AD-A034 249 STANFORD UNIV CALIF DEPT OF ENGINEERING-ECONOMIC SYSTEMS F/G 12/1
MARKOV DECISION PROCESSES WITH POLICY CONSTRAINTS.(U)
APR 76 J NAFEH NSF-GK-36491
UNCLASSIFIED EES-DA-76-3 NL

| OF 2
AD
A034249

MARKOV DECISION PROCESSES WITH POLICY CONSTRAINTS

STANFORD UNIVERSITY

John Nafeh

ADVANCED DECISION TECHNOLOGY PROGRAM

CYBERNETICS TECHNOLOGY OFFICE
DEFENSE ADVANCED RESEARCH PROJECTS AGENCY
Office of Naval Research • Engineering Psychology Programs

The objective of the Advanced Decision Technology Program is to develop and transfer to users in the Department of Defense advanced management technologies for decision making. These technologies are based upon research in the areas of decision analysis, the behavioral sciences and interactive computer graphics. The program is sponsored by the Cybernetics Technology Office of the Defense Advanced Research Projects Agency and technical progress is monitored by the Office of Naval Research – Engineering Psychology Programs. Participants in the program are:

Decisions and Designs, Incorporated
The Oregon Research Institute
Perceptronics, Incorporated
Stanford University
The University of Southern California

Inquiries and comments with regard to this report should be addressed to:

Dr. Martin A. Tolcott
Director, Engineering Psychology Programs
Office of Naval Research
800 North Quincy Street
Arlington, Virginia 22217

or

LT COL Roy M. Gulick, USMC
Cybernetics Technology Office
Defense Advanced Research Projects Agency
1400 Wilson Boulevard
Arlington, Virginia 22209

RESEARCH REPORT No. EES DA-76-3

MARKOV DECISION PROCESSES WITH POLICY CONSTRAINTS

by

John Nafeh

DECISION ANALYSIS PROGRAM

Professor Ronald A. Howard
Principal Investigator

Sponsored by

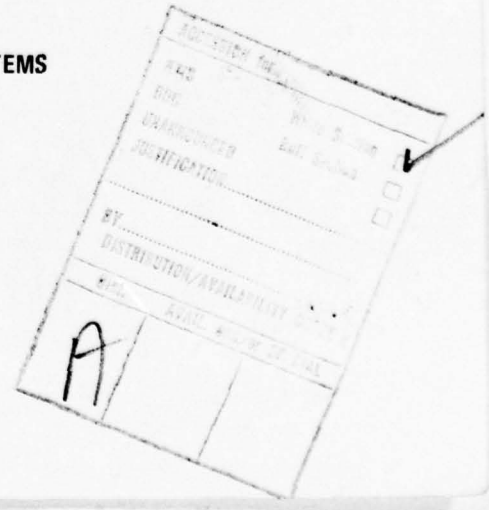
Defense Advanced Research Projects Agency
Under Subcontract from
Decisions and Designs, Incorporated

and

National Science Foundation, NSF Grant GK-36491

April, 1976

DEPARTMENT OF ENGINEERING-ECONOMIC SYSTEMS
Stanford University
Stanford, California 94305



SUMMARY

This work considers Markov decision processes with policy constraints. The selection of an optimal stationary policy for such processes, in the absence of policy constraints, is a problem which has received a great deal of attention and has been satisfactorily solved. Relatively little attention has been given to the case when policy constraints are present. Optimal policy sensitivity analysis is also a subject in which little has been achieved. We develop, in this paper, a computationally efficient iterative algorithm for selecting the optimal policy for completely ergodic, infinite-time horizon Markov decision processes with policy constraints for both the risk-indifferent and risk-sensitive cases. The sensitivity of optimal policies vis-a-vis the constraints is also analyzed, and the algorithm is used to quantify the analysis.

An important limitation of all previous analyses of Markov decision processes is the implicit assumption that selecting an alternative in any one state has no effect on alternative selection in any other state. If that assumption does not hold, we have "policy constraints". Some policies become "infeasible", i.e. unallowable. One method of dealing with such a situation was proposed for risk-indifferent Markov decision processes [10]. The policies can be ordered and, after determining the optimal policy, we can go backwards in the ordering, checking each policy for "feasibility". This method, however, becomes computationally

burdensome after the second-best policy. Moreover, no method of ordering has been devised for risk-sensitive Markov decision processes. The present work shows how to treat efficiently "policy constrained" problems for both risk-indifferent and risk-sensitive Markov decision processes by proceeding from one feasible policy to a better one until the optimal feasible policy is obtained. Our point of departure is reformulating the Markov decision process in the absence of policy constraints as a constrained maximization problem. The Lagrange multiplier rule is then applied to decompose the problem into two iteratively coupled ones. This yields the existing algorithms and indicates how to develop a new algorithm to solve "policy constrained problems".

Chapter II is devoted to risk-indifferent Markov Decision Processes. After reviewing previous work, namely, Howard's algorithm and the Linear Programming formulation, we embark upon formulating policy constraints. Then the Lagrange multiplier formulation is outlined and pursued to its consequences. This leads to the development of an efficient algorithm, along the VD-PI lines, whose convergence is proved. The algorithm is applied to Howard's famous taxicab example after policy constraints are introduced to it. All the foregoing deals with completely ergodic Markov processes in which all states are recurrent. We outline how the algorithm is modified when it encounters coupled

states which are transient. We also discuss periodic Markov processes.

Chapter III is devoted to risk-sensitive Markov Decision Processes. As in chapter II, Howard's and Matheson's algorithm is reviewed, a Lagrange multiplier formulation is developed, and an algorithm emerges. Its convergence is proved, and it is applied to the same previous example with a risk aversion coefficient. Finally, it is pointed out that transient states have no effect on the algorithm, and that since periodic processes are inherently deterministic problems, they are better solved by risk-indifferent methods.

Chapter IV deals with sensitivity analysis. The concepts of "constraint-indifferent" and "constraint-sensitive" optimal policies are introduced, and a procedure for computing the worth of individual constraints is outlined. It is explained by applying it to the example solved in chapter II.

In chapter V, we discuss modifications of the algorithm for problems having a large number of states and give the computational results of Howard's baseball problem. We also make some suggestions concerning future research.

CONTENTS

ILLUSTRATIONS

TABLES

ACKNOWLEDGMENTS

I would like to thank Prof. Ronald A. Howard, my principal research advisor, for introducing me to the world of Markov Decision Processes and for his help and guidance throughout the course of this work.

Thanks are also due to Prof. James E. Matheson and Prof. Edward J. Sondik, my readers, for their encouragement and the interesting points they raised, which made this a better work.

I would also like to thank Dr. Edward J. Cazalet and Dr. Dale M. Nesbitt for the illuminating discussions of Markov Decision Processes that we had at the Stanford Research Institute and my colleague in the Engineering-Economic Systems Department, Abdalla Abdelkader, for editing the English of the dissertation.

For financial support, I am indebted to the General Electric Company, with particular thanks to Dr. William R. DeHollander who supported my joining the Honor-Coop Program and encouraged me along the way. This research was also supported by the Advanced Research Projects Agency of the Department of Defense and was monitored by ONR under Contract N00014-76-C-0074 under subcontract from Decisions and Designs, Inc.

Last but not least, my thanks to Ursula G. Burger for her typing several drafts of the dissertation and for her support and patience throughout.

Chapter I

INTRODUCTION

A. Background and Motivation

The context of this work is completely ergodic Markov processes with rewards, which are allowed to run for an unlimited number of transitions. The basic problem we are concerned with is selecting, from among a set of such processes, one that yields the highest average return. Figure 1.1 illustrates such a Markov Decision Process. In states 1 and 3 we have three alternatives to choose from, and in state 2 we only have two alternatives. Associated with each alternative k in state i are three probabilities of transition p_{ij}^k $(j = 1, 2, 3)$ from state i to all states and the rewards r_{ij}^k of making said transitions under the given alternative. In each state, k can take on values of $1,2,\ldots,K_i$, where K_i is the number of alternatives available in state i (3, 2, and 3, respectively, for Figure 1.1). Once an alternative is chosen in each state (i.e., k given a value between 1 and k_i in each state i), we have what is called a stationary policy P. The i^{th} component of P is alternative selected in state i. In Figure 1.1, e.g., we gave k the values 1, 2, and 2 in states 1, 2, and 3, respectively (i.e., we selected the first alternative in state 1 and the second alternative in states 2 and 3). In other words, we have a policy

$$P = (1,2,2)$$

We denote the i^{th} component of P by $P(i)$. Thus, for the above policy $P(1) = 1$, $P(2) = 2$, $P(3) = 2$.

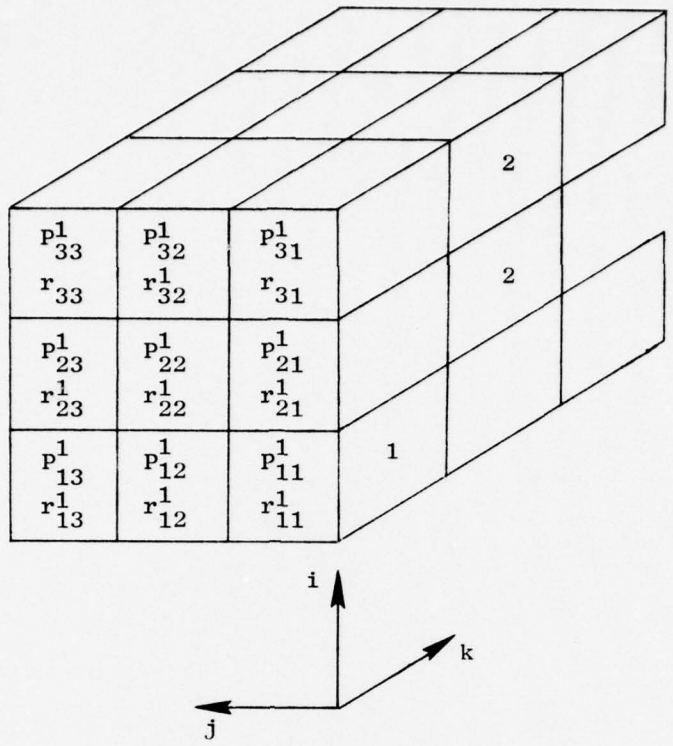


Fig. 1.1. THREE-STATE MARKOV DECISION PROCESS.

Such a policy is stationary in the sense that every time the process is in state i alternative P(i) is always selected.

Once a policy is selected, a Markov process with rewards is defined. The transition probability matrix and the reward matrix are composed of rows determined by each alternative in each state. For the policy mentioned in Figure 1.1, we get the transition probability matrix

$$P = \begin{bmatrix} p^1_{11} & p^1_{12} & p^1_{13} \\ p^2_{21} & p^2_{22} & p^2_{23} \\ p^2_{31} & p^2_{32} & p^2_{33} \end{bmatrix}$$

with a similar reward matrix R.

For a completely ergodic Markov process, the limiting state probabilities are independent of the starting state. They give the average number of stages the process spends in each state and are given by [4, 5]

$$\Pi^T \cdot P = \Pi^T$$

$$\sum_{i=1}^{N} \pi_i = 1 \tag{1.1}$$

where π is the column vector whose components π_i are the limiting state probabilities, P is the transition probability matrix of the process, and N is the number of states. Also associated with a Markov process is the vector q of immediate expected rewards defined by

$$q_i = \sum_{j=1}^{N} p_{ij} r_{ij} \tag{1.2}$$

For a risk-indifferent decision maker, Howard [4,5] has shown that if the Markov process is allowed to run for an unlimited number of transitions, the average reward of the process per transition, hereafter to be called the gain of the process, is given by

$$g = \sum_{i=1}^{N} \pi_i q_i \tag{1.3}$$

where the q_i are given by (1.2) and the π_i are given by (1.1) for the specified process (i.e., the selected policy).

For the problem illustrated in Figure 1.1, we have 18 policies to select among. Each results in a process for which (1.1) can be solved;

then (1.2) and (1.3) are used to compute the gain. A "brute force" method for selecting the optimal policy would be to do this for all 18 policies. However, the number of policies increases astronomically as the number of alternatives increases. Actually, the number of policies is $\prod_{i=1}^{N} K_i$. Thus, in the above example, if an additional alternative were introduced in state 2, we would immediately have 9 more policies to take into account. We are hence faced with an essentially combinatorial problem.

Howard [4,5] devised an extremely efficient iterative algorithm which exploits certain features of this problem to solve it. First, a value determination (VD) is made for a given policy. The VD consists of computing "relative values" v_i for each state, under the given policy, and the gain of that policy from

$$g + v_i = q_i + \sum_{j=1}^{N} p_{ij} v_j \qquad i = 1,2,3, \ldots, N \tag{1.4}$$

Then, using the v_i, an attempt is made to improve the policy, i.e., detect a policy of larger gain than the current one. To this end, test quantities are defined in each state for each alternative. Then, the alternative yielding the largest test quantity in each state is selected. Formally,

$$P(i) = \left\{ a : t_i^a = \max_{k=1,2,\ldots,K_i} \left[q_i^k + \sum_{j=1}^{N} p_{ij}^k v_j \right] \right\}$$

$$i = 1,2, \ldots, N \tag{1.5}$$

If the policy improvement (PI) does not change the current policy, it is the optimal policy. Otherwise, VD and PI, i.e., (1.4) and (1.5), alternate until we converge to the optimal policy. The efficiency of the VD-PI algorithm results from reducing the combinatorial problem of simultaneously selecting different alternatives in different states to a set of discrete maximization problems which select the alternatives in each state independently of other states.

Mine and Osaki [9] formulated the risk-indifferent Markov Decision Process as a linear program (LP). They showed that the VD is the solution of a dual problem, whence the relative values v_i are the simplex multipliers. They also showed that the PI is a simplex iteration with at most N simultaneous pivoting operations. Using the LP formulation, Nesbitt [10] showed how the policies can be ordered according to gain.

For the risk-sensitive decision maker possessing an exponential utility function, Howard and Matheson [6] define a "disutility contribution" matrix Q, whose elements are given by

$$q_{ij} = p_{ij} e^{-\gamma r_{ij}} \tag{1.6}$$

where the p_{ij} and r_{ij} are the probabilities and rewards selected by a given policy, and γ is the risk aversion coefficient. They derive a certain equivalent gain given by

$$\tilde{g} = -\frac{1}{\gamma} \ln \lambda \tag{1.7}$$

where λ is the "maximal eigenvalue" of Q (the largest positive eigenvalue which exceeds the modulii of all other eigenvalues of Q). It

is this $\tilde{g}$ that has to be maximized. They devised an algorithm quite similar to the VD-PI. It consists of a policy evaluation (PE) phase, the counterpart of the VD. In PE, the utilities u_i pertaining to a given policy are computed by solving the eigenvalue problem

$$\lambda u_i = \sum_{j=1}^{N} q_{ij} u_j \qquad i = 1,2, \ldots, N \tag{1.8}$$

where the q_{ij} are given by (1.6) and λ is the maximal eigenvalue of the corresponding matrix Q.

Then a policy improvement (PI) phase is undertaken. It is identical to the PI of the risk-indifferent case, except that the test quantities are different. Here, a policy is selected such that

$$P(i) = \left\{ a : t_i^a = \max_{k=1,2,\ldots,K_i} \left[\sum_{j=1}^{N} q_{ij}^k u_j \right] \right\}$$

$$i = 1,2, \ldots, N \tag{1.9}$$

In all of the afore-mentioned work, no restrictions are made on the manner in which alternatives are selected in different states. It is assumed that the selection of an alternative in a given state has no effect on alternative selection in any other state. In other words, there is no interaction, or "coupling," between alternatives in different states. This is the feature that allows considering each state separately in the PI. However, it is an idealized situation that might or might not hold in real life. With the profusion of rules and regulations governing economic activities in this day and age, it might very

well turn out that alternative selection is not as "coupling-free" as the idealized situation envisions. (Later in this work, we give a simple example of how alternatives in different states could be coupled.)

Introducing inter-state dependence in alternative selection in effect imposes constraints on the policies. Some policies become "infeasible," and we have to select the optimal "feasible" policy, where "feasibility" means satisfying the constraints. This work strives to do exactly that, in a computationally efficient manner. Nesbitt [10] suggests that for policy constrained problems, we start with the optimal policy in the absence of constraints and go backwards in the ordering checking feasibility until we hit the feasible policy yielding the highest gain. The problem with this brute force method is that, once we get beyond the second-to-optimum policy, we have to evaluate an increasingly large number of policies for each step in the ordering process. Moreover, we first have to solve the unconstrained policy problem before we can solve the constrained policy one. Also, there has been no work on policy ordering for the risk-sensitive case. What is really needed is an algorithm that retains as much of the simplicity and efficiency of Howard's VD-PI and PE-PI algorithms as possible, while taking feasibility into account as it progresses from one feasible policy to a better one.

Another aspect of the constrained policy problem is that there has been no work done on formulating the constraints mathematically in a systematic manner. This we also strive to achieve.

Also of interest in a constrained policy problem, is the "sensitivity" of optimal policies to the policy constraints. This "sensitivity" can best be expressed in terms of value, i.e., how much a rational decision maker would be willing to pay to remove a constraint.

To summarize then, this work is mainly concerned with three things: the formulation of policy constraints; developing a VD-PI type of algorithm for completely ergodic, infinite time horizon Markov Decision Processes; and sensitivity analysis of optimal policies vis-a-vis the policy constraints.

B. Methods and Results

Our point of departure is the LP formulation. There, a quantity d_i^k is introduced. That quantity represents the conditional probability that, given the process is in state i, alternative k is selected. In the LP formulation, it is proved that for each state i, only one $d_i^k = 1$ and the rest are zero. This is exactly what we want. We will formulate both the Markov Decision Process and the policy constraints in terms of the d_i^k.

