 \\
0.324 & 0.272 & 0.024 & 0.047 & 0.333
\end{pmatrix} \]

\[ R^3 = \begin{pmatrix}
0.226 & 0.403 & 0.371 \\
0.416 & 0.144 & 0.440 \\
0.462 & 0.319 & 0.219 \\
0.260 & 0.103 & 0.637 \\
0.508 & 0.078 & 0.414
\end{pmatrix} \]
\[
\gamma^3 = \begin{pmatrix}
2.3 \\
7.9 \\
6.2 \\
6.5 \\
3.3
\end{pmatrix}
\]
7. DATA SET 7 (Group 2)

- Number of states of the Core Process = 2

- Number of Actions = 2

- Number of states of the Observation Process = 2

ALTERNATIVE 1:

\[ P^1 = \begin{pmatrix} 0.5025 & 0.4975 \\ 0.1371 & 0.8629 \end{pmatrix} \]

\[ R^1 = \begin{pmatrix} 0.0848 & 0.9152 \\ 0.9360 & 0.0640 \end{pmatrix} \]

\[ \gamma^1 = \begin{pmatrix} 10.494 \\ 3.741 \end{pmatrix} \]

ALTERNATIVE 2:

\[ P^2 = \begin{pmatrix} 0.7766 & 0.2234 \\ 0.4162 & 0.5838 \end{pmatrix} \]

\[ R^2 = \begin{pmatrix} 0.0904 & 0.9096 \\ 0.9249 & 0.0751 \end{pmatrix} \]

\[ \gamma^2 = \begin{pmatrix} 1.305 \\ 7.701 \end{pmatrix} \]

\[ \gamma(T) = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \]
APPENDIX A. TEST DATA FOR FINITE HORIZON ALGORITHMS

8. DATA SET 8 (Group 2)

- Number of states of the Core Process = 2

- Number of Actions = 2

- Number of states of the Observation Process = 2

**ALTERNATIVE 1:**

\[
P^1 = \begin{pmatrix} 0.1218 & 0.8782 \\ 0.7614 & 0.2386 \end{pmatrix}
\]

\[
R^1 = \begin{pmatrix} 0.0132 & 0.9868 \\ 0.9883 & 0.0117 \end{pmatrix}
\]

\[
\gamma^1 = \begin{pmatrix} 9.832 \\ 2.624 \end{pmatrix}
\]

**ALTERNATIVE 2:**

\[
P^2 = \begin{pmatrix} 0.4010 & 0.5990 \\ 0.0406 & 0.9594 \end{pmatrix}
\]

\[
R^2 = \begin{pmatrix} 0.0289 & 0.9711 \\ 0.9569 & 0.0431 \end{pmatrix}
\]

\[
\gamma^2 = \begin{pmatrix} 6.584 \\ 5.772 \end{pmatrix}
\]

\[
\gamma(T) = \begin{pmatrix} 0 \\ 0 \end{pmatrix}
\]
9. DATA SET 9 (Group 2)

- Number of states of the Core Process = 3

- Number of Actions = 3

- Number of states of the Observation Process = 3

ALTERNATIVE 1:

\[
P^1 = \begin{pmatrix} 0.483 & 0.268 & 0.249 \\ 0.000 & 0.000 & 1.000 \\ 0.000 & 0.698 & 0.302 \end{pmatrix}
\]

\[
R^1 = \begin{pmatrix} 0.003 & 0.024 & 0.973 \\ 0.052 & 0.928 & 0.020 \\ 0.984 & 0.012 & 0.004 \end{pmatrix}
\]

\[
\gamma^1 = \begin{pmatrix} 5.100 \\ 5.200 \\ 9.500 \end{pmatrix}
\]