For policy constraints, we first concentrate on the "two-alternative-coupling" case. By this, we mean interaction between an alternative k in state i and another alternative ℓ in some other state j. The constraints will be expressed in terms of d_i^k and d_j^ℓ, to be denoted by a and b, respectively, for simplicity (a and b can only take on values of zero or unity). By exhaustion of all possible combinations and straightforward application of simple logic, we conclude that there can only be five different types of constraints:

$$\text{———} \qquad a + b \leq 1 \tag{1.10}$$

$$\text{<———>} \qquad a + b \geq 1 \tag{1.11}$$

$$\text{<═══>} \qquad a - b \geq 0 \tag{1.12}$$

$$a + b = 1 \tag{1.13}$$

$$a - b = 0 \tag{1.14}$$

The designations to the left of the constraints are those we use on a graph or table to indicate the type of constraint (as in Figure 1.2). Inequality (1.10) expresses the constraint stating that, at most, one of alternatives a and b is allowable in any feasible policy; (1.11) states that at least one has to be present; (1.12) states that alternative b is not allowable in any policy unless it is accompanied by a;

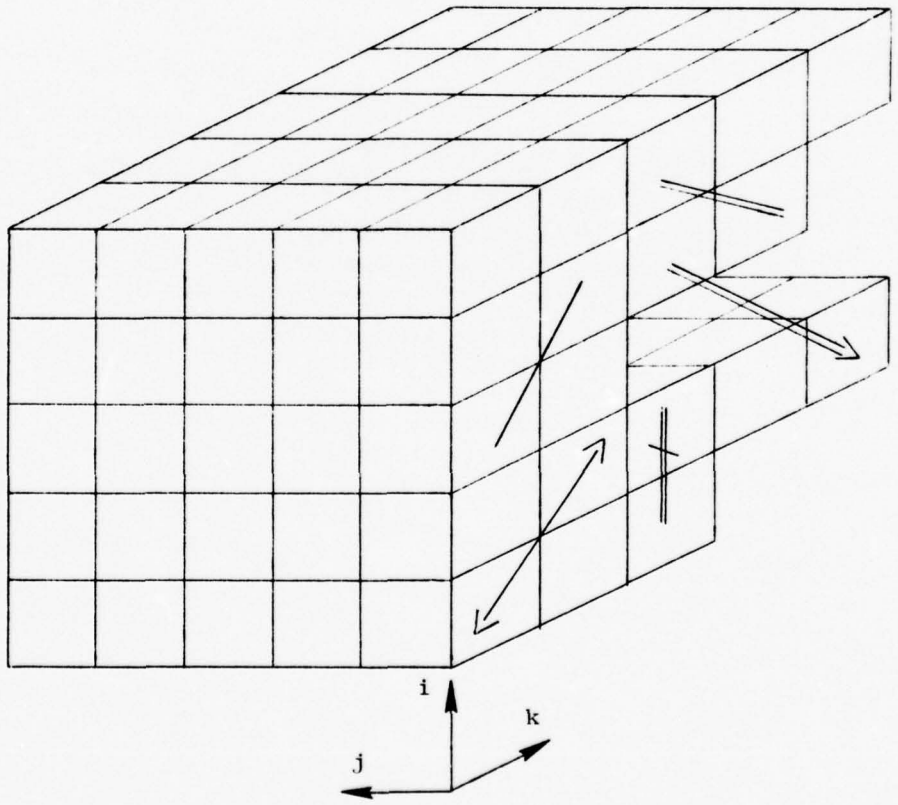


Fig. 1.2. FIVE-STATE POLICY CONSTRAINED MARKOV DECISION PROCESS.

(1.13) states that exactly one of the two alternatives must be present in any policy; (1.14) states that we can have either both or neither alternatives in any feasible policy. In Figure 1.2, e.g., alternative 4 in state 2 cannot be selected unless alternative 3 in state 1 is. Similarly, as regards alternative 2 in state 2 and alternative 1 in state 3, we cannot have more than one of them in any feasible policy. If more than two alternatives are coupled by any single constraint, we resort to the algebra of events to express the constraints as a Boolean expression, with truth and falsity being assigned the values unity and zero, respectively.

If the Boolean expression is an exclusive OR, it is equated to unity to give us our constraint. This is because, by definition, only one component of an exclusive or can be true. Otherwise, because of the possibility of more than one component being true simultaneously, the expression is set greater or equal to unity. Of course, some constraints involving more than one alternative can be intuitively translated into an algebraic relationship without recourse to the algebra of events. One such type of constraint is considered here because it is of special significance later on. Assume that we have a policy P whose first M components ($M > 1$) are a,b,c,...,m, respectively. If we want to make P and all policies that differ with P in exactly one of the first M components infeasible, we can do that by the simple constraint

$$d_1^a + d_2^b + \ldots + d_M^m \leq M - 2 \qquad (1.15)$$

Once we develop a methodology for expressing policy constraints, we turn to developing a VD-PI type of algorithm to handle policy constrained problems. To this end, we exploit the fact that the original problem we are faced with is a constrained optimization one, even in the absence of policy constraints. For the risk-indifferent case, the objective function to be maximized is the gain. The constraints are the equations defining the limiting state probabilities and those requiring the d_i^k to sum to unity in each state. If we have policy constraints, they will be additional relationships between d_i^k of different states.

The realization that we are dealing with a constrained optimization problem leads us to the Lagrange multiplier rule, which enables us to reduce the problem to two iteratively coupled problems defined on the associated Lagrangian. One of them turns out to be the VD, while the other is the maximization of the Lagrangian over the discrete set of feasible policies. It turns out that the relevant quantity to be maximized is the weighted sum of the test quantities. The weights are the limiting state probabilities. In the absence of policy constraints, the individual components of the sum can be maximized in order to maximize the sum. The weight π_i in each state then becomes irrelevant, and we just maximize the test quantities, which is what the PI does. Hence, the PI actually maximizes the Lagrangian, exploiting the absence of interstate coupling to decompose that maximization, an essentially combinatorial problem, to N much simpler maximizations. In the presence of policy constraints, however, such a decomposition can no longer be effected. We have to face the combinatorial maximization problem head on. This we do by adapting a branch and bound technique to our problem. This is a method whereby maximization is achieved without having to enumerate the feasible set.

We prove that such a method converges to a policy that maximizes the gain over the set of feasible policies differing with it in exactly one state. Therefore, we add a constraint of type (1.15) and start on another set. In this manner, we remove whole subsets from consideration without having to consider all the elements belonging to them.

To retain as much of the VD-PI as possible, we introduce the notions of "free" and "coupled" states. The former are states in which no alternatives are involved in any policy constraints. The "coupled" states are those which are not "free." As long as the original PI yields feasible policies, we do not use branch and bound. Failing that, we maximize over the free states by original PI and over the coupled states by branch and bound. This algorithm has two advantages. First, it achieves computational efficiency by sticking to Howard's PI as much as possible. Secondly, if the policy yielding the highest gain in the absence of policy constraints is not made infeasible by the introduction of these constraints, the algorithm detects it without having to exhaust the feasible policy set (by removing successive subsets). This is of particular significance for sensitivity analysis.

A Lagrange multiplier formulation is also applied to the risk-sensitive case. Here, the objective function is the maximal eigenvalue of the Q matrices associated with the feasible policies. The constraints are the eigenvalue problem defining the maximal eigenvalue, plus the policy constraints. As in the risk-indifferent case, Howard's and Matheson's PE-PI algorithm is shown to be the transformation of the original problem, via Lagrange multipliers, to two problems. The breakdown of the PI when policy constraints are introduced is shown, and a similar algorithm is developed.

Finally, sensitivity analysis is considered. As mentioned before, our algorithm is capable of detecting whether or not the policy constraints have any effect on the optimal policy in their absence, without having to solve the unconstrained policy problem. The value of removing individual policy constraints is explored for those situations where they affect optimal policy selection.

C. Outline

Chapter II is devoted to risk-indifferent Markov Decision Processes. After reviewing previous work, namely Howard's VD-PI algorithm and the LP formulation, we embark upon formulating policy constraints. Then the Lagrange multiplier formulation is outlined and pursued to its consequences. This leads to the development of an efficient algorithm, along the VD-PI lines, whose convergence is proved. The algorithm is applied to Howard's famous taxi cab example after policy constraints are introduced to it. All the foregoing deals with completely ergodic Markov processes, in which all states are recurrent. We outline how the algorithm is modified when it encounters coupled states which are transient. We also discuss periodic Markov processes.

Chapter III is devoted to risk-sensitive Markov Decision Processes. As in Chapter II, Howard's and Matheson's PE-PI algorithm is reviewed, a Lagrange multiplier formulation is developed, and an algorithm emerges. Its convergence is proved, and it is applied to the same previous example with a risk aversion coefficient.

Chapter IV deals with sensitivity analysis. The concepts of "constraint-indifferent" and "constraint-sensitive" optimal policies are introduced, and a procedure for computing the worth of individual

constraints is outlined. It is explained by applying it to the example solved in Chapter II.

In Chapter V, we discuss modifications of the algorithm for problems having a large number of states and give the computational results for Howard's baseball problem. We also make some suggestions concerning future research.

Chapter II

RISK-INDIFFERENT MARKOV DECISION PROCESSES

A. Introduction

In this chapter, we deal with risk-indifferent Markov decision processes, where we progress from unconstrained policies to constrained ones.

Section B deals exclusively with unconstrained policies. First, Howard's value determination-policy iteration algorithm (hereafter referred to as VD-PI) is developed. Then, the linear programming formulation of the problem is developed. Most of this section appears in the literature but is included here because it forms the foundation on which the results of this work are based. For example, the linear programming formulation provides us with the mathematical encoding of the process of selecting one alternative in each state. The conditional probability d_i^k of selecting alternative k, given the system is in state i, together with the important result that all d_i^k's are zero or unity, enables us to express policy constraints.

Section C deals with constrained policies. The definition of what we mean by constraints on the policies is spelled out. We mean interaction, or "coupling," between alternatives in different states, such as the selection of one alternative in a certain state preventing the selection of another alternative in some other state. First, we deal with "couplings" between two alternatives only, and we show that all such couplings reduce to five types of constraints. The general case is treated by the algebra of events. We give an example of a 3-alternative coupling and show how one of the 2-alternative couplings can be derived from the general case. Then, we show that a constrained policy problem can be

reduced to a number of LP's, which is unacceptable on account of that number being, more often than not, astronomical.

The approach we take to solve the problem is the realization that, even in the absence of policy constraints, we are faced, basically, with a constrained maximization problem. The objective function is the gain, and the constraints are the equations defining the limiting state probabilities. We consequently use a Lagrange multiplier (LM) formulation of the problem to reduce it to two unconstrained, iteratively coupled, problems. One of them turns out to be the VD. The other one is the Maximization of the Lagrangian L over the discrete set of feasible policies. This is an essentially combinatorial problem. We show that, in the absence of policy constraints, the lack of "coupling" facilitates the reduction of that problem to a number of simple discrete maximization problems, yielding Howard's PI. The presence of coupling, however, in the case of constrained policies destroys the reduction feature. Thus, we seek an efficient means for solving the combinatorial problem of maximizing L over the discrete set of feasible policies.

In Section C, we also point out the fact that maximizing L per se in PI does not guarantee selection of a policy having a higher gain, i.e., policy improvement. Rather, the fact that L and g (the gain we are trying to maximize) have the same value at the optimum and after each VD justifies trying to increase L. Improving the policy has to be guaranteed outside the Lagrangian framework. This we do by introducing a sufficient condition for improving the policy. This condition, which was derived by Howard [4,5], is satisfied by the maximization of L when no policy constraints are present. Actually, it is also sufficient to guarantee the VD-PI convergence to an optimum policy.

In Section D, we develop an algorithm for solving the problem, when faced with policy constraints, on the basis of the LM formulation of Section C. The algorithm is composed of the usual VD, plus a new PI which maximizes the gain over subsets of the set of feasible policies. It is based on the branch and bound (BB) technique for solving combinational problems [2,3,7]. One such method is adapted to our problem, and we show that it converges to a policy that maximizes the gain over the set of all feasible policies differing with it in exactly one state. This set is immediately removed from further consideration by a simple constraint, thus reducing the set of feasible policies. To increase computational efficiency, we introduce the notions of "free" and "coupled" states. A "free" state is one in which no alternative is coupled with any other alternative in any state, i.e., not involved in any policy constraints. A "coupled" state is one that has at least one alternative in it "coupled" with some alternative in another state, i.e., involved in some policy constraint(s). Our PI is invoked only if regular PI yields an infeasible policy. In this case, the free states are maximized by regular PI and the coupled states by branch and bound. In either case, the sufficient condition of Section C is satisfied, and we have an improved policy. This has a further advantage. If the policy constraints do not make the optimum policy (without constraints) infeasible, then it can be detected once it is encountered, and we do not have to exhaust the feasible policy set to reach the optimum.

The convergence of the developed algorithm is proved in Section D.

In Section E, we apply the algorithm to Howard's famous taxicab example after some policy constraints are imposed on it.

Sections B through E deal with policies that do not result in transient states; all states are recurrent. In Section F, we address the problem of transient coupled states. We also consider periodic Markov processes, and from them we infer that the manner in which we handle transient coupled states and how we obtain an initial feasible policy are both cases where we profess complete ignorance. In the former case, the zero value of π_i for a transient state obliterates our accumulated knowledge about that state, as far as the Lagrangian is concerned. In the latter case, the lack of an initial feasible policy is equivalent to complete ignorance of the Markov process we are dealing with.

B. Markov Decision Processes without Policy Constraints

The objective here is to select a stationary policy that maximizes the average return per transition of the completely ergodic system, where all states are recurrent, if it is allowed to make many transitions, i.e., over an infinite time horizon.

This is achieved by the value determination-policy improvement algorithm, which computes values for a given policy, then obtains a better policy, until the optimum policy is obtained.

1. Value Determination-Policy Improvement Formulation

a. Value Determination

We start out with a finite time horizon, i.e., allow the system to make only n transitions, then extend the horizon. We denote the expected total earnings in the next n transitions if the system is in state i by $v_i(n)$. To compute this quantity, we note that, if a transition is made to state j, its value will be the r_{ij} earned by the

transition plus the amount earned by starting in state j with one transition fewer remaining, i.e., $v_j(n-1)$. Thus, the previous amount must be weighed by the probability of making the transition from i to j, i.e., p_{ij}. Since the transitions from i are mutually exclusive, $v_i(n)$ is simply the sum of the weighed quantities. In other words,

$$v_i(n) = \sum_{j=1}^{N} p_{ij}\left[r_{ij} + v_j(n-1)\right] \qquad \begin{matrix} i = 1, \ldots, N \\ n = 1,2,3, \ldots \end{matrix} \tag{2.1}$$

If we define the immediate expected reward for a transition from state i by

$$q_i = \sum_{j=1}^{N} p_{ij} r_{ij} \qquad i = 1, \ldots, N \tag{2.2}$$

we can write Equations (2.1) as

$$v_i(n) = q_i + \sum_{j=1}^{N} p_{ij} v_j(n-1) \qquad \begin{matrix} i = 1, \ldots, N \\ n = 1,2,3, \ldots \end{matrix} \tag{2.3}$$

It can be shown [4,5] that, for a completely ergodic process, the asymptotic behavior of (2.3) is given by

$$v_i(n) = ng + v_i \qquad i = 1,2, \ldots, N \tag{2.4}$$

where v_i is the "relative value" of being in state i, and

$$g = \sum_{i=1}^{N} \pi_i q_i \tag{2.5}$$

Hence, g is the average return per transition of the system if it is allowed to make many transitions under a given policy. Such a policy is stationary in the sense that it does not depend on n, i.e., if we find ourselves in a given state, we select a particular alternative, irrespective of n. We are seeking a policy which maximizes this gain g. Once a policy is determined, the π's and q's are available, and hence (2.5) gives us the gain of that policy. However, we have no means of finding a better policy, if one exists. The key to this lies in (2.4). For large n,

$$v_i(n) = ng + v_i \qquad i = 1,2, \ldots, N$$

We also have that (2.3) holds for all n:

$$v_i(n) = q_i + \sum_{j=1}^{N} p_{ij} v_j(n - 1) \qquad i = 1,2, \ldots, N$$

Thus, for an infinite time horizon, we can substitute (2.4) into (2.3) to get

$$ng + v_i = q_i + \sum_{j=1}^{N} p_{ij} \left[(n - 1)\, g + v_j\right] \qquad i = 1,2, \ldots, N \qquad (2.6)$$

which, by virtue of $\sum_{j=1}^{N} p_{ij} = 1$, is reduced to

$$g + v_i = q_i + \sum_{j=1}^{N} p_{ij} v_j \qquad i = 1,2, \ldots, N \qquad (2.7)$$

Here, we have a set of N simultaneous linear equations in the N variables v_i and g, a total count of $N + 1$. We notice that adding a constant c to each v_i in (2.7) gives

$$g + v_i + c = q_i + \sum_{j=1}^{N} p_{ij}(v_j + c) \tag{2.8}$$

i.e.,

$$g + v_i = q_i + \sum_{j=1}^{N} p_{ij} v_j \tag{2.9}$$

But (2.9) are the original equations (2.7). Hence, the true values of v_i have no real significance in processes with infinite horizons. It is the differences between the v_i's that matter. This is shown by

$$v_i(n) = ng + v_i \tag{2.10}$$

$$v_j(n) = ng + v_j \tag{2.11}$$

whence

$$v_i(n) - v_j(n) = v_i - v_j \tag{2.12}$$

Thus, setting any one of the v_i's equal to zero, usually v_N, and solving (2.7) gives us the gain of the given policy and a set of v's we call the relative values of the policy. Those are used to select a policy having a higher gain than the given one.

b. Policy Improvement

Here, we also start with a finite horizon, then extend it by applying (2.4). If we define $v_i(n)$ as the total expected return

in n stages, if we start in state i and an optimal policy is followed, then applying the principle of optimality of dynamic programming, we have for any n

$$v_i(n+1) = \max_k \sum_{j=1}^{N} p_{ij}^k \left[r_{ij}^k + v_j(n) \right] \qquad n = 0,1,2, \ldots \qquad (2.13)$$

This may be written as

$$v_i(n+1) = \max_k \left[q_i^k + \sum_{j=1}^{N} p_{ij}^k v_j(n) \right] \qquad n = 0,1,2, \ldots \qquad (2.14)$$

Thus, if we have an optimal policy up to stage n, we can find the best alternative in state i at stage $n+1$ by maximizing

$$q_i^k + \sum_{j=1}^{N} p_{ij}^k v_j(n)$$

over all alternatives k in state i. For an infinite horizon, we substitute for $v_j(n)$ from (2.4) to obtain

$$q_i^k + \sum_{j=1}^{N} p_{ij}^k (ng + v_j)$$

as the test quantity to be maximized in each state.