ALTERNATIVE 2:

\[
P^2 = \begin{pmatrix} 0.665 & 0.335 & 0.000 \\ 0.407 & 0.223 & 0.370 \\ 0.695 & 0.000 & 0.305 \end{pmatrix}
\]

\[
R^2 = \begin{pmatrix} 0.023 & 0.011 & 0.966 \\ 0.0 & 0.940 & 0.060 \\ 0.988 & 0.002 & 0.010 \end{pmatrix}
\]
\begin{align*}
\gamma^2 &= \begin{pmatrix}
5.800 \\
4.600 \\
1.400 
\end{pmatrix} \\
\text{ALTERNATIVE 3:} \\
P^3 &= \begin{pmatrix}
0.363 & 0.361 & 0.276 \\
0.430 & 0.000 & 0.570 \\
1.000 & 0.000 & 0.000
\end{pmatrix} \\
R^3 &= \begin{pmatrix}
0.006 & 0.007 & 0.987 \\
0.000 & 0.981 & 0.019 \\
0.945 & 0.042 & 0.013
\end{pmatrix} \\
\gamma^3 &= \begin{pmatrix}
7.900 \\
6.800 \\
7.300 
\end{pmatrix} \\
\gamma(T) &= \begin{pmatrix}
0 \\
0 \\
0
\end{pmatrix}
\end{align*}
10. DATA SET 10 (Group 2)

- Number of states of the Core Process = 3

- Number of Actions = 6

- Number of states of the Observation Process = 3

ALTERNATIVE 1:

\[
P^1 = \begin{pmatrix} 0.000 & 0.388 & 0.612 \\ 0.580 & 0.420 & 0.000 \\ 0.379 & 0.180 & 0.441 \end{pmatrix}
\]

\[
R^1 = \begin{pmatrix} 0.030 & 0.030 & 0.940 \\ 0.002 & 0.962 & 0.036 \\ 0.984 & 0.006 & 0.010 \end{pmatrix}
\]

\[
\gamma^1 = \begin{pmatrix} 1.900 \\ 6.200 \\ 9.600 \end{pmatrix}
\]

ALTERNATIVE 2:

\[
P^2 = \begin{pmatrix} 0.000 & 1.000 & 0.000 \\ 0.479 & 0.000 & 0.521 \\ 0.529 & 0.000 & 0.471 \end{pmatrix}
\]

\[
R^2 = \begin{pmatrix} 0.011 & 0.001 & 0.988 \\ 0.020 & 0.956 & 0.024 \\ 0.991 & 0.002 & 0.007 \end{pmatrix}
\]
\[ \gamma^2 = \begin{pmatrix} 2.700 \\ 8.700 \\ 5.600 \end{pmatrix} \]

**ALTERNATIVE 3:**

\[ P^3 = \begin{pmatrix} 0.324 & 0.291 & 0.385 \\ 0.506 & 0.000 & 0.494 \\ 0.000 & 1.000 & 0.000 \end{pmatrix} \]

\[ R^3 = \begin{pmatrix} 0.006 & 0.081 & 0.913 \\ 0.003 & 0.972 & 0.025 \\ 0.934 & 0.042 & 0.024 \end{pmatrix} \]

\[ \gamma^3 = \begin{pmatrix} 1.800 \\ 6.200 \\ 8.100 \end{pmatrix} \]

**ALTERNATIVE 4:**

\[ P^4 = \begin{pmatrix} 0.716 & 0.002 & 0.282 \\ 0.171 & 0.612 & 0.216 \\ 0.124 & 0.214 & 0.661 \end{pmatrix} \]

\[ R^4 = \begin{pmatrix} 0.006 & 0.021 & 0.973 \\ 0.000 & 0.991 & 0.009 \\ 0.966 & 0.002 & 0.032 \end{pmatrix} \]

\[ \gamma^4 = \begin{pmatrix} 7.200 \\ 4.100 \\ 0.400 \end{pmatrix} \]

**ALTERNATIVE 5:**

\[ P^5 = \begin{pmatrix} 0.822 & 0.058 & 0.120 \\ 0.175 & 0.726 & 0.099 \\ 0.215 & 0.065 & 0.721 \end{pmatrix} \]
\[ R^5 = \begin{pmatrix} 0.003 & 0.020 & 0.977 \\ 0.044 & 0.952 & 0.004 \\ 0.982 & 0.006 & 0.012 \end{pmatrix} \]

\[ \gamma^5 = \begin{pmatrix} 2.000 \\ 2.400 \\ 9.800 \end{pmatrix} \]