The fact that $\sum_{j=1}^{N} p_{ij} = 1$, irrespective of k, reduces this to

$$q_i^k + \sum_{j=1}^{N} p_{ij}^k v_j + ng$$

Since ng does not depend on the policy that is selected, it is sufficient to maximize

$$q_i^k + \sum_{j=1}^{N} p_{ij}^k v_j \tag{2.15}$$

over all alternatives k in state i. Thus, for each state we select an alternative k, and this results in a new policy P. Thus, given a policy A, we solve

$$v_i^A + g^A = q_i^A + \sum_{j=1}^{N} p_{ij}^A v_j^A$$

by setting $v_N^A = 0$ to obtain v_j^A, $i = 1,2,\ldots,N-1$ and the gain g^A of policy A. Then, using the v's of policy A, we select an alternative k in each state i to maximize

$$q_i^k + \sum_{j=1}^{N} p_{ij}^k v_j^A$$

The alternatives k make up the new policy B, say. If it is identical to A, it is the optimum policy. Otherwise, a new iteration is started.

2. Linear Programming Formulation

The Markov decision process can also be formulated as a linear programming (LP) problem. To do this, we first recall that the function to be maximized is

$$g = \sum_{i=1}^{N} \pi_i(P)\, q_i(P) \tag{2.16}$$

where (2.16) is merely (2.5) rewritten so as to emphasize the dependence of the π's and q's on the policy and where the maximization takes place over all possible policies.

The next thing we do is to introduce a set of new variables d_i^k. Each d_i^k is the conditional probability of selecting alternative k, given that the system is in state i. (Those variables, hence, have to have a value of zero or unity. This, however, will be proved to result from the basic properties of linear programming, rather than setting it as a constraint.) Hence, in any state i, the expected immediate reward $q_i(P)$ is the sum of the q_i^k that result from selecting the various alternatives k in state i, weighted by the probabilities of selecting those alternatives, i.e.,

$$q_i(P) = \sum_{k=1}^{K_i} q_i^k d_i^k \tag{2.17}$$

whence our objective function becomes

$$g = \sum_{i=1}^{N} \sum_{k=1}^{K_i} \pi_i(P)\, q_i^k d_i^k \tag{2.18}$$

Here, the π's and d's are variables, whence our function is no more linear. However, using the definition of conditional probability, and denoting the joint probability *of being in state* i, and selecting alternative k by x_i^k, and recalling that π_i is the steady state probability of being in state i, we get

$$d_i^k = \frac{x_i^k}{\pi_i(P)}$$

which gives

$$x_i^k = \pi_i(P)\, d_i^k \tag{2.19}$$

The constraints on the d_i^k follow from their being probabilities

$$\sum_{k=1}^{K_i} d_i^k = 1 \qquad i = 1,2, \ldots, N \tag{2.20}$$

$$d_i^k \geq 0$$

Now, the original constraints on the π's were

$$\sum_{i=1}^{N} \pi_i(P)\, p_{ij}(P) - \pi_j(P) = 0 \qquad j = 1,2, \ldots, N \tag{2.21}$$

$$\sum_{j=1}^{N} \pi_j(P) = 1 \tag{2.22}$$

Using (2.19), it can be shown [9] that our linear programming problem is

$$\max_{x_i^k} \sum_{i=1}^{N} \sum_{k=1}^{K_i} q_i^k x_i^k \tag{2.23}$$

subject to

$$\sum_{i=1}^{N} \sum_{k=1}^{K_i} p_{ij}^k x_i^k - \sum_{k=1}^{K_j} x_j^k = 0 \tag{2.24}$$

$$\sum_{j=1}^{N} \sum_{k=1}^{K_j} x_j^k = 1 \qquad (2.25)$$

Now we proceed to prove that this LP yields values for d_i^k, which are either zero or unity, whence the d_i^k become the mathematical encoding of selecting one alternative in each state.

Theorem 2.1.

Any basic feasible solution to the LP defined by (2.23) through (2.25) has the property that for each i, there is only one k such that $x_i^k > 0$ and $x_i^k = 0$ otherwise.

Proof.

For the completely ergodic process rank, $(I-P) = N - 1$. Thus, one of the constraints (2.24) is redundant, and the rank of the constraints is N. From the basic properties of linear programming, it follows that any basic feasible solution has N positive variables x_i^k with the rest of the variables zero. Now, let us look at the equations of the constraints in detail (Table 2.1). Because $-p_{ij}$ $(i \neq j)$ is negative and $(1 - p_{ii})$ is positive, it follows that, in each of the first N equations, there has to be at least one x_i^k associated with a term $(1-p_{ii}^k)$ which is not zero, e.g., in the first equation if $x_i^k = 0$ for $k = 1,2, \ldots,k$, then the x_i^k which are not zero (i.e., positive) are all multiplied by negative coefficients and hence sum up to a negative number, contradicting the value of the R.H.S. Also, the fact that the first N equations contain a redundant one does not change the fact that it has to be satisfied. Thus, for each i, there has to be at least one $x_i^k > 0$

Table 2.1

LINEAR PROGRAMMING CONSTRAINTS

$$\left(1-p_{11}^{1}\right)x_1^1+\left(1-p_{11}^{2}\right)x_2^2+\cdots+\left(1-p_{11}^{K_1}\right)x_1^{K_1} \quad -p_{21}^{1}x_2^1-p_{21}^{2}x_2^2-\cdots-p_{21}^{K_2}x_2^{K_2} \quad -p_{N1}^{1}x_N^1-p_{N1}^{2}x_N^2-\cdots-p_{N1}^{K_N}x_N^{K_N} = 0$$

$$-p_{12}^{1}x_1^1-p_{12}^{2}x_2^2-\cdots-p_{12}^{K_1}x_1^{K_1} \quad \left(1-p_{22}^{1}\right)x_2^1+\left(1-p_{22}^{2}\right)x_2^2+\cdots+\left(1-p_{22}^{K_1}\right)x_2^{K_1} \quad -p_{N2}^{1}x_{N2}^1-p_{N2}^{2}x_N^2-\cdots-p_{N2}^{K_N}x_N^{K_N} = 0$$

$$\vdots \qquad \vdots \qquad \vdots$$

$$-p_{1N}^{1}x_1^1-p_{1N}^{2}x_1^2-\cdots-p_{1N}^{K_1}x_1^{K_1} \quad -p_{2N}^{1}x_2^1-p_{2N}^{2}x_2^2-\cdots-p_{2N}^{K_2}x_2^{K_2} \quad \left(1-p_{NN}^{1}\right)x_N^1+\left(1-p_{NN}^{2}\right)x_N^2+\cdots+\left(1-p_{NN}^{K_N}\right)x_N^{K_N} = 0$$

$$x_1^1+x_1^2+x_1^{K_1} \quad + \quad \cdots \quad \cdots \quad x_N^{K_N} = 1$$

for some k. If, for some state i, more than one $x_i^k > 0$, then there remains less than $(N-1)$ nonzero x_i^k for the remaining $(N-1)$ states. This would mean that, for some i, all $x_i^k = 0$, contradicting the fact that at least one such $x_i^k > 0$. Thus, for each i, there can be at most one k such that $x_i^k > 0$. "At most one" and "at least one" mean "only one."

The following corollary to this theorem provides us with the result we sought to prove.

Corollary.

Any basic feasible solution to the LP defined by (2.23) through (2.25) yields a pure stationary strategy, i.e., for each i, $d_i^k = 1$ for some k and zero for all other k.

Proof.

Equations (2.19) give

$$\sum_{k=1}^{K_i} x_i^k = \pi_i \sum_{k=1}^{K_i} d_i^k \tag{2.26}$$

Substituting (2.20) in (2.26),

$$\sum_{k=1}^{K_i} x_i^k = \pi_i \tag{2.27}$$

Substituting (2.27) in (2.19),

$$x_i^k = d_i^k \sum_{k=1}^{K_i} x_i^k \tag{2.28}$$

i.e.,

$$d_i^k = \frac{x_i^k}{\sum_{k=1}^{K_i} x_i^k} \qquad i = 1,2, \ldots, N \tag{2.29}$$

The theorem states that for any given i,

$$x_i^k = 0 \qquad k \neq \ell \qquad x_i^\ell > 0 \qquad 1 \leq \ell \leq K_i \tag{2.30}$$

Hence,

$$\sum_{k=1}^{K_i} x_i^k = x_i^\ell \tag{2.31}$$

Thus,

$$d_i^k = x_i^k / x_i^\ell \tag{2.32}$$

Whence, for $k \neq \ell$, $d_i^k = 0$, and

$$d_i^\ell = x_i^\ell / x_i^\ell = 1 \tag{2.33}$$

i.e., only one alternative is chosen in each state. This important result will be used when extending the Markov decision process to the case where the policies are constrained.

C. Markov Decision Processes with Policy Constraints

1. Formulation of Policy Constraints

Our point of departure here is the LP formulation for the unconstrained policies case. Rewriting (2.23) through (2.25) after substituting from (2.19) and (2.22), we get our original nonlinear problem:

$$\max \sum_{i=1}^{N} \pi_i \sum_{k=1}^{K_i} q_i^k d_i^k \tag{2.34}$$

subject to

$$\pi_j - \sum_{i=1}^{N} \pi_i \sum_{k=1}^{K_i} p_{ij}^k d_i^k = 0 \qquad j = 1,2, \ldots, N \tag{2.35}$$

$$\sum_{i=1}^{N} \pi_i \sum_{k=1}^{K_i} d_i^k = 1 \tag{2.36}$$

$$\left.\begin{aligned} \sum_{k=1}^{K_i} d_i^k &= 1 \\ d_i^k &\geq 0 \end{aligned}\right\} \qquad i = 1,2, \ldots, N \tag{2.37}$$

Note that (2.34) through (2.37) define the same problem as (2.23) through (2.25). Hence, whatever applies to (2.23) through (2.25) applies to (2.34) through (2.37). Specifically, we know beforehand that in the solution of (2.34) to (2.37) the d_i^k are either zero or unity, with $d_i^k = 1$ for only one k in each state i. The significance of this will become apparent later.

Now, we introduce constraints on policies. By constraints, we mean interaction, or coupling, between alternatives in different states. For example, it might happen that selecting alternative j, when the system is in state i, prevents the selection of alternative ℓ in state k. Thus, any policy having $P(i) = j$ and $P(k) = \ell$ is nonfeasible. Assuming that the mathematical encoding of alternative selection is valid,

i.e., d_i^k is zero or unity and is unity for only one k in each i, the above constraint may formally be expressed as

$$d_i^j + d_k^\ell \leq 1 \tag{2.38}$$

(2.38) plus $d_i^k = 0$ or 1 imply that no more than one of d_i^j and d_k^ℓ can be unity. Of course, they can both be zero.

Now we consider the encoding of policy constraints in general. First, we handle constraints that only couple two alternatives in different states, i.e., d_i^j and d_k^ℓ, for example. We shall hereafter refer to such constraints as binary constraints. We will show that no matter how the constraint is stated, it reduces to one of five relations.

Theorem 2.2.

Any policy constraint consisting of an interaction between alternative j in state i and alternative ℓ in state k can be expressed as one of the following:

$$a + b \geq 1 \tag{2.39}$$

$$a - b \geq 0 \tag{2.40}$$

$$a + b \leq 1 \tag{2.41}$$

$$a + b = 1 \tag{2.42}$$

$$a - b = 0 \tag{2.43}$$

where $a = d_i^j$ and $b = d_k^\ell$, and both are either zero or unity.

Proof.

We go about proving the above by simply exhausting all possibilities. Since a and b can both have only one of two values, the pair (a,b) cannot have more than four values, and any constraints merely limit the number of values that pair can have. Thus, we translate the constraint as outlawing certain of those values. First, we deal with the trivial cases.

Allowing all values (i.e., outlawing none) is equivalent to saying that we have no constraints, while outlawing all four values is a contradiction. The pair (a,b) is assured to exist and belong to the set $\{(0,0),(0,1),(1,0),(1,1)\}$. Outlawing three values and only allowing one is a case where we do not need any constraints. This is because we are saying that a value has been assigned to both a and b. If the value of a is zero, say, it means that alternative j in state i is not allowed. Thus, we just discard it. (Actually, this is a contradiction on the part of the decision maker. On the one hand he is saying that there is a number of alternatives available in state i, and on the other hand he is saying that one of those alternatives does not exist.) Likewise, if the value of a is unity, this means alternative j will always be selected in state i, whence we should discard all other alternatives in that state. (Yet, another contradiction; hereafter, whenever the value of a or b is predetermined by a constraint, we will consider that to be a contradiction and point it out.) What applies to a applies to b in the foregoing. Hence, we are left with two cases, namely those where only one or two pair values are outlawed.

Inequality (2.39) outlaws (0,0) and allows the three other possible values. This constraint can be stated as follows: any policy has to

have either a or b, or both. (2.40) outlaws the pair (1,0) which in plain English says that, if alternative b is not selected, then neither can a. (If the constraint is the other way around, i.e., not selecting a prevents selection of b, merely renaming a and b makes (2.40) applicable.) Inequality (2.41) outlaws (1,1) which is the type of constraint we already discussed (2.38). This exhausts the case where only one value of the pair (a,b) is outlawed. Equation (2.42) outlaws (0,0) and (1,1); in effect, it says that at least one of a and b must be selected, but the selection of one prevents selecting the other. Equation (2.43) outlaws (1,0) and (0,1); the type of constraint which says that selecting (nonselecting) one alternative necessitates selecting (nonselecting) the other. There remain, however, four combinations of values that have not been outlawed by any of (2.39) through (2.43). We show that they represent contradictions. Outlawing (0,0) and (0,1) means that the only feasible values are (1,0) or (1,1). But, here, $a = 1$, which we previously showed represents a contradiction on the part of the decision maker. Likewise, outlawing (1,0) and (1,1) leaves us with (0,0) or (0,1) implying $a = 0$. In the same manner, outlawing (0,0) and (1,0) implies $b = 1$, while outlawing (0,1) and (1,1) implies $b = 0$.

If the policy constraint involves more than two alternatives interacting with each other, we resort to the algebra of events to obtain a logical (or Boolean) expression for the constraint and then transform it into an algebraic constraint. An example illustrates this. Suppose we have three alternatives, the selection of each being denoted by the events A, B, and C, respectively (each alternative being, of course, in a different state). Not selecting an alternative will be denoted by the complement, e.g., A'. The values of the d_i^j will be denoted by

a,b,c. Assume that the constraint is that A and B cannot occur simultaneously unless C also occurs. This means that ABC' is outlawed. Hence, the Boolean expression that has to be true is

$$(ABC')' = A' + B' + C \tag{2.44}$$

If the values of a, b, and c are to represent the events A, B, and C, then the values representing A', B', and C' are (1 - a), (1 - b), and (1 - c), respectively (since a,b,c can only be zero for nonselection and unity for selection). The Boolean expression (2.44) is false only if all of its components are false (i.e., of value zero). Thus, algebraically we want the corresponding values to sum to something other than zero. This means

$$\begin{aligned} (1 - a) + (1 - b) + c &\geq 1 \\ -a - b + c &\geq -1 \\ a + b - c &\leq 1 \end{aligned} \tag{2.45}$$

Two things have to be noted here. First, if the reduction of the Boolean expression to its minimal sum involves intersections of events, then the algebraic constraint corresponding to it will involve products. Secondly, (2.45) was derived by requiring the L.H.S. representing (2.44) to be greater or equal to one. This is because truth of any component is sufficient to establish the truth of the whole expression, whence the truth of more than one component causes the sum to exceed unity. This does not hold, however, if the Boolean expression is an exclusive OR. There, only one component can be true, whence the sum of values can never exceed unity. In this case, the equivalent of (2.45) is derived by setting the sum equal to unity. Now, we formally derive the previous.

We are interested in "translating" a Boolean expression representing combinations of events into an algebraic expression. By "translation," we specifically mean that we are seeking an algebraic expression which holds if, and only if, the corresponding Boolean expression is true. To this end, we start by defining algebraic variables to correspond with the events. Since an event X has only two possible values (true and false, representing the event's occurrence or lack of it), we define an associated algebraic variable x which can only take on the values 1 and 0. Thus:

Definition.

Let X be an event. Its associated algebraic variable is a real number x restricted to the values 1 and 0 such that X is true if and only if $x = 1$.

Hereafter, we will denote the values true and false by T and F.

Proposition 2.1.

If X is an event whose associated algebraic variable is x, then the algebraic variable associated with X', the complement of X, is $1 - x$.

Proof.

Let $Y = X'$, $y = 1 - x$

Then $Y = T \iff X = F$

We already have $X = T \iff x = 1$

Assume $Y = T$

Then $X = F \Rightarrow x \neq 1 \Rightarrow x = 0 \Rightarrow y = 1 - x = 1$

Hence $Y = T \Rightarrow y = 1$

Assume $y = 1$

Then $x = 1 - y = 0 \neq 1 \Rightarrow X = F \Rightarrow Y = T$

i.e., $y = 1 \Rightarrow Y = T$

So, we have proved that $Y = T \Leftrightarrow y = 1$ which is the definition of the associated algebraic variable.