**ALTERNATIVE 6:**

\[ P^6 = \begin{pmatrix} 0.666 & 0.033 & 0.301 \\ 0.114 & 0.652 & 0.233 \\ 0.217 & 0.116 & 0.667 \end{pmatrix} \]

\[ R^6 = \begin{pmatrix} 0.006 & 0.030 & 0.964 \\ 0.020 & 0.976 & 0.004 \\ 0.922 & 0.014 & 0.064 \end{pmatrix} \]

\[ \gamma^6 = \begin{pmatrix} 6.200 \\ 9.600 \\ 4.300 \end{pmatrix} \]

\[ \gamma(T) = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \]
11. DATA SET 11 (Group 3)

- Number of states of the Core Process = 2
- Number of Actions = 2
- Number of states of the Observation Process = 1

ALTERNATIVE 1:

\[
P^1 = \begin{pmatrix} 0.3401 & 0.6599 \\ 0.9797 & 0.0203 \end{pmatrix}
\]

\[
\gamma^1 = \begin{pmatrix} 5.365 \\ 11.761 \end{pmatrix}
\]

ALTERNATIVE 2:

\[
P^2 = \begin{pmatrix} 0.3807 & 0.6193 \\ 0.2589 & 0.7411 \end{pmatrix}
\]

\[
\gamma^2 = \begin{pmatrix} 4.553 \\ 10.949 \end{pmatrix}
\]

\[
\gamma(T) = \begin{pmatrix} 0 \\ 0 \end{pmatrix}
\]
12. DATA SET 12 (Group 3)

- Number of states of the Core Process = 5

- Number of Actions = 2

- Number of states of the Observation Process = 1

ALTERNATIVE 1:

\[
P^1 = \begin{pmatrix}
0.0015 & 0.0007 & 0.2122 & 0.4354 & 0.3503 \\
0.0137 & 0.9492 & 0.0014 & 0.0139 & 0.0218 \\
0.0246 & 0.0112 & 0.3922 & 0.5482 & 0.0238 \\
0.1472 & 0.0866 & 0.0103 & 0.3576 & 0.3983 \\
0.1027 & 0.0915 & 0.0167 & 0.7462 & 0.0429
\end{pmatrix}
\]

\[
\gamma^1 = \begin{pmatrix}
7.802 \\
12.574 \\
11.761 \\
4.553 \\
10.949
\end{pmatrix}
\]

ALTERNATIVE 2:

\[
P^2 = \begin{pmatrix}
0.0010 & 0.2160 & 0.4354 & 0.3452 & 0.0023 \\
0.0017 & 0.0155 & 0.0240 & 0.0147 & 0.9442 \\
0.5431 & 0.0255 & 0.0257 & 0.0114 & 0.3942 \\
0.3963 & 0.1421 & 0.0905 & 0.0102 & 0.3608 \\
0.1055 & 0.0926 & 0.0158 & 0.7411 & 0.0451
\end{pmatrix}
\]
\[ \gamma^2 = \begin{pmatrix} 17.345 \\ 3.741 \\ 10.137 \\ 16.533 \\ 2.929 \end{pmatrix} \]

\[ \gamma(T') = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \]
13. DATA SET 13 (Group 3)

- Number of states of the Core Process $= 8$

- Number of Actions $= 2$

- Number of states of the Observation Process $= 1$

**ALTERNATIVE 1:**

$$P^1 = \begin{pmatrix}
0.0942 & 0.1767 & 0.0013 & 0.0838 & 0.1662 & 0.2487 & 0.0733 & 0.1558 \\
0.0000 & 0.2209 & 0.0583 & 0.1347 & 0.2112 & 0.0485 & 0.1250 & 0.2015 \\
0.0000 & 0.0000 & 0.0586 & 0.1740 & 0.2894 & 0.0440 & 0.1593 & 0.2747 \\
0.0000 & 0.0000 & 0.0000 & 0.0506 & 0.2500 & 0.4494 & 0.0253 & 0.2247 \\
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.2577 & 0.3788 & 0.1212 & 0.2423 \\
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.5221 & 0.1519 & 0.3260 \\
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.7939 & 0.2061 \\
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 1.0000
\end{pmatrix}$$

$$\gamma^1 = \begin{pmatrix}
1.0000 \\
5.772 \\
6.584 \\
7.396 \\
8.208 \\
12.980 \\
13.792 \\
14.604
\end{pmatrix}$$
ALTERNATIVE 2:

\[ P^2 = \begin{pmatrix}
0.2207 & 0.0497 & 0.1301 & 0.2105 & 0.0395 & 0.1199 & 0.2003 & 0.0293 \\
0.2207 & 0.0497 & 0.1301 & 0.2105 & 0.0395 & 0.1199 & 0.2003 & 0.0293 \\
0.2207 & 0.0497 & 0.1301 & 0.2105 & 0.0395 & 0.1199 & 0.2003 & 0.0293 \\
0.2207 & 0.0497 & 0.1301 & 0.2105 & 0.0395 & 0.1199 & 0.2003 & 0.0293 \\
0.2207 & 0.0497 & 0.1301 & 0.2105 & 0.0395 & 0.1199 & 0.2003 & 0.0293 \\
0.2207 & 0.0497 & 0.1301 & 0.2105 & 0.0395 & 0.1199 & 0.2003 & 0.0293 \\
0.2207 & 0.0497 & 0.1301 & 0.2105 & 0.0395 & 0.1199 & 0.2003 & 0.0293 \\
0.2207 & 0.0497 & 0.1301 & 0.2105 & 0.0395 & 0.1199 & 0.2003 & 0.0293 \\
\end{pmatrix} \]

\[ \gamma^2 = \begin{pmatrix}
8.596 \\
8.596 \\
8.596 \\
8.596 \\
8.596 \\
8.596 \\
8.596 \\
8.596 \\
\end{pmatrix} \]

\[ \gamma(T) = \begin{pmatrix}
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
\end{pmatrix} \]
14. DATA SET 14 (Group 3)

- Number of states of the Core Process = 2
- Number of Actions = 2
- Number of states of the Observation Process = 1

**ALTERNATIVE 1:**

\[ P^1 = \begin{pmatrix} 0.3417 & 0.6583 \\ 0.0000 & 1.0000 \end{pmatrix} \]

\[ \gamma^1 = \begin{pmatrix} 7.497 \\ 13.893 \end{pmatrix} \]

**ALTERNATIVE 2:**

\[ P^2 = \begin{pmatrix} 0.3279 & 0.6721 \\ 0.3279 & 0.6721 \end{pmatrix} \]

\[ \gamma^2 = \begin{pmatrix} 13.244 \\ 13.244 \end{pmatrix} \]

\[ \gamma(T) = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \]
15. DATA SET 15 (Group 4)

- Number of states of the Core Process = 3

- Number of Actions = 3

- Number of states of the Observation Process = 3

**ALTERNATIVE 1:**

\[
P^1 = \begin{pmatrix} 0.483 & 0.268 & 0.249 \\ 0.171 & 0.612 & 0.216 \\ 0.000 & 0.698 & 0.302 \end{pmatrix}
\]

\[
R^1 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}
\]

\[
\gamma^1 = \begin{pmatrix} 5.10 \\ 5.20 \\ 9.50 \end{pmatrix}
\]

**ALTERNATIVE 2:**

\[
P^2 = \begin{pmatrix} 0.665 & 0.335 & 0.000 \\ 0.408 & 0.223 & 0.369 \\ 0.695 & 0.000 & 0.305 \end{pmatrix}
\]

\[
R^2 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}
\]
ALTRENATIVE 3:

\[
\gamma^2 = \begin{pmatrix} 5.80 \\ 4.60 \\ 1.40 \end{pmatrix}
\]

\[
P^3 = \begin{pmatrix} 0.363 & 0.361 & 0.276 \\ 0.430 & 0.000 & 0.570 \\ 1 & 0.000 & 0.000 \end{pmatrix}
\]

\[
R^3 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}
\]

\[
\gamma^3 = \begin{pmatrix} 7.90 \\ 6.80 \\ 7.30 \end{pmatrix}
\]

\[
\gamma(T) = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}
\]
16. DATA SET 16 (Group 4)

- Number of states of the Core Process = 3

- Number of Actions = 3

- Number of states of the Observation Process = 3

**ALTERNATIVE 1:**

\[
P^1 = \begin{pmatrix}
0.716 & 0.002 & 0.282 \\
0.171 & 0.612 & 0.216 \\
0.124 & 0.214 & 0.661 \\
\end{pmatrix}
\]

\[
R^1 = \begin{pmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1 \\
\end{pmatrix}
\]

\[
\gamma^1 = \begin{pmatrix}
7.20 \\
4.10 \\
0.40 \\
\end{pmatrix}
\]

**ALTERNATIVE 2:**

\[
P^2 = \begin{pmatrix}
0.822 & 0.058 & 0.120 \\
0.175 & 0.726 & 0.099 \\
0.215 & 0.065 & 0.721 \\
\end{pmatrix}
\]

\[
R^2 = \begin{pmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1 \\
\end{pmatrix}
\]
\[ \gamma^2 = \begin{pmatrix} 2.00 \\ 2.40 \\ 9.80 \end{pmatrix} \]

**ALTERNATIVE 3:**

\[ P^3 = \begin{pmatrix} 0.666 & 0.033 & 0.301 \\ 0.114 & 0.652 & 0.233 \\ 0.217 & 0.116 & 0.667 \end{pmatrix} \]

\[ R^3 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \]

\[ \gamma^3 = \begin{pmatrix} 6.20 \\ 9.60 \\ 4.30 \end{pmatrix} \]

\[ \gamma(T) = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \]
17. DATA SET 17 (Group 4)

- Number of states of the Core Process = 4

- Number of Actions = 4

- Number of states of the Observation Process = 4

**ALTERNATIVE 1:**

\[ P^1 = \begin{pmatrix} 0.379 & 0.271 & 0.000 & 0.350 \\ 0.000 & 0.341 & 0.407 & 0.252 \\ 0.000 & 0.434 & 0.000 & 0.566 \\ 0.000 & 0.414 & 0.000 & 0.586 \end{pmatrix} \]

\[ R^1 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \]

\[ \gamma^1 = \begin{pmatrix} 0.20 \\ 5.40 \\ 2.50 \\ 8.00 \end{pmatrix} \]

**ALTERNATIVE 2:**

\[ P^2 = \begin{pmatrix} 0.000 & 0.631 & 0.000 & 0.369 \\ 0.632 & 0.000 & 0.000 & 0.368 \\ 0.000 & 0.000 & 0.523 & 0.477 \\ 0.000 & 0.521 & 0.000 & 0.479 \end{pmatrix} \]
\[ R^2 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \]

\[ \gamma^2 = \begin{pmatrix} 5.80 \\ 8.40 \\ 7.50 \\ 1.70 \end{pmatrix} \]

**ALTERNATIVE 3:**

\[ P^3 = \begin{pmatrix} 0.319 & 0.339 & 0.342 & 0.000 \\ 0.190 & 0.332 & 0.188 & 0.290 \\ 0.517 & 0.000 & 0.000 & 0.483 \\ 0.000 & 0.378 & 0.000 & 0.622 \end{pmatrix} \]

\[ R^3 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \]

\[ \gamma^3 = \begin{pmatrix} 9.30 \\ 5.60 \\ 9.70 \\ 8.80 \end{pmatrix} \]

**ALTERNATIVE 4:**

\[ P^4 = \begin{pmatrix} 0.000 & 1.000 & 0.000 & 0.000 \\ 0.416 & 0.000 & 0.584 & 0.000 \\ 0.000 & 0.199 & 0.292 & 0.509 \\ 0.402 & 0.000 & 0.272 & 0.326 \end{pmatrix} \]
\[ R^4 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \]

\[ \gamma^4 = \begin{pmatrix} 4.20 \\ 9.40 \\ 4.40 \\ 2.60 \end{pmatrix} \]

\[ \gamma(T) = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \]
18. DATA SET 18 (Group 4)

- Number of states of the Core Process = 5

- Number of Actions = 5

- Number of states of the Observation Process = 5

ALTERNATIVE 1:

\[ P^1 = \begin{pmatrix}
0.000 & 0.234 & 0.000 & 0.292 & 0.474 \\
0.266 & 0.397 & 0.000 & 0.000 & 0.337 \\
0.000 & 0.000 & 0.344 & 0.339 & 0.318 \\
0.000 & 0.362 & 0.638 & 0.000 & 0.000 \\
0.274 & 0.248 & 0.000 & 0.303 & 0.175
\end{pmatrix} \]

\[ R^1 = \begin{pmatrix}
1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 1
\end{pmatrix} \]

\[ \gamma^1 = \begin{pmatrix}
2.40 \\
5.80 \\
4.20 \\
2.70 \\
2.60
\end{pmatrix} \]
APPENDIX A. TEST DATA FOR FINITE HORIZON ALGORITHMS