The importance of Proposition 2.1 is that, whenever we have the complement of an event, we can substitute the "algebraic complement" of its associated algebraic variable. In other words, if whenever we encounter X' we set $Y = X'$ in the Boolean expression, then the resulting y in the algebraic expression can be set to $1 - x$. This enables us to only consider uncomplemented events, without loss of generality. What follows applies in general if the mentioned substitutions are made.

Now we consider a Boolean expression composed of the sums (OR's) of products (AND's). First, we consider products.

Proposition 2.2.

$B = ABC \ldots Z$ is true $\Leftrightarrow$ $abc \ldots z = 1$

Proof.

$ABC \ldots Z = T \Leftrightarrow A = B = C = \ldots = Z = T$ from rules of Boolean Algebra

$\Leftrightarrow a = b = c = \ldots = z = 1$ from definition

Proposition 2.3.

$$B = B_1 + B_2 + \ldots + B_N \text{ is true } \Leftrightarrow b_1 + b_2 + \ldots + b_n \geq 1$$

where B_i is a Boolean product of events and b_i is the corresponding product of the associated algebraic variables.

Proof.

From the rules of Boolean Algebra, we have

$$B = T \Leftrightarrow \exists i \ni B_i = T \Leftrightarrow b_i = 1$$

If only one such i exists, then the b_i sum to unity; otherwise, their sum exceeds unity.

Corollary.

Let $B = B_1 + B_2 + \ldots + B_N$ where B_i is the Boolean product of events. Let

$$I = \{1,2, \ldots, N\}$$

If $$B_i B_j = F \qquad \forall i \in I$$

Then $$B = T \Leftrightarrow \sum_{i=1}^{N} b_i = 1$$

Proof.

Assume that $$\exists_{i,j} \ni B_i = T, \qquad B_j = T$$

Then $$B_i B_j = T$$

But this contradicts $B_i B_j = F$, $\forall_{i,j}$. Hence, there cannot exist more than one $i \ni B_i = T$. In this case,

$$B = T \iff \exists_i \ni B_i = T \text{ and } B_j = F \quad \text{for } j \neq i$$

$$\iff \exists_i \ni b_i = 1 \text{ and } b_j = 0 \quad \text{for } j \neq i$$

$$\iff \sum_{i=1}^{N} b_i = 1$$

The preceding is the case of an exclusive OR. A special case of this is when only one of N events is allowed to be true. This can be detected by the special form the Boolean expression takes. It is formed of the sum of N products, where each product is formed of an event and the complements of the remaining $N-1$ events. For example, for three events A, B, C:

$$AB'C' + A'BC' + A'B'C$$

In this case, $a + b + c = 1$. This is because the only way the Boolean expression can be true is for only one event to be true, and the rest false. This happens if, and only if, one associated algebraic variable is unity and the rest are zero. For instance, (2.43) may be derived in this fashion. Here, the two alternatives A and B are either both selected (1,1) or both not selected (0,0). The Boolean expression for this is

$$AB + A'B'$$

which is an exclusive OR. Moreover, it is of the special form we just illustrated. Putting $C = B'$, we get

$$AC' + A'C$$

Then, the algebraic expression is

$$a + c = 1$$

Since

$$C = B' \Rightarrow c = 1 - b$$

we get

$$a + 1 - b = 1$$

i.e.,

$$a - b = 0$$

Note that the "translation" is not unique. The general procedure outlined in the proposition, however, always yields a valid "translation." For instance, an alternative form of (2.43) could be derived by applying the general procedure to the Boolean expression

$$AB + A'B'$$

We would get

$$ab + (1 - a)(1 - b) \geq 1$$

i.e.,

$$2ab - a - b \geq 0$$

For a and b restricted to 0 and 1, this inequality defines exactly the same constraint as (2.43) (substituting the four possible values, verifies this). If we had noted that the Boolean expression is an exclusive OR, we would have obtained

$$2ab - a - b = 0,$$

i.e.,

$$2ab - a^2 - b^2 = 0, \qquad \text{because } a^2 = a,\ b^2 = b$$

Thus,

$$(a - b)^2 = 0$$

$$a - b = 0$$

which, again, is equivalent to (2.43).

Because of the *nonuniqueness* of "translation," we suggest that the general procedure be used only as a last resort, in order that we get the simplest possible constraints. Unless the constraint is too complicated to intuitively translate, it is expressed via algebra of events. A check is made to see if it is of the special form mentioned previously. If so, the d_i^k's are summed to unity. Otherwise, the corresponding d_i^k is substituted for its event and $1 - d_i^k$ for the complement of the event. If the Boolean expression is an exclusive OR, the resultant algebraic expression is equated to unity; otherwise, it is set greater than or equal to unity. In this manner, any policy constraint can be translated into an algebraic constraint under the assumption that the d_i^k involved in the constraints are all either zero or unity.

Note that the general procedure for translating Boolean to algebraic expressions only applies to sums of products.

The foregoing then implies that the Markov decision process with policy constraints can be formulated as (2.34) through (2.37) plus some extra constraints on a subset of the d_i^k.

Now, we proceed to prove an important result.

Proposition 2.4.

For the problem

$$\max \sum_{i=1}^{N} \pi_i \sum_{k=1}^{K_i} q_i^k d_i^k \tag{2.34}$$

subject to

$$\pi_j - \sum_{i=1}^{N} \pi_i \sum_{k=1}^{K_i} p_{ij}^k d_i^k = 0 \qquad j = 1,2, \ldots, N \tag{2.35}$$

$$\sum_{i=1}^{N} \pi_i \sum_{k=1}^{K_i} d_i^k = 1 \tag{2.36}$$

$$\left.\begin{aligned} \sum_{k=1}^{K_i} d_i^k &= 1 \\ d_i^k &\geq 0 \end{aligned}\right\} \qquad i = 1,2, \ldots, N \tag{2.37}$$

$$\left.\begin{aligned} C_\ell\left(d_i^k\right) &\leq 0 \qquad \ell = 1,2, \ldots, m \qquad (i,k) \in S_1 \\ C_p\left(d_i^k\right) &= 0 \qquad p = 1,2, \ldots, q \qquad (i,k) \in S_2 \end{aligned}\right\} \tag{2.46}$$

Where S_1 and S_2 are subsets of $S = \{(i,k)\}$, the following holds.

If d_i^k is assumed to only take on values of zero or unity for $(i,k) \in S_1 \cup S_2$, then d_i^k will only take on values of zero or unity for $(i,k) \in S$.

Proof.

Since the set S of all possible alternatives is finite, all of its subsets are finite. Moreover, restricting d_i^k to values of zero or unity makes the possible number of ways in which (2.46) can be satisfied finite. In other words, we have a finite number of values for the $(m+q)$-tuples representing the values of d_i^k for $(i,k) \in S_1 \cup S_2$; of those values, only a limited number will satisfy (2.46) unless the constraints are contradictory.

Now, for each $(m+q)$-tuple satisfying (2.46), the d_i^k's involved have certain given values (zero or unity). Substituting those values in (2.34) through (2.37) yields an identical problem with, possibly, fewer constraints. The structure of the problem, however, is the same. Thus, it is our LP problem (2.23) through (2.25) for which we proved that the d_i^k's are all zero or unity. Hence, our problem can be reduced to a finite number of LP's, each defined on a subset of the set defined by the constraints (2.35) through (2.37), and consequently yielding values of zero or unity for the d_i^k's not involved in the constraints (2.46).

The foregoing implies that we can formulate the constrained policy problem as a finite number of LP's corresponding to the number of ways (2.46) can be satisfied. We could then solve each problem (either as an LP or, even better, using the VD-PI algorithm), and the optimum policy would be that belonging to the problem yielding the highest gain. This, of course, is unacceptable from a computational point of view. Not only is the sub-problem of determining how many LP's we have a combinatorial one, but also the number of LP's we would have to solve could be astronomical. That is why we proceed to use the Lagrange multiplier method to reduce our constrained problem to two unconstrained ones.

2. Lagrange Multiplier Formulation

The Lagrange multiplier rule for constrained maximization problems provides us with a powerful technique for reducing the constrained problem to a number of unconstrained ones.

One form of the Lagrange multiplier rule is the following [8]: Let $X^* \in E^n$ maximize $f(x)$ subject to

$$C_i(x) = 0 \qquad i = 1,2, \ldots, m$$

Then there exist real numbers λ_i^*, $i = 1,2,\ldots,m$ such that the point $(x_1^*,x_2^*,\ldots,x_n^*,\lambda_1^*,\lambda_2^*,\ldots,\lambda_m^*) = (x^*,\lambda^*) \in E^{n+m}$ is a critical point of the function

$$L(x,\lambda) = f(x) + \sum_{i=1}^{m} \lambda_i C_i(x) \tag{2.47}$$

i.e.,

$$\nabla L(x^*,\lambda^*) = 0$$

Moreover,

$$L(x^*,\lambda^*) = f(x^*)$$

The appealing feature of this form of the Lagrange multiplier rule is that it transforms the constrained maximization of the function f into finding critical points of the "Lagrangian" L, as defined by (2.47), which is unconstrained. Of course, the cost incurred here is the increase of dimensionality from n to $n+m$. However, the Lagrangian offers the possibility of an iterative algorithm. A set of λ's is chosen for a point x; then x and λ are changed successively, until we get to the critical point. The fact that at the solution the values of the

original function and the Lagrangian are equal, plus the fact that we are trying to maximize such a value, can be used to move from one set of variables to a new one. We make the increase of L our objective. Actually, we could divide the variables (x,λ) whichever way we choose, alternatively changing each set until we arrive at the solution. However, the convergence of such an iterative algorithm to the desired maximum has to be proved. For one thing, the proposition does not state that any critical point of L is, necessarily, a constrained maximum of f. Only the converse is guaranteed. Moreover, nothing in the proposition guarantees convergence even to a critical point of L. It merely establishes the existence of such a point if the function f has a constrained maximum. With this in mind, we proceed to apply the Lagrange multiplier rule to solving the Markov decision process.

First, we rewrite the form of the general problem (including policy constraints):

$$\max \sum_{i=1}^{N} \pi_i \sum_{k=1}^{K_i} q_i^k d_i^k \tag{2.34}$$

subject to

$$\sum_{i=1}^{N} \pi_i \sum_{k=1}^{K_i} p_{ij}^k d_i^k - \pi_j = 0 \qquad j = 1,2, \ldots, N \tag{2.35}$$

$$1 - \sum_{i=1}^{N} \pi_i \sum_{k=1}^{K_i} d_i^k = 0 \tag{2.36}$$

$$\sum_{i=1}^{K_i} d_i^k - 1 = 0 \tag{2.37}$$

$$\left.\begin{aligned} C_\ell\left(d_i^k\right) &\le 0 \qquad \ell = 1,2,\ldots,m \qquad (i,k) \in S_1 \\ C_p\left(d_i^k\right) &= 0 \qquad p = 1,2,\ldots,q \qquad (i,k) \in S_2 \end{aligned}\right\} \tag{2.46}$$

where S_1 and S_2 are subsets of the set $S = \{(i,k)\}$ of (i,k) pairs defining each alternative in each state.

We will treat the problem as follows. Since we have already shown that (2.46) merely restricts us to subsets of S, and that setting $d_i^k = 0$ or 1 for $(i,k) \in S_1 \cup S_2$ results in the rest of d_i^k being unity or zero, we will consider (2.34) through (2.37) and then, when we change the d_i^k, we will take (2.46) into consideration and only choose zero or unity for those d_i^k involved in (2.46).

Corresponding to (2.34) through (2.37), we have a Lagrangian L involving $2N+1$ multipliers λ. We will name them according to the constraints, whence they will later acquire significance. For (2.35), we define $v_1,\ldots,v_N$ (i.e., $\lambda_1,\ldots,\lambda_N$). For (2.36), we define the multiplier g (i.e., λ_{N+1}). For (2.37), our multipliers $(\lambda_{N+2},\ldots,\lambda_{2N+1})$ will be named $\beta_1,\ldots,\beta_N$.

Hence, our Lagrangian is

$$L = \sum_{i=1}^{N} \pi_i \sum_{k=1}^{K_i} q_i^k d_i^k + \sum_{j=1}^{N} v_j \left[\sum_{i=1}^{N} \pi_i \sum_{k=1}^{K_i} p_{ij}^k d_i^k - \pi_j \right]$$

$$+ g\left[1 - \sum_{i=1}^{N} \pi_i \sum_{k=1}^{K_i} d_i^k\right] + \sum_{i=1}^{N} \beta_i \left[\sum_{k=1}^{K_i} d_i^k - 1\right] \tag{2.48}$$

It is a function of the limiting state probabilities π_i, the conditional probabilities d_i^k of selecting alternative k given state i, and the Lagrange multipliers v_j, g, and β_i.

All in all, we have $3N + \sum_{i=1}^{N} K_i + 1$ variables (the dimension of the Euclidean space over which L is defined). Our objective is to find a critical point for L. This we do iteratively. Starting with d_i^k's that satisfy (2.37), we set the partial derivatives of L with respect to the π's equal to zero. This gives values for the v's and g (and also, as we will show, for the π's; this is equivalent to setting the partial derivatives w.r.t. the v's and g to zero). Then, we use the LP result which tells us that we know beforehand that the d_i^k are zero or unity. The way we use it is to change the d_i^k in the zero-unity subspace, rather than set partial derivatives w.r.t. d_i^k equal to zero. The partial derivatives of L w.r.t. the π's is

$$\frac{\partial L}{\partial \pi_i} = \sum_{k=1}^{K_i} q_i^k d_i^k - v_i + \sum_{j=1}^{N} v_j \sum_{k=1}^{K_i} p_{ij}^k d_i^k - g \sum_{k=1}^{K_i} d_i^k$$

$$i = 1,2, \ldots, N \qquad (2.49)$$

Given a policy P, i.e., values of d_i^k satisfying (2.37) (and for constrained policies, (2.46) also), we get only one k for each i, and (2.49) reduces to

$$\frac{\partial L(P)}{\partial \pi_i} = q_i(P) - v_i + \sum_{j=1}^{N} p_{ij}(P)\, v_j - g \qquad i = 1,2, \ldots, N \qquad (2.50)$$

Setting the derivatives equal to zero gives us

$$q_i + \sum_{j=1}^{N} p_{ij} v_j = g + v_i \qquad i = 1,2, \ldots, N \qquad (2.51)$$

but (2.51) is the VD phase of the VD-PI algorithm. Thus, that phase is one in which Lagrange multipliers are updated. In (2.51), we have N equations in $N + 1$ unknowns. However, since (2.35) contains a redundant equation, any one of (2.35) can be discarded, which is equivalent to setting one of the v's to zero, whence (2.51) becomes a nonsingular system of equations. In matrix form, it can be written as

$$\begin{bmatrix} 1 - p_{11} & -p_{12} & \cdots & -p_{1N} & 1 \\ -p_{21} & 1 - p_{22} & \cdots & -p_{2N} & 1 \\ \cdot & \cdot & \cdots & \cdot & \cdot \\ \cdot & \cdot & \cdots & \cdot & \cdot \\ \cdot & \cdot & \cdots & \cdot & \cdot \\ -p_{N-1,1} & -p_{N-1,2} & \cdots & -p_{N-1,N} & 1 \\ -p_{N,1} & -p_{N,2} & \cdots & 1 - p_{NN} & 1 \end{bmatrix} \cdot \begin{bmatrix} v_1 \\ v_2 \\ \cdot \\ \cdot \\ \cdot \\ v_N \\ g \end{bmatrix} = \begin{bmatrix} q_1 \\ q_2 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ q_N \end{bmatrix}$$

Regarding the multiplication of a matrix by a vector as taking a linear combination of the matrix columns, the above states that we are attempting to form the q vector as a linear combination of the $n + 1$ columns of the matrix on L.H.S. The elements of the combination are the v's and g.

Since v_N is set to zero, however, this means that we can drop the N^{th} column to get our $N \times N$ system of equations:

$$\begin{bmatrix} 1 - p_{11} & -p_{12} & \cdots & -p_{1,N-1} & 1 \\ -p_{21} & 1 - p_{22} & \cdots & -p_{2,N-1} & 1 \\ \cdot & \cdot & \cdots & \cdot & \cdot \\ \cdot & \cdot & \cdots & \cdot & \cdot \\ \cdot & \cdot & \cdots & \cdot & \cdot \\ -p_{N,1} & -p_{N,2} & \cdots & -p_{N,N-1} & 1 \end{bmatrix} \cdot \begin{bmatrix} v_1 \\ v_2 \\ \cdot \\ \cdot \\ v_{N-1} \\ g \end{bmatrix} = \begin{bmatrix} q_1 \\ q_2 \\ \cdot \\ \cdot \\ \cdot \\ q_N \end{bmatrix}$$

Denoting the $N \times N$ matrix on the L.H.S. by $\tilde{P}$, we get

$$\begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_{N-1} \\ g \end{bmatrix} = [\tilde{P}]^{-1} \begin{bmatrix} q_1 \\ q_2 \\ \vdots \\ q_N \end{bmatrix} \tag{2.52}$$

Now we proceed to show that the π's can be obtained as a by-product of VD (which has been compactly stated by (2.52)). If we differentiate the Lagrangian w.r.t. the v's and g, we get (2.35) and (2.36), i.e.,

$$\frac{\partial L}{\partial v_j} = \sum_{i=1}^{N} \pi_i \sum_{k=1}^{K_i} p_{ij}^k d_i^k - \pi_j \tag{2.53}$$

$$\frac{\partial L}{\partial g} = 1 - \sum_{i=1}^{N} \pi_i \sum_{k=1}^{K_i} d_i^k \tag{2.54}$$

For a given policy P, if we set the partial derivatives equal to zero, we get our original definition of limiting state probabilities, namely,

$$\pi_j - \sum_{i=1}^{N} p_{ij}\pi_i = 0 \qquad j = 1,2, \ldots, N$$

$$\sum_{i=1}^{N} \pi_i = 1$$

Since the first N equations contain a redundant one, we can drop the N^{th} equation and get N equations in the N unknown π_i. In matrix form, this may be written as

$$\begin{bmatrix} 1 - p_{11} & -p_{21} & \cdots & -p_{N1} \\ -p_{12} & 1 - p_{22} & \cdots & -p_{N2} \\ \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ -p_{1,N-1} & -p_{2,N-1} & \cdots & -p_{N,N-1} \\ 1 & 1 & \cdots & 1 \end{bmatrix} \cdot \begin{bmatrix} \pi_1 \\ \pi_2 \\ \cdot \\ \cdot \\ \cdot \\ \pi_{N-1} \\ \pi_N \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \cdot \\ \cdot \\ \cdot \\ 0 \\ 1 \end{bmatrix}$$

But the matrix on the L.H.S. is the transpose of $\tilde{P}$. Hence, the π's are the solution of

$$\begin{bmatrix} \pi_1 \\ \pi_2 \\ \vdots \\ \pi_N \end{bmatrix} = \left([\tilde{P}]^{-1}\right)^T \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix} \tag{2.55}$$

Hence, the π's are the last column of $([\tilde{P}]^{-1})^T$, i.e., the transpose of the last row of $\tilde{P}^{-1}$. Since we compute $\tilde{P}^{-1}$ in the course of VD, this means that we actually have the π's without any additional computational effort whatsoever. The significance of this will become apparent when we consider policy constraints. Hence, setting the partial derivatives of L w.r.t. the v's and g equal zero, actually means setting all its partial derivatives (except for those involving the d_i^k's and their β's) equal to zero.