ALTERNATIVE 2:

\[
P^2 = \begin{pmatrix}
0.000 & 0.233 & 0.315 & 0.000 & 0.452 \\
0.347 & 0.360 & 0.000 & 0.000 & 0.293 \\
0.192 & 0.192 & 0.246 & 0.105 & 0.265 \\
0.000 & 0.000 & 0.279 & 0.721 & 0.000 \\
0.314 & 0.000 & 0.304 & 0.226 & 0.157
\end{pmatrix}
\]

\[
R^2 = \begin{pmatrix}
1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 1
\end{pmatrix}
\]

\[
\gamma^2 = \begin{pmatrix}
3.70 \\
5.10 \\
7.60 \\
5.60 \\
9.00
\end{pmatrix}
\]

ALTERNATIVE 3:

\[
P^3 = \begin{pmatrix}
0.275 & 0.257 & 0.284 & 0.184 & 0.000 \\
0.000 & 0.381 & 0.000 & 0.358 & 0.261 \\
0.294 & 0.000 & 0.000 & 0.335 & 0.371 \\
0.343 & 0.264 & 0.000 & 0.265 & 0.128 \\
0.595 & 0.000 & 0.405 & 0.000 & 0.000
\end{pmatrix}
\]

\[
R^3 = \begin{pmatrix}
1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 1
\end{pmatrix}
\]
\[ \gamma^3 = \begin{pmatrix} 5.50 \\ 1.60 \\ 1.00 \\ 3.70 \\ 1.70 \end{pmatrix} \]

**ALTERNATIVE 4:**

\[ P^4 = \begin{pmatrix} 0.167 & 0.301 & 0.250 & 0.282 & 0.000 \\ 0.343 & 0.000 & 0.207 & 0.450 & 0.000 \\ 0.288 & 0.000 & 0.170 & 0.228 & 0.314 \\ 0.416 & 0.251 & 0.000 & 0.333 & 0.000 \\ 0.264 & 0.433 & 0.000 & 0.304 & 0.000 \end{pmatrix} \]

\[ R^4 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \]

\[ \gamma^4 = \begin{pmatrix} 3.90 \\ 5.90 \\ 3.70 \\ 3.90 \\ 4.30 \end{pmatrix} \]

**ALTERNATIVE 5:**

\[ P^5 = \begin{pmatrix} 0.177 & 0.000 & 0.390 & 0.289 & 0.144 \\ 0.274 & 0.286 & 0.141 & 0.183 & 0.116 \\ 0.284 & 0.145 & 0.261 & 0.310 & 0.000 \\ 0.386 & 0.403 & 0.000 & 0.211 & 0.000 \\ 0.239 & 0.000 & 0.253 & 0.000 & 0.508 \end{pmatrix} \]
\[ R^5 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \]

\[ \gamma^5 = \begin{pmatrix} 3.20 \\ 1.80 \\ 4.30 \\ 1.50 \\ 9.10 \end{pmatrix} \]

\[ \gamma(T) = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \]
Appendix B

CALCULATION OF THE EXPECTED REWARD

The expected immediate reward of making one transition and producing some output \( z \), given the three dimensional random variable \( s(t) = i = (i_1, i_2, i_3) \) and \( a_t \), can be represented as follows:

Let

\( a_R \) be the alternative - Stop the mill and replace all the rolls without inspecting them.

\( a_C \) be the alternative - Continue the finishing mill operation with no examination of the rolls.

\( a_{1234} \) be the alternative - Stop the mill, inspect the rolls, and replace all that are in states 2, 3 or 4.
\( a_{134} \) be the alternative - Stop the mill, inspect the rolls, and replace all that are in states 3 or 4.

\( a_{14} \) be the alternative - Stop the mill, inspect the rolls, and replace all that are in state 4.

\( a_t \) be the alternative selected at time \( t \).

\( C^o \) be the operating cost per decision period.

\( C^R \) be the replacement cost (\$/replacement).

\( T^R \) be the replacement time.

\( C^{I-R} \) be inspection cost of the rolls (\$/inspection).

\( C^z \) be the revenue received from a coil with quality \( z \).