Now, we proceed to interpret the Lagrange multipliers. (2.53) and (2.48) imply that the v's are involved in L in the form

$$\sum_{j=1}^{N} v_j \frac{\partial L}{\partial v_j}$$

where $\partial L/\partial v_j$ gives the amount by which the j^{th} constraint on Π is violated. It is necessary that it be zero for all j. Otherwise, we do not have a critical point for L, whence it can never be a constrained maximum. Hence, v_j gives us the cost of violating the j^{th} constraint by one unit. But that constraint represents the equilibrium of probabilistic flows in the steady state. Thus, v_j is the value of being in state j (albeit a relative one). If, in that state, the equilibrium of probabilistic flows does not hold (e.g., by virtue of using P_{ij}'s belonging to a policy different than the one the π's were computed for), v_j gives us the cost per unit of "disequilibrium" for that state. It might be that we gain, in terms of the value of L, by doing that, i.e., we increase L. However, there is no guarantee that when we compute the new π's and v's we will get a net increase in L. This will be explained shortly.

The interpretation of g is straightforward. From (2.52), g is the inner product of the last row in $\tilde{P}^{-1}$ and the q vector. By virtue of (2.55), that row is merely the transpose of the π vector. Hence,

$$g = \Pi^T \cdot q = \sum_{i=1}^{N} \pi_i q_i$$

But this is the value of our original function we are trying to maximize. Hence, for a given policy, the value of the gain is one of the Lagrange multipliers that make the aforementioned partial derivatives equal to zero.

The VD, therefore, results in equating some of the partial derivatives of L to zero for a given policy. The remainder of those partial derivatives are not necessarily zero though. We would like to equate them to zero. Rather than do that, however, we can do better. We already know that d_i^k's have to be zero or unity for unconstrained policies (and for constrained policies if we set the extra ones to zero or unity). Hence, it would be more efficient to look upon this part of the maximization process as a discrete problem and try to increase the value of L (see [1] for example). To do this, we rewrite (2.48), rearranging the terms, so as to bring out the dependence on d_i^k.

$$L = \sum_{i=1}^{N} \pi_i \sum_{k=1}^{K_i} \left(q_i^k + \sum_{j=1}^{N} p_{ij}^k v_j \right) d_i^k - \sum_{j=1}^{N} v_j \pi_j$$

$$+ g\left(1 - \sum_{i=1}^{N} \pi_i \sum_{k=1}^{K_i} d_i^k\right) + \sum_{i=1}^{N} \beta_i \left[\sum_{k=1}^{K_i} d_i^k - 1 \right]$$

Our aim is to select the d_i^k such that we get the largest increase possible in L. Since we will be selecting them as zero or unity, according to our prior knowledge for unconstrained policies (and our forcing them for constrained ones), and since the π_i are held constant during improvement of the policy, we need only concern ourselves with the first term in L. The second one does not involve d_i^k, and the last two vanish. Thus, we concentrate on maximizing the quantity:

$$\sum_{i=1}^{N} \pi_i \sum_{k=1}^{K_i} \left(q_i^k + \sum_{j=1}^{N} p_{ij}^k v_j \right) d_i^k \qquad (2.56)$$

(2.56) is the inner product of two N-component vectors. The first is the vector $\Pi(P)$, as determined by the current policy P. Its components are nonnegative. The second vector is a variable. For each policy P', where P'(i) = k', the $d_i^{k'}$ satisfy (2.37), whence the summation over k reduces to one term for each state, namely $t_i^{k'} = [q_i^{k'} + \sum_{j=1}^{N} p_{ij}^{k'} v_j]$ the i^{th} component of the vector selected by P'. We will denote that vector by T(P'). Hence, the maximization of (2.56) reduces to selecting that vector T(P'), i.e., that policy P' which yields the largest value of inner product with the constant vector $\Pi(P)$. This is, essentially, a combinatorial problem. The absence of policy constraints reduces it to a much simpler problem. In the absence of such constraints, all possible vectors T(P') are allowable (i.e., feasible). There are $\prod_{i=1}^{N} K_i$ such vectors. Among them, is that vector for which

$$t_i = \max_{k=1,2,\ldots,K_i} \left[q_i^k + \sum_{j=1}^{N} p_{ij}^k v_j \right] \qquad i = 1,2,\ldots,N \qquad (2.57)$$

The maximization of (2.57) yields, for each i, an alternative K_i^*. Those alternatives make up a policy P^* with the corresponding vector $T(P^*)$. Now consider the inner product of $T(P^*)$ and $\Pi(P)$. Since each component of $T(P^*)$ is greater than the corresponding components of all other $T(P)$ and, since the components of Π are nonnegative, it immediately follows that P^* is the policy that maximizes (2.56). Hence, for unconstrained policies, the combinatorial problem of maximizing (2.56) reduces to the N discrete, "uncoupled," maximization problems (2.57). But (2.57) is the PI phase of the VD-PI algorithm. Thus, that phase is actually the maximization of L with respect to the d_i^k, using the values of Π and V for the current policy. This can be represented, schematically, as in Figure 2.1. There, we grouped the variables into three axes. The P's are represented by an axis, as are the v's and π's. Since the v's do not exist in the original constrained problem (2.34) through (2.37), its set of feasible points lies in the (Π,P) plane. Moreover, since the constraints yield unique values for Π, the problem reduces to selecting from a discrete set of points A_i in that plane. Regarding the Lagrangian, for each policy P, it is a function of Π and V. However, the VD results in points whose Π is identical to that of the given policy, whence the points a_i have the same Π component as the points A_i. As shown in Figure 2.1, the PI consists of moving from a_1, say, along the P direction, holding Π and V constant, to maximize L. This results in point b_2, say. The VD then takes us to a_2, from which we go into the PI etc.

A word of caution is necessary here. It pertains to setting up the increase in L as a criterion for selecting a policy having a

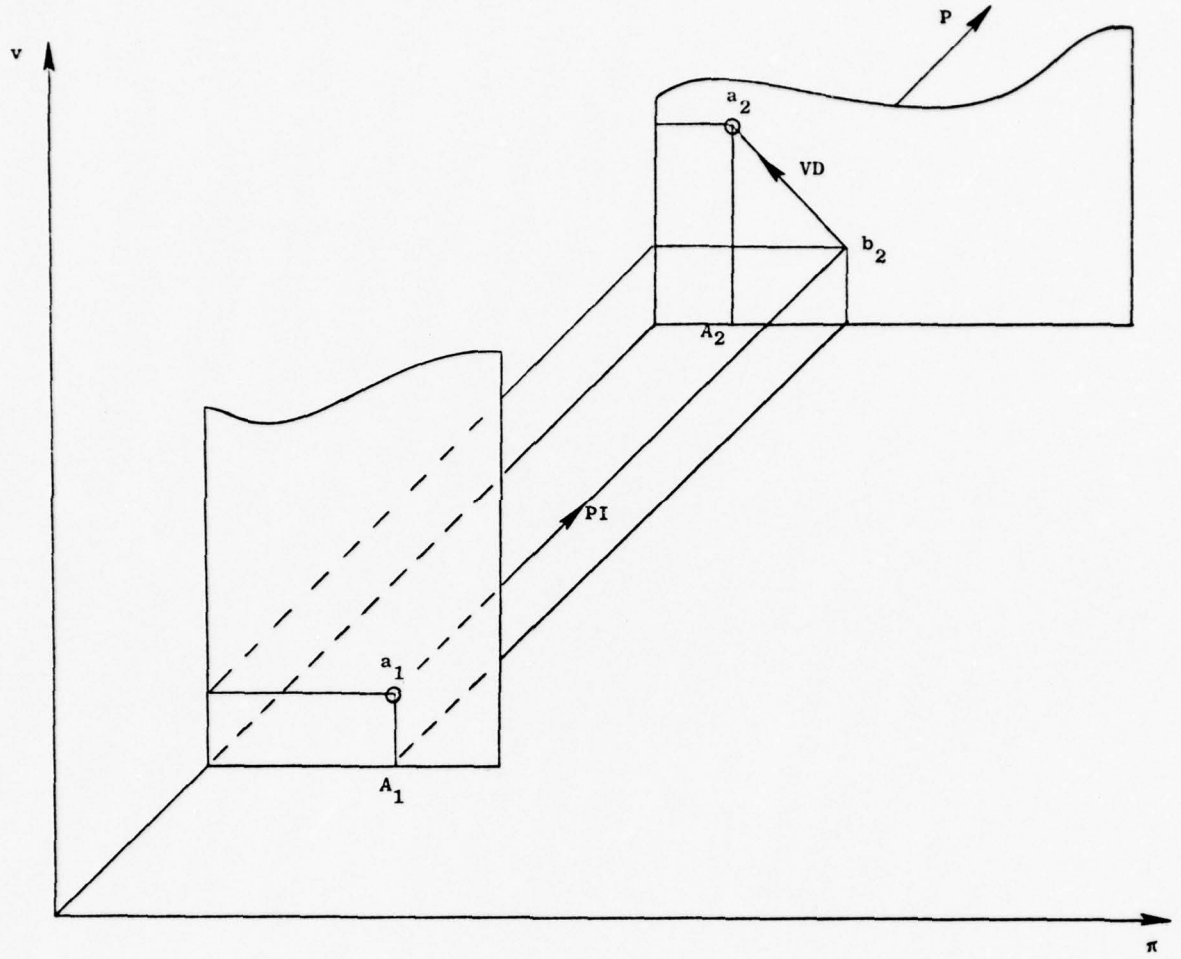


Fig. 2.1. A PI-VD ITERATION.

higher gain than the current one. Firstly, the fact that L is maximized along the "ray" emanating from a point a_i does not guarantee that the resultant a_j will give the highest gain possible on this iteration (i.e., the best improvement).

Referring to Figure 2.2, the PI surveys the points b_2 through b_5 which lie on the "ray" emanating from a_1. It then selects that point b_i at which L is maximum. For Figure 2.1, b_2 happens to be that point. The VD then gives a_2. However, it cannot be proved that another point b_3, say, at which L is less than at b_2, will necessarily yield a point a_3 for which the gain is less than a_2. In other words, the ordering of the values of L at b_i is not necessarily identical to the ordering of the gain values at the corresponding a_i. Moreover, merely increasing L along the a_i "ray" does not guarantee a better policy. That guarantee is to be provided outside the Lagrangian framework, as we shall explain later. The question might arise, therefore, of whether it is at all appropriate to maximize L for policy improvement. The appropriateness of this procedure is justified for two reasons. First, we know that at the optimum the value of the constrained function we are trying to maximize is identical to that of L, as well as at all points a_i. Second, since we are trying to maximize our original function, it would pay to try increasing L as we go along. This gives us an insight into how to go about policy improvement when there are policy constraints, as long as we bear in mind two things. The first is that the sole purpose of PI is to obtain a policy having a higher gain, irrespective of how much higher it is (e.g., it might happen that applying (2.57) to only one state yields a policy P_1 whose gain is higher than the policy P_2 obtained by applying (2.57) to all states.

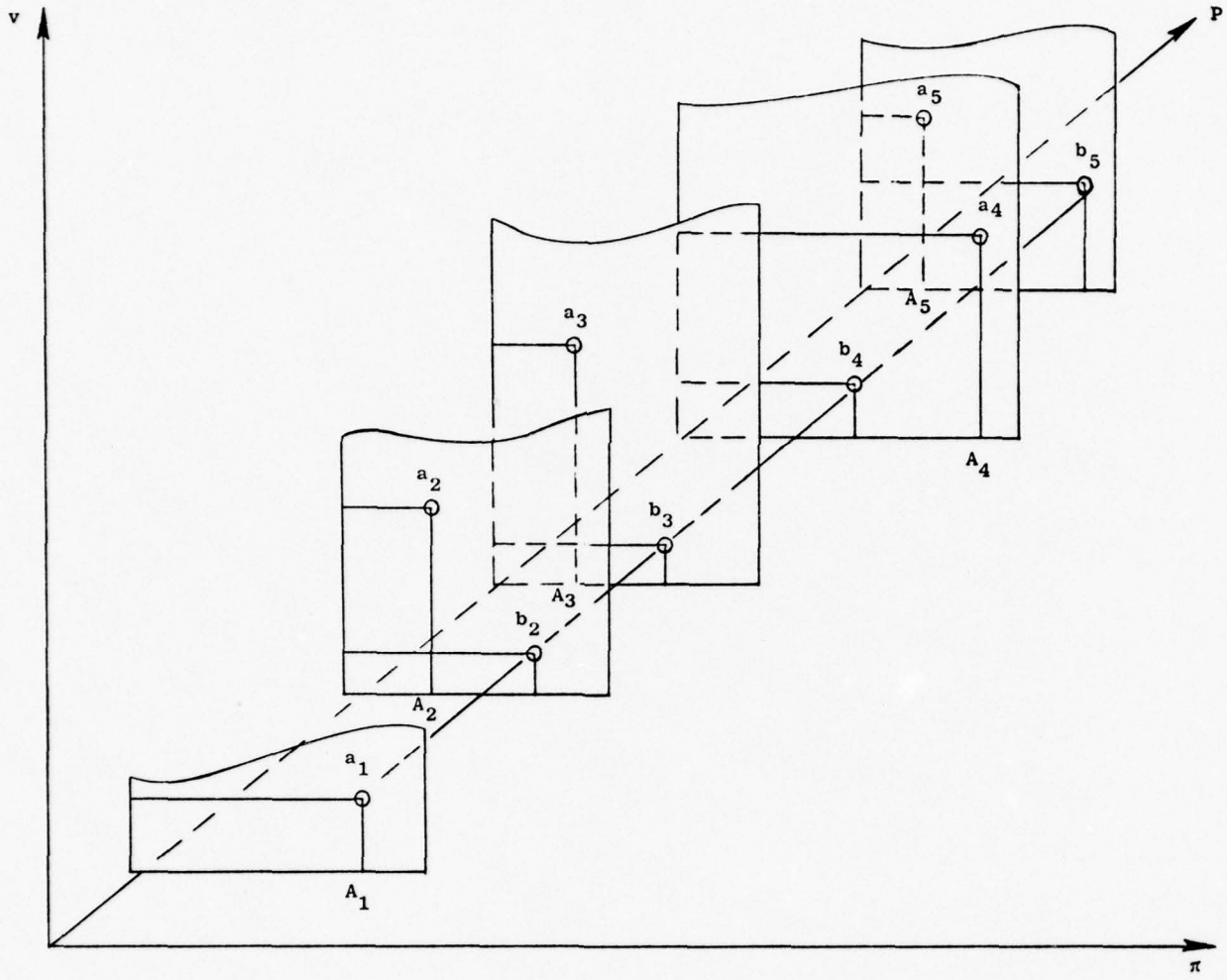


Fig. 2.2. THE POLICY IMPROVEMENT ROUTINE.

The computational cost of trying to detect this is prohibitive. Consequently, we apply (2.57) to at least one state, the result being a policy having a higher gain than the current one, i.e., merely an improvement (not necessarily the best improvement). The second thing to bear in mind is that merely increasing L from a_i to b_j, say, does not guarantee that a_j is a better policy. It might very well happen that L increases from a_i to b_j and then decreases from b_j to a_j (where its value is g) such that $L(a_j) < L(a_i)$.

Now we reconsider (2.56), namely,

$$\sum_{i=1}^{N} \pi_i \sum_{k=1}^{K_i} \left(q_i^k + \sum_{j=1}^{N} p_{ij}^k v_j \right) d_i^k \tag{2.56}$$

For unconstrained policies, it reduces to (2.57), which gave us a policy P^*. If the policy constraints do not make P^* infeasible, then we select P^*. Otherwise, we have to solve the combinatorial problem of selecting that vector T(P) from the feasible set of such vectors (corresponding to the feasible set of policies), which maximizes (2.56) and guarantees an increase in the gain. If we rewrite (2.56) as

$$\sum_{i=1}^{N} \pi_i \sum_{k=1}^{K_i} \left(q_i^k + \sum_{j=1}^{N} p_{ij}^k v_j \right) d_i^k = \sum_{i=1}^{N} \sum_{k=1}^{K_i} \pi_i \left(q_i^k + \sum_{j=1}^{N} p_{ij}^k v_j \right) d_i^k$$

and define

$$t_i^k = q_i^k + \sum_{j=1}^{N} p_{ij}^k v_j$$

we find that we are trying to maximize

$$\sum_{i=1}^{N} \sum_{k=1}^{K_i} \pi_i t_i^k d_i^k \tag{2.58}$$

Since d_i^k is zero for all k's but one in any state i, and the t_i^k are the PI test quantities of yesteryear, (2.58) tells us that we are trying to maximize the sum of those test quantities, one in each state, weighted by how much time the system spends, on the average, in each state. This makes intuitive sense. Moreover, maximizing the individual components of a sum, automatically maximizes the whole sum. In the absence of policy constraints, it is possible, as we have shown, to maximize the individual components. We do not have to worry about feasibility. In this case, the π_i become mere scaling factors common to the components we are trying to maximize, whence they can be ignored. Once we introduce constraints on the policies, however, we have to take feasibility into consideration. The states become "coupled" through the constraints, such that it might not be feasible to maximize the individual components independently of one another. It is the "noncoupling" of states which allows the reduction of (2.58) to (2.57). Thus, when policy constraints are present, we have to consider the sum as a whole and seek an efficient method of solving the nonreducible combinatorial problem. Here, we will have to take into consideration the Π of the current policy. This would seem to imply additional computations (solving N simultaneous linear equations) per iteration. However, this is not so. We have shown that the π's are already there as a by-product of VD ((2.52) and (2.55)). Thus, taking the π's into consideration does not involve any extra computational effort. The test quantities are merely

multiplied by the corresponding π's before we embark upon our maximization. That maximization, as we noted earlier, does not, of itself, guarantee an improvement in the gain. However, we do have a sufficient condition (developed by Howard [4]) for a gain improvement from one policy to another.

Assume that we have a policy P for which the VD has been performed (i.e., the v's and π's computed). Consider any other policy P'. Each policy has a vector of test quantities T associated with it, where

$$t_i^k(P) = q_i^k(P) + \sum_{j=1}^{N} p_{ij}^k(P)\, v_j(P) \qquad i = 1,2, \ldots, N \qquad (2.59)$$

$$t_i^k(P') = q_i^k(P') + \sum_{j=1}^{N} p_{ij}^k(P')\, v_j(P) \qquad i = 1,2, \ldots, N \qquad (2.60)$$

where the v_j's are those obtained from the VD for policy P. Specifically,

$$q_i^k(P) + \sum_{j=1}^{N} p_{ij}^k(P)\, v_j(P) = v_i(P) + g(P) \qquad i = 1,2, \ldots, N \qquad (2.61)$$

Thus, (2.59) reduces to

$$t_i^k(P) = v_i(P) + g(P) \qquad (2.62)$$

Had we solved the VD for policy P', we would have had

$$q_i^k(P') + \sum_{j=1}^{N} p_{ij}^k(P')\, v_j(P') = v_i(P') + g(P') \qquad i = 1,2,\ldots,N \qquad (2.63)$$

Combining (2.63) and (2.60), we can immediately write

$$t_i^k(P') = v_i(P') + g(P') + \sum_{j=1}^{N} p_{ij}^k(P')\left[v_j(P) - v_j(P')\right]$$

$$i = 1,2, \ldots, N \qquad (2.64)$$

Now compute the components γ_i of the difference vector $\gamma = T(P') - T(P)$ from (2.64) and (2.62):

$$\gamma_i^k = t_i^k(P') - t_i^k(P)$$

$$= v_i(P') + g(P') + \sum_{i=1}^{N} p_{ij}^k(P')\left[v_j(P) - v_j(P')\right] - v_i(P) - g(P)$$

Setting

$$\Delta v_i = v_i(P') - v_i(P)$$

$$\Delta g = g(P') - g(P)$$

and rearranging terms, we get

$$\Delta g + \Delta v_i = \gamma_i + \sum_{j=1}^{N} p_{ij}^k(P')\,\Delta v_j \qquad (2.65)$$

But (2.65) has exactly the same form as (2.51) (the VD equations) where the correspondence is

$$g \longleftrightarrow \Delta g, \qquad v \longleftrightarrow \Delta v, \qquad \gamma \longleftrightarrow q$$

We previously showed that the solution for g was $\Sigma\, \pi_i q_i$, where the π's are the limiting state probabilities of the policy under

consideration. Therefore, we can immediately write the solution of (2.65) as

$$g(P') - g(P) = \Delta g = \sum_{i=1}^{N} \pi_i(P')\, \gamma_i \qquad (2.66)$$

Since the π's are nonnegative, (2.66) states that the sufficient condition for improving the gain (from P to P') is that γ_i be nonnegative for all states and strictly positive in at least one state. Note that the definition of the test quantity differences γ_i involves quantities known for policy P. In other words, we do not have to solve the VD for every alternate policy P'. Without knowing $\Pi(P')$, we can guarantee that P' is better than P if at least one γ_i is positive. We refer to this as an "improvement" in state i. Note, however, that the derived condition is not necessary. There might very well be another policy P" which gives positive and negative γ_i's in different states but has a Π vector which makes the R.H.S. of (2.66) positive. We can not discover this, however, without solving the VD for P". Thus, the best procedure for guaranteeing a gain improvement is to improve the test quantity in at least one state. In the absence of policy constraints, (2.57) does that. In the presence of policy constraints, we have to maximize (2.58), subject to improving at least one state. As mentioned earlier, that maximization is a combinatorial problem. Solving it, constitutes a modification of PI to handle constrained policies, giving us the algorithm we are seeking. This we do in the next section.

D. Development and Convergence of the Algorithm

As explained earlier, the algorithm consists of the VD intact, plus a modified PI to handle policy constraints. First, we address the combinatorial problem:

$$\max_{P \in F} \sum_{i=1}^{N} \sum_{k=1}^{K_i} \pi_i t_i^k d_i^k \tag{2.58}$$

subject to

$$\left.\begin{aligned} C_\ell\left(d_i^k\right) \leq 0 \qquad & \ell = 1,2, \ldots, m \qquad (i,k) \in S_1 \\ C_p\left(d_i^k\right) = 0 \qquad & p = 1,2, \ldots, q \qquad (i,k) \in S_2 \end{aligned}\right\} \tag{2.46}$$

where all the inequality type of constraints have been grouped together, as are those of the equality type.

F is the set of feasible policies defined by (2.46), where a policy, by definition, means selecting one alternative in each state, i.e., satisfying the constraints (2.37).

One of the most efficient techniques for solving combinatorial problems is the branch and bound method (or the multiple choice programming) [2,3,7]. Here, the space we are optimizing over is divided into subsets in such a manner that only a portion of the whole space is examined. We will develop a method based on these concepts. Central to this method are the concepts of branching and fathoming, which we proceed to define. Branching is the process of obtaining one or more points from a given infeasible point in the policy space. Fathoming is a property of a point being considered. If no branching can be made from a point, or if no benefit is going to result from such branching, the point is said to be

fathomed. Thus, fathoming is basically the termination of branching. An unfathomed point is eligible for branching.

We will illustrate the method of branching and bounding by considering an example. Assume that we have three states, with three alternatives to choose between in each state. Assume, furthermore, that the VD has been carried out for a given feasible policy, resulting in the test quantities t_i^k, defined by (2.59). We will consider the t_i^k multiplied by the corresponding π_i's and list them as in Table 2.2, where they are in descending order in each state (e.g., $t_2^1 > t_2^3 > t_2^2$, i.e., alternative 1 in state 2 maximizes the test quantity).

Table 2.2

TABLE OF ORDERED TEST QUANTITIES

State	Ordered Test Quantities
1	t_1^2 t_1^3 t_1^1
2	t_2^1 t_2^3 t_2^2
3	t_3^2 t_3^1 t_3^3

Table 2.2 also illustrates the policy constraints. Here, we assume constraints of the simple mutually exlusive type. Corresponding to the "couplings" in Table 2.2, we have the constraints:

$$d_1^2 + d_2^1 \leq 1 \tag{2.67}$$

$$d_1^1 + d_2^3 \leq 1 \tag{2.68}$$

$$d_2^3 + d_3^2 \leq 1 \tag{2.69}$$

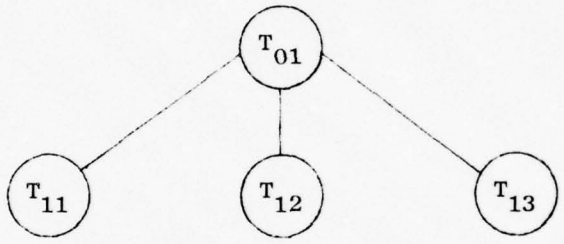
$$d_2^3 + d_3^1 \leq 1 \tag{2.70}$$

Now we start "branching and bounding" to maximize (2.58), subject to (2.67) through (2.70) (the equivalent of (2.46)). Since a point "branches" into other points, we will have a "tree." We will also have a lower bound for the optimum. (The lower bound is initialized to the artificial value of $-\infty$.) Whenever branching gives us a feasible point, the value of L (the function we are trying to maximize) is compared to the current lower bound. If it exceeds that bound, the bound is updated, and any point yielding a value of L which is lower than the new bound is fathomed. Feasible points are fathomed by definition. The search terminates when no more unfathomed points exists. We start out with the point representing that T vector whose components are given by (2.57), i.e., the largest test quantity in each state. Denote it by T_{01}. Our tree then initially consists of one node

where $T_{01} = (t_1^2, t_2^1, t_3^2)$.

The corresponding policy is $P_{01} = (2,1,2)$, i.e., selecting alternatives 2, 1, and 2 in states 1, 2, and 3, respectively. This is the policy the PI would select. However, in our example, it is infeasible (it violates (2.67)). Hence, our lower bound remains at its initial value of $-\infty$, and we have one unfathomed point T_{01} to branch from. Branching consists of selecting each alternative in state 1, in turn,

i.e., changing the first component of T_{01}. The remainder of the components are chosen such that they are the largest test quantities in their states, consistent with the constraint imposed by the first component, if any. Thus, selecting t_1^2, e.g., would prohibit selecting t_2^1, and hence we have to select t_2^3 instead. For the third component, we can select t_3^2 because it is not "coupled" with t_1^2. The fact that t_2^3 and t_3^2 are coupled is postponed to the next level of branching, if we get there. Hence, our tree becomes



where

$$T_{11} = \left(t_1^2, t_2^3, t_3^2\right)$$

$$T_{12} = \left(t_1^3, t_2^1, t_3^2\right)$$

$$T_{13} = \left(t_1^1, t_2^1, t_3^2\right)$$

The corresponding policies are:

$$P_{11} = (2,3,2)$$

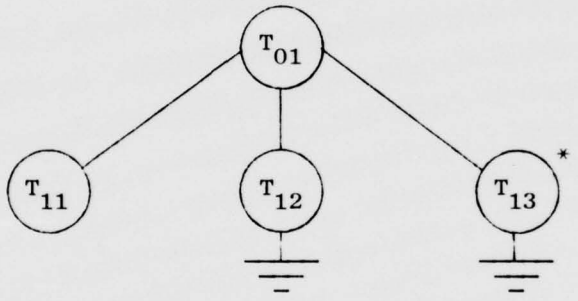
$$P_{12} = (3,1,2)$$

$$P_{13} = (1,1,2)$$

Only P_{11} is infeasible, whence T_{12} and T_{13} are immediately fathomed. Moreover, the value of the Lagrangian at those two points is compared to the lower bound $(-\infty)$ and to that at T_{11}. Assume that $L(T_{11}) > L(T_{13}) > L(T_{12})$. In this case, the lower bound is updated to

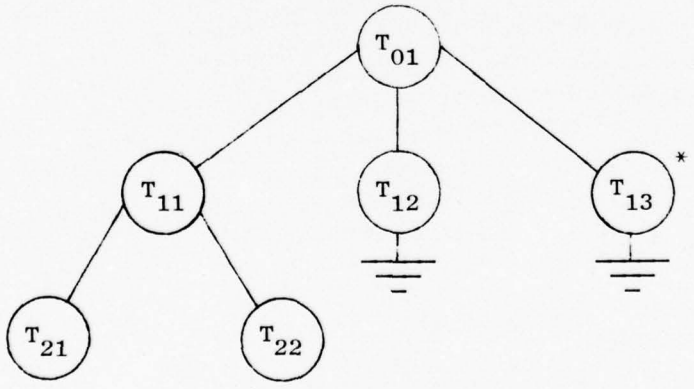
$$L = L(T_{13})$$

and T_{11} is the only point available for branching. If we represent fathoming by "grounding" the point in the tree, the situation becomes:



Moreover, P_{13} is the optimum policy so far. Now we start branching from T_{11}, as we did from T_{01}. Here, we select, in state 2, each alternative in turn. However, the alternative must be uncoupled from t_1^2. In other words, at each level in the tree, we have a fixed alternative in a number of states. T_{01} represents level 0. No states have fixed alternatives. The next level of T_{11}, T_{12}, and T_{13} has the alternative in state 1 fixed (this is level 1). Only T_{11} goes down one further level (to level 2) to attempt fixing alternatives in state 2, consistent with the constraints imposed by the alternative fixed in the previous level. Since T_{11} has t_1^2 fixed, and t_1^2 is coupled with t_2^1, we cannot fix the latter. Thus, we only have two successor points to T_{11}. The remaining components are selected such that they

are the maximum test quantities in their states, under the restriction that they satisfy any constraints imposed by fixing the previous levels. Since we are down to the last level, the points we obtain, if any, have to be feasible. Thus, fixing t_2^3 imposes selecting t_3^3, while fixing t_2^2 enables us to select t_3^2. Note that, if t_2^3 were coupled with t_3^3 also, we would only have one successor point to T_{11}. If, in addition, t_2^2 were coupled with all alternatives in state 3, no branching would be possible from T_{11}, and it would be fathomed. Those are interesting cases because such constraints imply that t_2^3 and t_2^2 are not really alternatives at all; they can never be selected in a feasible policy. This can be detected by manipulating the constraints to discover that they impose $d_2^3 = d_2^2 = 0$. However, we are more inclined to let the branch and bound discover this (along with nonfeasible problems). Thus, at this step, our tree would become:



where

$$T_{21} = \left(t_1^2, t_2^3, t_3^3\right)$$

$$T_{22} = \left(t_1^2, t_2^2, t_3^2\right)$$

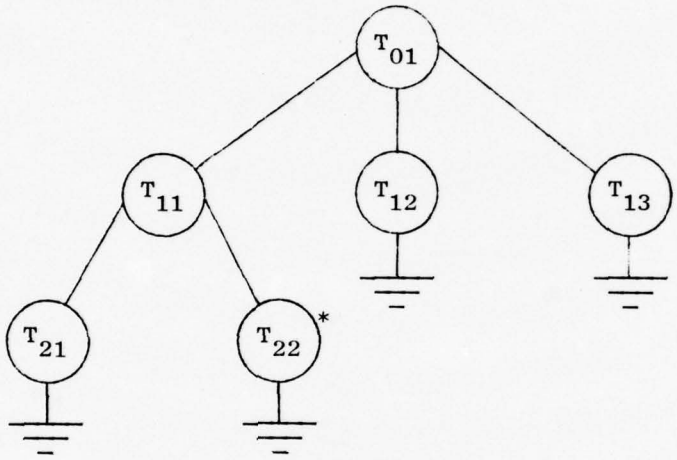
$$P_{21} = (2,3,3)$$

$$P_{22} = (2,2,2)$$

The * next to T_{13} implies that this is the best point obtained so far.

Both P_{21} and P_{22} are feasible, whence we compute $L(T_{21})$ and $L(T_{22})$ and compare them to the current bound (the best value of L so far). Also, both points are fathomed. Assume that $L(T_{22}) > L(T_{21}) > L$.

In this case, we update L, set P_{22} as our optimum so far, and survey the tree for unfathomed points:



Since no more branching is possible, T_{22} is the optimum. If we reach the end without encountering any feasible points, the problem is unfeasible. This is detected by the lower bound still being at its original value of $-\infty$.

Note that in this example there are 27 different policies. Of those, only 17 are feasible, and we only considered 6. The efficiency of branch and bound techniques (BB) results from the manner in which branching and fathoming are implemented. The branching takes feasibility into account in a piecemeal fashion, one state at a time, while being always biased towards sets of points where L has larger values. This gives us a chance to seize upon a feasible policy of large L relatively quickly.

Then, the bounding eliminates, via fathoming, whole sets of points from any further consideration.

Now we embark upon proving that the outlined BB method maximizes the Lagrangian over the set of feasible policies.

First, we prove that, as we move into deeper levels in the tree, the value of L cannot increase.

Proposition 2.5.

If branching occurs from some infeasible node I at level k in the tree, the value of L at the resultant nodes cannot exceed that at I.

Proof.

The value of L at any node corresponding to a policy P is merely the sum of the components of the vector $T(P)$ corresponding to that policy.

Now consider $T(I)$ at the infeasible node I at level k. Its first k components do not violate any constraints. Moreover, starting from the $k+1$ component, each component is maximum in its state, subject to the constraints imposed by the first k components.

Now any node resulting from I has a T which agrees with $T(I)$ in the first k component. The $k+1$ component is any one in state $k+1$, subject to the constraints imposed by the first k components. Hence, it cannot exceed the $k+1$ component of $T(I)$. Starting from component $k+2$, each component of a branch is the maximum in its state, subject to the constraints imposed by the first $k+1$ components. They might be the same as, or more than, the constraints imposed

by the first k components, but not less. Hence, from component $k+2$, no component of a branch node can exceed the corresponding component of $T(I)$. Thus, we have shown that, starting from component $k+1$, no component of a branch node can exceed the corresponding one in $T(I)$. This proves the proposition.

Proposition 2.6.