\( LD \) be a vector of average values of diameter reduction as a function of the surface condition of the rolls.

\( LD_j \) be the average loss in diameter, when rolls are in state \( j \) \((j = 1, 2, 3, 4)\).

\( GCR \) be the grind charge rate.
$SRR$ be the stock removal rate.

$RP$ be the roll's price.

$MLD$ the maximum loss in diameter per roll.

$LDP_1$, $LDP_2$, $LDP_3$ be the vectors of average values of loss diameter per period per stand (1, 2 and 3) respectively.

\[
LDP_i = \begin{pmatrix}
LDP_{i_1} \\
LDP_{i_2} \\
LDP_{i_3} \\
LDP_{i_4}
\end{pmatrix}
\]

$LDP_{i_j}$ be the average loss in diameter per period, when rolls at stand $i$ are in state $j$ ($j = 1, 2, 3, 4$).

$C_1^{I-P}$ be the inspection cost of the sample taken from the coil produced from every $20^{th}$ slab.

$C_2^{I-P}$ be the inspection cost incurred when the inspection of the sample reveals that the surface quality of the strip is lower than the surface quality required by the customer. In this case, a subsequent inspection of the previously produced coils is arranged.

$(ANR|x)$ be the average number of reapplications as a function of the surface quality observed from the end sample taken at 20-coil intervals.
Let

\[ U' = \begin{cases} 
1 & \text{Under } a_t \text{ there is an inspection of the rolls} \\
0 & \text{otherwise}
\end{cases} \]

\[ V_1 = \begin{cases} 
1 & \text{Under } a_t \text{ stand 1 is replaced} \\
0 & \text{otherwise}
\end{cases} \]

\[ V_2 = \begin{cases} 
1 & \text{Under } a_t \text{ stand 2 is replaced} \\
0 & \text{otherwise}
\end{cases} \]

\[ V_3 = \begin{cases} 
1 & \text{Under } a_t \text{ stand 3 is replaced} \\
0 & \text{otherwise}
\end{cases} \]

\[ W = \begin{cases} 
1 & \text{Under } a_t, V_1 \text{ or } V_2 \text{ or } V_3 \text{ are equal to 1} \\
0 & \text{otherwise}
\end{cases} \]
\[
\gamma_{i}^{a_{i}} = \sum_{j} \sum_{z \leq 3W} \{ -C^{o} - (U \times C^{l-R}) + 20C^{z} - C^{l-P} \\
- \frac{(2 \times RP)(LDP_{1i} + LDP_{2i2} + LDP_{3i3})}{(MLD)} - W \times C^{R} \\
- (GCR)(SRR)(2 \times V1 \times LD_{i1} + 2 \times V2 \times LD_{i2} + 2 \times V3 \times LD_{i3}) \} P_{ij}^{a_{i}r_{j}^{a_{j}}} \\
+ \sum_{j} \sum_{z > 3W} \{ -C^{o} - (U \times C^{l-R}) + C^{z}(ANR/z) + C^{3W}(20 - ANR/z) \\
-C^{l-P} - C^{l-P}(NCi) - \frac{(2 \times RP)(LDP_{1i1} + LDP_{2i2} + LDP_{3i3})}{(MLD)} \\
W \times C^{R} - (GCR)(SRR)(2 \times V1 \times LD_{i1} + 2 \times V2 \times LD_{i2} + 2 \times V3 \times LD_{i3}) \} \}
\]

\[
P_{ij}^{a_{i}r_{j}^{a_{j}}}
\]

(B.1)
Appendix C

OBSERVED SEQUENCES OF ROLLS’ DETERIORATION

The following sets of data represent the observed sequences of states of the deterioration process for rolls 1, 2, and 3.
1. OBSERVED SEQUENCES ROLL 1