If a given feasible policy P is not in the BB tree and there exists a fathomed policy P' in the tree at level k such that P' and P agree in the first k components, then the value of L at P cannot exceed the optimum value obtained by the BB.

Proof.

We have two cases to consider:

(a) P' is feasible. Since P' agrees with P in the first k components and the rest of the components of $T(P')$ are the maximum in their states, subject to the constraints imposed by the first k components, then no component of $T(P)$ can exceed the corresponding component of $T(P')$.

(b) P' is infeasible. Assume that P' is fathomed because no branching is possible from it. This means that the first k components impose constraints which make all alternatives in state $k+1$ infeasible. But P agrees with P' in the first k components, whence they impose the same constraints. Hence, the $k+1$ component of P violates some constraint, i.e., P is infeasible. But this contradicts the assumptions. Hence, P' was fathomed because there exists some node F elsewhere in the tree yielding a larger value for L than node P'. Since

P is obtainable from P' by branching, Proposition 2.5 says that L at P cannot exceed that at P', whence it cannot exceed that at F.

Thus, in (a) and (b), we have shown that there exists a feasible policy in the tree where the value of L is not less than that at P. But since the optimum obtained by BB is the largest value for L over all feasible nodes in the whole tree, the proposition is proved.

Proposition 2.7.

If a given feasible policy P is not in the BB tree and there exists an unfathomed policy I in the tree at level k, such that P and I agree in the first k components, then there exists a policy P' in the tree at level $k+1$ such that P and P' agree in the first $k+1$ components.

Proof.

Since I is not fathomed, branching has occurred from it. Consider the branch nodes. They all agree with I, whence with P, in the first k components. Component $k+1$ takes on all values in state $k+1$ such that the constraints imposed by the first k components are not violated. But component $k+1$ of P satisfies the same constraints (because it satisfies all constraints). Thus, one of the branch nodes from I agrees with P in component $k+1$, whence it agrees with it in the first $k+1$ components. Since this node is at level $k+1$, the proposition is proved.

Proposition 2.8.

If a given feasible policy P is not in the BB tree and there exists a node N in the tree at level k such that N and P agree in the first k components, then the value of L at P cannot exceed the optimum obtained by BB.

Proof.

Consider the node N. It is at level k and agrees with P in the first k components. If N is fathomed, apply Proposition 2.6.

If N is not fathomed, apply Proposition 2.7 repeatedly. Every time we get to an unfathomed node at level ℓ, there is a node branching from it at level $\ell+1$, agreeing with P in $\ell+1$ components. Finally, we reach a fathomed node at some level $m \leq M$ (where M is the deepest level the tree reaches) and apply Proposition 2.6.

Proposition 2.9.

No feasible policy P can yield a value of L greater than the optimum obtained by BB.

Proof.

If P is in the tree, the proof is trivial. Consider a feasible policy P not in the tree. The first component in P is an alternative in state 1. Now look at level 1 in the tree. It has as many nodes as state 1 has alternatives. Each node has one alternative in state 1 as its first component. Thus, there exists a node N in the tree at level 1 such that N and P agree in the first component. We have satisfied the assumptions of Proposition 2.8, whence its result applies, completing the proof.

Now that we have a method for solving the combinatorial problem of maximizing the Lagrangian over the set of feasible policies, we have to guarantee an improvement in the gain. This we do by guaranteeing that the quantity defined by (2.66) never be negative. To ensure this, we do not allow negative γ_i's in each iteration. To illustrate, we first repeat Table 2.2.

State	Ordered Test Quantities
1	t_1^2 t_1^3 t_1^1
2	t_2^1 t_2^3 t_2^2
3	t_3^2 t_3^1 t_3^3

Now we assume that we entered with policy (3.3.3). We want to disallow any test quantity that is less than that of the current policy in each state (whence no γ_i can be negative). In our case, the policies we consider in BB in this iteration would be given by the following table.

State	Ordered Test Quantities
1	t_1^2 t_1^3
2	t_2^1 t_2^3
3	t_3^2 t_3^1 t_3^3

No resulting policy can have any negative γ_i, whence (2.66) can never be negative, i.e., we prevent selection of a policy with lower gain. Now assume that BB yields policy (2,3,3). Assume, furthermore, that the VD for that policy results in the following table.

State	Ordered Test Quantities
1	t_1^1 t_1^3 t_1^2
2	t_2^3 t_2^1 t_2^2
3	t_3^3 t_3^1 t_3^2

(Note that the "coupling" is between alternatives in different states, not between test quantities. That is why the "couplings" look different in this table.) Now we enter BB with policy (2,3,3), and, to restrict the γ_i, we only consider the following.

State	Ordered Test Quantities
1	t_1^1 t_1^3 t_1^2
2	t_2^3
3	t_3^3

It is obvious that discarding alternatives reduces the amount of computations involves in each iteration. To increase the efficiency, we can renumber the states in each iteration such that state 1 has the least number of alternatives. The obvious question is what if the BB results in the same policy we entered with? What is the characteristic of such a policy? In the following theorem, we show that it maximizes the gain over a subset of the feasible policies.

Theorem 2.3.

If the maximization of the Lagrangian over the set of feasible policies, subject to $\gamma_i \geq 0$ for all i, yields a policy P for which $\gamma_i = 0$ for all i, then that policy maximizes the gain over the set of all feasible policies that differ with it in eactly one state.

Proof.

Let P' be a feasible policy differing with P in the k^{th} componant (i.e., state). Let $P(k) = \ell$ and $P'(k) = m$. Assume that $t_k^m > t_k^\ell$. Since $T(P)$ differs from $T(P')$ only in the k^{th} component, the Lagrangian at P' is greater than at P. This implies that there exists a feasible policy P' yielding a value of L greater than the optimum obtained by BB. But this contradicts Proposition 2.9.

Hence,

$$t_k^m \leq t_k^\ell$$

Thus,

$$\gamma_k \leq 0 \quad \text{and} \quad \gamma_i = 0 \quad \text{for} \quad i \neq k$$

where

$$\gamma = T(P') - T(P)$$

Hence,

$$g(P') - g(P) = \sum_{i=1}^{N} \pi_i(P')\, \gamma_i$$

$$= \pi_k \gamma_k \leq 0$$

i.e.,

$$g(P') \leq g(P)$$

Thus, any feasible policy differing with P in exactly one state cannot have a higher gain than P.

Thus, when BB converges to a policy, that does not necessarily mean that it is the overall optimum. What we propose to do in this case is to make that policy, and all policies differing with it in exactly one

state, infeasible, thus removing a whole subset from further consideration. This can be achieved by adding just one constraint to the set of policy constraints. If the policy P is given by $P(i) = k$, the constraint is

$$\sum_{i=1}^{N} d_i^k \leq N - 2 \tag{2.71}$$

(As a matter of fact, (2.71) is a special case of a general form. For example, to make an individual policy infeasible, the R.H.S. would be $N-1$. In general, to make a policy, and all those differing with it in exactly $M < N$ states infeasible, the R.H.S. of (2.71) would be $N - M - 1$.)

In order to increase computational efficiency, we divide the states into two types. The "free states" are those which are not involved in any policy constraints, i.e., their d_i^k's are only involved in relations of the type (2.37). The states involved in relations of the type (2.46), i.e., having alternatives that are "coupled" with each other, we refer to as "coupled states." Given a feasible policy A for which VD has been performed, we first attempt a regular PI (maximizing test quantities over all states). If this does not change A, we have an overall optimum policy. If the resultant policy is feasible, we start a new VD. Otherwise, we maximize over the free states by regular PI, and over coupled states by BB with $\gamma_i \geq 0$. This results in a policy B. If B differs from A, we enter VD. Otherwise, we know that A maximizes the gain over the set of feasible policies differing with A in exactly one coupled state. In this case, we make policy A, as well as all policies differing with it in exactly one coupled state,

infeasible by a (2.71) type constraint (here, the N in (2.71) would be the number of coupled states). Then, we enter BB again to maximize the Lagrangian over the currently feasible policy set, with the $\gamma_i \geq 0$ restriction removed. Basically, we are looking for a feasible policy, rerespective of gain, whence it makes sense to use the latest values of t_i^k since they contain a certain amount of the algorithm's history up to this point.

Every time BB converges to a subset maximizer, we compare its gain to the best previous subset maximizer and retain the one with the higher gain. As a result of removing subsets over which we maximize, the feasible policy set quickly shrinks until it becomes empty. When BB results in an infeasible problem, we have an optimum policy.

To get an initial feasible policy, we maximize the sum of the immediate expected rewards q_i^k over the feasible policy set using BB. Schematically, then, our algorithm can be represented in Figure 2.3. The convergence of this algorithm to an optimum feasible policy is readily proved.

Consider a feasible problem (nonfeasible ones are detected at the outset by exiting from E1). Now consider any feasible policy P other than the one selected by the algorithm.

If we exited the algorithm from E2, then policy A has the highest gain of any policy (whether feasible or not). This is because, for A, each test quantity is the maximum in its state. Any policy which is not identical with A has to be different from it in at least one state. Consequently, any policy other than A results in nonnegative γ_i's, with at least one γ_i strictly negative. Equation (2.66) then implies $g(P) < g(A)$. This is the case where the policy constraints do not affect

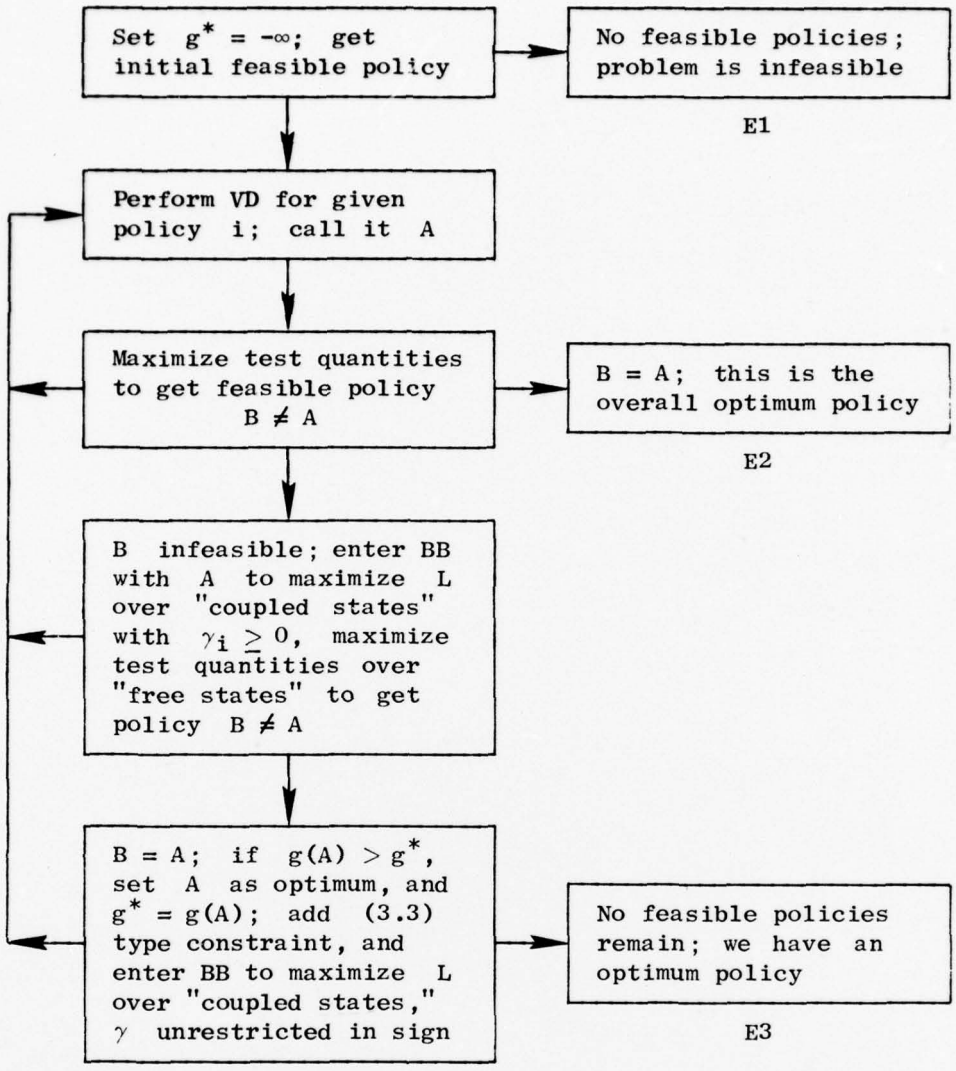


Fig. 2.3. ALGORITHM FOR RISK-INDIFFERENT CASE.

the feasibility of that policy yielding the highest gain in the absence of any such constraints. Thus, we have retained the ability to detect such a policy, without having to exhaust the feasible policy set, by (2.71) type constraints.

If we exited the algorithm from E3, then, at that point, the given feasible policy P had become infeasible. The only way this can come

about is from (2.71) type constraints. Hence, P belongs to some subset over which we have maximized the gain, whence g(P) cannot exceed the gain of the policy selected by the algorithm.

Since no feasible policy can have a gain higher than that of the policy selected by the algorithm, the latter is the optimum feasible policy.

E. The Example

We will take Howard's famous taxicab example [4] and add some policy constraints to it. The taxi-cab driver works in an area encompassing three towns A, B, and C. In towns A and C, he has three alternatives.

1. He can cruise in the hope of being hailed by a passenger.
2. He can drive to the nearest cab stand and wait in line.
3. He can pull over and wait for a radio call.

If he is in town B, alternative 3 is not available because there is no radio cab service in that town. For a given town and alternative, there is a probability that the next trip will be to each of the towns A, B, and C, and a corresponding net monetary reward associated with each such trip. If towns A, B, and C are identified with states 1, 2, and 3, then Table 2.3 gives the probabilities and rewards. Now we introduce the constraints. In towns A and B, there is a union for taxi-cab drivers. The union owns the cab stands in both towns and the radio cab service in town A. Nonmembers are denied the use of union facilities. Union membership, however, has strings attached to it. To become a member, a driver has to use the facilities, except that he can only use one cab stand (either

Table 2.3

PROBABILITIES AND REWARDS FOR THE TAXICAB EXAMPLE

State	Alternative	p_{ij}^k	r_{ij}^k	Rewards
i	k	j = 1 2 3	j = 1 2 3	$q_i^k = \sum_{j=1}^{N} p_{ij}^k r_{ij}^k$
1	1	1/2 1/4 1/4	10 4 8	8
	2	1/16 3/4 3/16	8 2 4	2.75
	3	1/4 1/8 5/8	4 6 4	4.25
2	1	1/2 0 1/2	14 0 18	16
	2	1/16 7/8 1/16	8 16 8	15
3	1	1/4 1/4 1/2	10 2 8	7
	2	1/8 3/4 1/8	6 4 2	4
	3	3/4 1/16 3/16	4 0 8	4.5

in town A or town B to give other members a chance). Thus, if our friend joins the union, alternative 1 (cruising) is not available for him in states 1 and 2, whereas if he does not, alternative 1 becomes the only available one in both states. In other words, either $d_1^1 = d_2^2 = 0$ or $d_1^1 = d_2^1 = 1$. This is a type (2.43) constraint. Specifically,

$$d_1^1 - d_2^1 = 0 \tag{2.72}$$

Also, since he cannot select alternative 2 in both states 1 and 2, no matter what, we have a (2.41) type constraint. Specifically,

$$d_1^2 + d_2^2 \leq 1 \tag{2.73}$$

Our friend wants to select a policy that yields the highest gain, subject to those constraints.

The first step in the algorithm is to set the optimum gain $g^* = -\infty$ and get an initial feasible policy. We use BB to maximize the sum of immediate expected rewards over the feasible set. Our first node in the tree is the one that picks the maximum q_i^k in each state:

$$T_{01} = (8,16,7) \qquad P_{01} = (1,1,1)$$

P_{01} is feasible, whence T_{01} is fathomed, and we have an initial feasible policy $A = (1,1,1)$.

Performing the VD for A gives:

State	Ordered Test Quantities		
1	$t_1^1 = 4.213$	$t_1^2 = 3.373$	$t_1^3 = 2.207$
2	$t_2^2 = 4.323$	$t_2^1 = 3.333$	
3	$t_3^2 = 3.907$	$t_3^1 = 3.680$	$t_3^3 = 2.387$

and $g(A) = 9.200$

Constraint (2.72) is represented by a double line coupling the two alternatives involved, while (2.73) is represented by a single line. Since the alternatives in state 3 are not coupled to any other alternatives, state 3 comprises the set of "free states" we defined earlier. The dotted line separates the free states from the coupled ones.

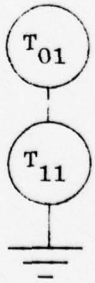
The regular PI gives us policy (1,2,2) which is infeasible; it violates (2.72). Hence, we maximize over the free state to get P(3) = 2. For the coupled states, we enter BB with $\gamma_i \geq 0$, i.e., the following table:

State	Ordered Test Quantities
1	t_1^1
2	t_2^2 $\quad$ t_2^1

The first node is the maximum in each state, an infeasible one:

$$T_{01} = (4.213, 4.323) \qquad P_{01} = (1,2)$$

To branch to the next level, we fix alternatives in state 1. Since we only have one alternative, we only have one branch. Also, $d_1^1 = 1$ and (2.72) make $d_2^1 = 1$, whence we can only select alternative 1 in state 2. This makes the branch node feasible, whence it is immediately fathomed.



$$T_{11} = (4.213, 3.333) \qquad P_{11} = (1,1)$$

Therefore, $P(1) = 1$ and $P(2) = 1$. Thus, we emerge with a policy $B = (1,1,2)$, different from A. So, we call this new policy A, i.e., $A = (1,1,2)$ and perform the VD for A. This results in:

State	Ordered Test Quantities		
1	$t_1^1 = 3.960$	$t_1^2 = 3.148$	$t_1^3 = 2.077$
2	$t_2^2 = 6.720$	$t_2^1 = 5.197$	
3	$t_3^2 = 2.741$	$t_3^1 = 2.620$	$t_3^3 = 1.617$

$A = (1,1,2)$ $\quad g(A) = 9.366$

Maximizing in all states, i.e., regular PI, results in $(1,2,2)$ which is infeasible. It violates (2.72). Thus, we maximize in 3 to get $P(3) = 2$, and we enter BB, for the coupled states only, discarding any alternatives having t_i^k less than the one we entered with (i.e., $\gamma_i \geq 0$). Thus, what we consider is given by:

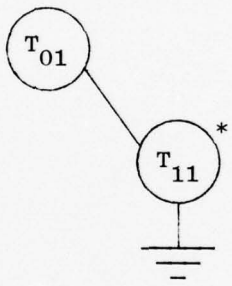
State	Ordered Test Quantities	
1	t_1^1	
2	t_2^2	t_2^1

The third component of our policy will always be 2, so we only write the first 2 for brevity.