1 1 2 2 3 1 1 1
1 1 1 2 3 3 4 4
1 1 2 3 3 3 1 1
1 1 2 3 1 1 4 4
1 1 2 2 2 3 3 4
1 1 2 3 4 1 4 4
1 1 2 2 4 4 1 1
1 1 2 3 3 4 4 4
1 1 1 2 3 3 1 1
1 1 1 2 1 4 4 1
1 1 1 2 3 4 1 1
1 1 1 2 2 3 4 4
1 1 2 3 4 4 4 4
1 1 2 3 4 4 4 4
1 1 2 3 3 3 3 4
1 1 2 4 4 4 4 4
1 1 2 3 3 3 3 4
1 1 2 2 4 4 4 4
1 1 2 2 3 3 4 4
1 1 2 2 3 3 3 4
1 1 1 2 3 3 3 3
1 1 2 3 4 4 4 4
1 1 2 2 3 4 4 4
1 1 2 3 3 4 4 4
1 1 3 3 3 4 4 4
1 1 2 3 3 4 4 4
1 1 2 3 3 4 4 4
1 1 3 3 4 4 4 4
3. OBSERVED SEQUENCES ROLL 3

```
1 1 1 1 2 2 2 3
1 1 1 1 2 2 3 3
1 1 1 1 2 2 3 4
1 1 1 1 2 3 3 1
1 1 1 1 2 2 3 3
1 1 1 1 2 2 2 3
1 1 1 1 2 3 3 1
1 1 1 1 2 2 3 4
1 1 1 2 2 3 3 3
1 1 1 2 3 3 3 4
1 1 1 2 3 3 3 3
1 1 1 2 3 3 3 1
1 1 1 2 2 3 3 4
1 1 1 2 2 3 3 4
1 1 1 2 2 3 3 4
1 1 1 2 2 3 3 4
1 1 1 2 3 3 3 4
1 1 1 2 2 3 3 3
1 1 1 2 2 2 3 3
1 1 1 2 2 2 3 3
1 1 1 2 2 2 3 3
1 1 1 2 2 2 3 3
1 1 1 2 2 2 3 3
1 1 1 2 2 2 3 3
1 1 1 2 2 2 3 3
1 1 1 2 2 2 3 3
1 1 1 2 2 2 3 3
1 1 1 2 2 2 3 3
1 1 1 2 2 2 3 3
1 1 1 2 2 2 3 3
1 1 1 2 2 2 3 3
1 1 1 2 2 2 3 3
1 1 1 2 2 2 3 3
1 1 1 2 2 2 3 3
1 1 1 2 2 2 3 3
1 1 1 2 2 2 3 3
1 1 1 2 2 2 3 3
1 1 1 2 2 2 3 3
1 1 1 2 2 2 3 3
1 1 1 2 2 2 3 3
1 1 1 2 2 2 3 3
1 1 1 2 2 2 3 3
1 1 1 2 2 2 3 3
1 1 1 2 2 2 3 3
1 1 1 2 2 2 3 3
1 1 1 2 2 2 3 3
1 1 1 2 2 2 3 3
1 1 1 2 2 2 3 3
1 1 1 2 2 2 3 3
1 1 1 2 2 2 3 3
1 1 1 2 2 2 3 3
1 1 1 2 2 2 3 3
1 1 1 2 2 2 3 3
1 1 1 2 2 2 3 3
1 1 1 2 2 2 3 3
1 1 1 2 2 2 3 3
1 1 1 2 2 2 3 3
1 1 1 2 2 2 3 3
1 1 1 2 2 2 3 3
1 1 1 2 2 2 3 3
1 1 1 2 2 2 3 3
1 1 1 2 2 2 3 3
1 1 1 2 2 2 3 3
1 1 1 2 2 2 3 3
1 1 1 2 2 2 3 3
1 1 1 2 2 2 3 3
1 1 1 2 2 2 3 3
1 1 1 2 2 2 3 3
1 1 1 2 2 2 3 3
1 1 1 2 2 2 3 3
1 1 1 2 2 2 3 3
1 1 1 2 2 2 3 3
1 1 1 2 2 2 3 3
1 1 1 2 2 2 3 3
1 1 1 2 2 2 3 3
1 1 1 2 2 2 3 3
1 1 1 2 2 2 3 3
1 1 1 2 2 2 3 3
1 1 1 2 2 2 3 3
1 1 1 2 2 2 3 3
1 1 1 2 2 2 3 3
1 1 1 2 2 2 3 3
1 1 1 2 2 2 3 3
```
IMAGE EVALUATION
TEST TARGET (QA-3)

APPLIED IMAGE, Inc
1853 East Main Street
Rochester, NY 14609 USA
Phone: 716/482-0300
Fax: 716/288-5989

© 1993 Applied Image, Inc., All Rights Reserved