Our first node, as usual, is the maximum in each state. This, we already know is infeasible, whence we will branch from T_{01}.

$$T_{01} = (3.960, 6.720) \qquad P_{01} = (1,2)$$

Our first level is obtained by fixing the first component to each available alternative in state 1 and selecting the maximum in state 2, consistent with the constraints imposed by the first component. Here, we only have one alternative available in state 1, namely, alternative 1. Hence, we set $d_1^1 = 1$. This immediately makes $d_2^1 = 1$ by virtue of (2.72). Hence,



$$T_{11} = (3.960, 3.333) \qquad P_{11} = (1,1)$$

P_{11} is feasible, whence T_{11} is immediately fathomed and also gives the largest Lagrangian over the defined set. Hence, BB yields a policy $B = (1,1,2)$. But this is the same as policy A that we entered with.

Hence, A maximizes the gain over the set of all feasible policies that differ with it in exactly one component in the coupled states. This is achieved by adding the constraint:

$$d_1^1 + d_2^1 \leq M_c - 2$$

where M_c is the number of coupled states. Here, $M_c = 2$. Thus,

$$d_1^1 + d_2^1 \leq 0 \qquad (2.74)$$

Note that (2.74) makes $d_1^1 = d_2^1 = 0$, i.e., those alternatives are removed from further consideration. We let the algorithm deal with that, however, rather than scanning every constraint.

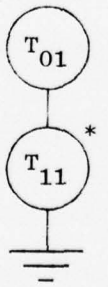
Now we compare $g(A) = 9.366$ to $g^* = -\infty$. $g(A)$ is greater, so we set our optimum policy P_o, so far, as

$$P_o = A = (1,1,2) \qquad g^* = 9.366$$

and look for a feasible solution by maximizing L over the coupled states, with γ_i unrestricted in sign. Thus, the values considered in the BB are given by

State	Ordered Test Quantities
1	t_1^1 t_1^2 t_1^3
2	t_2^2 t_2^1

The BB tree is given:



$$T_{01} = (3.960, 6.720) \qquad P_{01} = (1,2)$$
$$T_{11} = (2.077, 6.720) \qquad P_{11} = (3,2)$$

where P_{11} is feasible.

Hence, we have a new policy A = (3,2,2) for which we perform VD to obtain

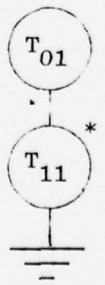
State	Ordered Test Quantities		
1	$t_1^2 = 1.041$	$t_1^1 = 0.579$	$t_1^3 = 0.311$
2	$t_2^2 = 2$	$t_2^1 = 9.086$	
3	$t_3^2 = 1.511$	$t_3^1 = 0.948$	$t_3^3 = -0.182$

A = (3,2,2) g(A) = 12.774

Regular PI gives (2,2,2) an infeasible policy. Hence, we maximize in 3 to get P(3) = 2, and we enter BB for states 1 and 2 with the following table.

State	Ordered Test Quantities		
1	t_1^2	t_1^1	t_1^3
2	t_2^2		

The BB tree for maximizing the Lagrangian is



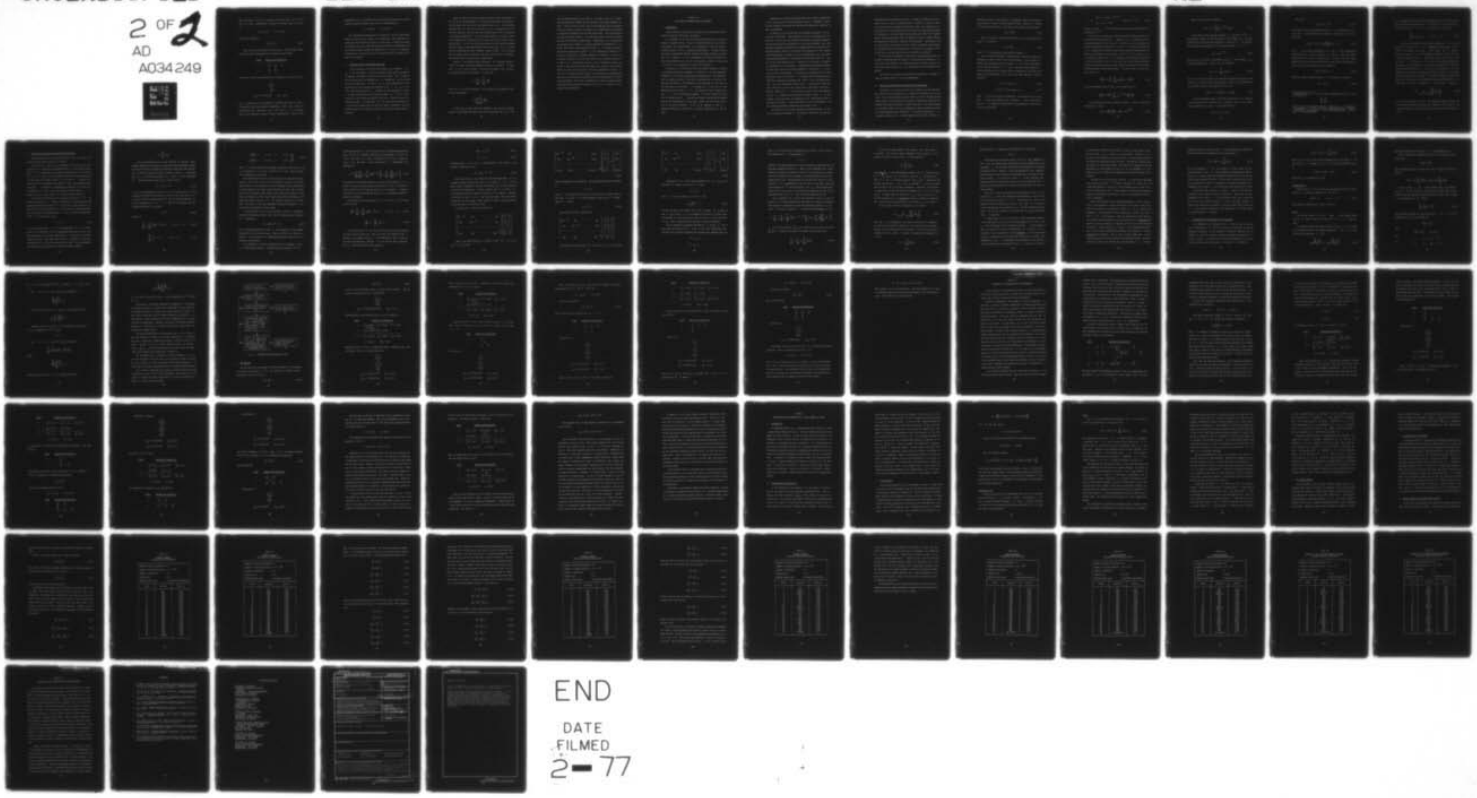
T_{01} = (1.041,20.69) P_{01} = (2,2)

T_{11} = (0.311,20.69) P_{11} = (3,2)

AD-A034 249 STANFORD UNIV CALIF DEPT OF ENGINEERING-ECONOMIC SYSTEMS F/G 12/1
MARKOV DECISION PROCESSES WITH POLICY CONSTRAINTS.(U)
APR 76 J NAFEH NSF-GK-36491

UNCLASSIFIED EES-DA-76-3 NL

2 OF 2
AD
A034249



END

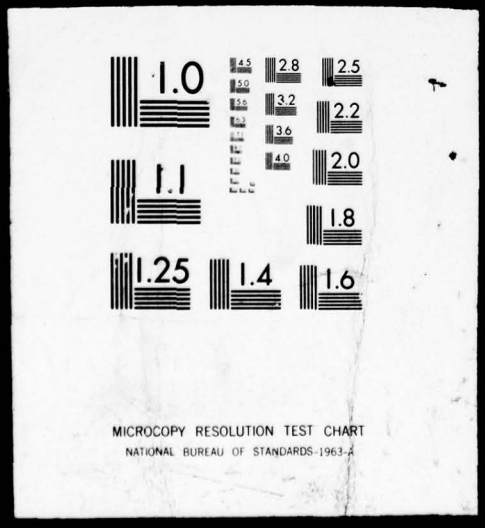
DATE
FILMED
2-77

IFIED

2 OF 3

AD

A034249



Thus, BB gives us (3,2,2), the policy we entered with. $g(A) = 12.774$ and $g^* = 9.366$. Consequently, we update our optimum policy and gain to

$$P_o = (3,2,2) \qquad g^* = 12.774$$

and add the constraint

$$d_1^3 + d_2^2 \leq 0 \tag{2.75}$$

Then we try to get another feasible policy. The following table gives the values considered by BB, followed by the BB tree.

State	Ordered Test Quantities		
1	t_1^2	t_1^1	t_1^3
	‖	‖	
2	t_2^2	t_2^1	

(Of course, we have the additional two constraints (2.74) and (2.75).)

$$T_{01} = (1.041,20.69) \qquad P_{01} = (2,2)$$

P_{01} is infeasible, and no branching is possible from level 0. This is an indication that the problem is infeasible. (This is the effect of (2.74) and (2.75). They force $d_2^1 = 0$ and $d_2^2 = 0$, i.e., no alternatives can be selected in state 2, whence infeasibility.) Thus, we have

exhausted the set of feasible policies (without evaluating each policy, merely by removing subsets), and we have an optimum policy:

$$P_o = (3,2,2) \qquad g^* = 12.774$$

The unconstrained problem had its optimum at (2,2,2). That policy, however, violates (2.73), whence it is infeasible. The introduction of constraints thus affects the policy (in some cases, it might not be true, as we shall later show, when discussing sensitivity to constraints), and the feasible policy yielding the highest gain is the one we obtained. (Incidentally, it turns out to be more beneficial for our friend to become a union member.)

F. Transient States and Periodic Processes

In the foregoing, all states were assumed to be recurrent, i.e., $\pi_i > 0$ for all states. If we have transient states and they happen to be coupled, the theorem of Section C would not apply. This is because the test quantities are multiplied by π_i, whereas the π_i are defined on the test quantities. As long as $\pi_i > 0$, then inequalities of test quantities are not affected by multiplication by π_i. If $\pi_i = 0$, however, then multiplication by π_i equates all test quantities in state i to zero, and that does not necessarily imply γ_i's of zero. Since the proof of the theorem was based on that fact, it does not hold for transient states. In other words, it is not true that BB converges to a policy that maximizes the gain over the subset of feasible policies differing with it in exactly one state if one of the coupled states is transient.

Hence, we need to deal with coupled transient states separately. We do this in the following manner. We fix the alternatives in the recurrent coupled states to those of the current policy (i.e., set the corresponding $d_i^k = 1$). Then we use BB with $\gamma_i \geq 0$ to maximize the sum of the test quantities over the transient coupled states. This is basically a feasibility exploration. If this process results in a change in transient coupled state alternatives (in at least one state), we have an improved policy. If not, we fix the alternatives in the transient coupled states to those of the current policy and use BB with $\gamma_i \geq 0$ to maximize the Lagrangian over the recurrent coupled states. If the policy does not change, then it maximizes the gain over the set of policies that differ with it in exactly one coupled state.

Finally, a word about periodic processes. By a periodic process, we mean one whose transition probability matrix is periodic. In this case, $\pi_i = 1/N$ for all states. The case of interest is one in which all feasible policies are periodic. We know that

$$g = \sum_{i=1}^{N} \pi_i \sum_{k=1}^{K_i} q_i^k d_i^k$$

and it is g we want to maximize. If all policies are periodic, then we want to maximize

$$\frac{1}{N} \sum_{i=1}^{N} \sum_{k=1}^{K_i} q_i^k d_i^k$$

In this case, we only need one iteration. Our initial feasible policy is the optimum one since we use BB to maximize the sum of q_i^k

over the feasible policy set to get it. Of course, this is an inherently deterministic problem (we just maximize the average immediate reward). Unless we know a priori that all feasible policies are periodic, we do not seek to find that out (since it is computationally equivalent to explicitly enumerating the feasible policy set). The only reason for discussing periodic policies, is to shed some light on initial feasible policies and transient coupled states. Maximizing a sum is equivalent to maximizing the average if a uniform probability distribution is assumed. The latter is characteristic of the π_i of periodic policies. The uniform distribution, however, is the mathematical encoding of a Bayseian's profession of complete ignorance of a process. In the case of periodic policies, we do not know where we will find the process if we enter it at a random point in time. This is essentially what we are saying at the start of the algorithm when we do not have any feasible policy available (we do not have a transition probability matrix). In the case of transient coupled states, our Lagrangian has degenerated, making all test quantities equivalent and thus obliterating our accumulated knowledge to this point. Consequently, we profess complete ignorance about those states and maximize the sum (i.e., average) of the original test quantities.

Chapter III

RISK-SENSITIVE MARKOV DECISION PROCESSES

A. Introduction

In this chapter, we develop an algorithm for risk-sensitive Markov Decision Processes with policy constraints.

As in Chapter II, we start by reviewing previous work in Section B. Instead of expected values, we deal with utility functions and certain equivalents, the standard method of incorporating a decision maker's attitude towards risk (i.e., uncertain propositions). The policy evaluation-policy improvement (PE-PI) algorithm developed by Howard and Matheson [6] is outlined. It is the counterpart of the VD-PI algorithm for the risk-indifferent case. Just as our algorithm of Chapter II was based on the VD-PI, our algorithm here is based on the PE-PI.

In Section C, we use the fact that the original problem is actually a constrained optimization problem to formulate it in the Lagrangian framework. We show that decomposing the optimization of the Lagrangian into two problems results in the PE-PI algorithm when no policy constraints are present. The dependence of the algorithm on the sign of the risk aversion coefficient γ is brought out. In the case of a risk preferring decision maker ($\gamma < 0$), the problem is one of maximization. For $\gamma > 0$, it is a minimization problem. In PE-PI, making the utilities of opposite sign to γ during the PE phase, transforms the problem into one of maximization for both cases. Hence, the sign of the utilities is not really arbitrary. It has to be opposite to that of γ; otherwise minimization would have to replace maximization in the PI phase.

Introduction of policy constraints makes the PI phase inapplicable, just as was the case for risk-indifferent analysis. Therefore, we have to look elsewhere for solving the essentially combinatorial problem of policy improvement.

In Section C, we also show that the Lagrange multipliers are the utilities of the PE-PI. They represent the cost of violating the constraints; in this case, the constraints defining an eigenvalue problem. The realization that the constraints in the risk-indifferent case (the equations defining the limiting state probabilities) also define an eigenvalue problem, leads us to discover the counterpart of the limiting state probabilities. Whereas, in the risk-indifferent case, the eigenvalue problem pertained to the transition probability matrix of a policy; in the risk-sensitive case, it pertains to the matrix of "disutility contributions" of the policy, a combined measure of the probabilities of transitions and how much they contribute to "disutility." In this case, we have an eigenvector Z defining the equilibrium flow of those quantities, just as the vector Π of limiting state probabilities defined the equilibrium of probabilistic flows in the risk-indifferent case. It is the components of this vector Z which should weight the test quantities in each state when policy constraints are present. Whereas, weighting by Π was intuitively obvious in the risk-indifferent case (π_i being the time the process spends in state i, on the average, in the long run), the weighting in the risk-sensitive case almost defies intuition. At the very least, it is not transparent. It is the Lagrange multiplier formulation that brings it out.

In Section D, we set out to develop an algorithm similar to the one we developed in Chapter II. We outline a sufficient condition for

guaranteeing policy improvement (since, as before, maximizing the Lagrangian does not, per se, guarantee that). As before, it turns out that, if in every state we disregard alternatives whose test quantities are less than that of the current policy, the policy is guaranteed to improve if PI changes it. We still retain the division into "free" and "coupled" states, improving the free states by regular PI and the coupled states by BB imposing the sufficient condition for improvement. Convergence of BB to a given policy still means that that policy is optimum over the subset of feasible policies that differ with it in exactly one coupled state. In such cases, we make those policies infeasible and continue.

If the policy constraints do not make the best possible policy infeasible, then retaining the feature of maximizing test quantities in each state still enables us to detect that policy without having to exhaust the feasible policy set. The convergence of the algorithm is also proved.

In Section E, we apply the algorithm to the problem of Chapter II when the taxicab driver is not risk-indifferent.

B. Markov Decision Processes without Policy Constraints

In Chapter II, the decision maker was assumed to be risk-indifferent, whence the basic premise was to maximize the expected value of outcomes. For a risk-sensitive decision maker, we have to maximize the expected value of the "utilities" of outcomes, where those "utilities" are defined by the decision maker's "utility function." The latter encodes his attitude towards risk if he subscribes to certain arguments regarding risky propositions or "lotteries." An outcome having value v is assigned a utility $u(v)$, and the expected value of the utilities is

called the utility of the lottery. An important concept in risk-sensitive analysis is that of certain equivalent (CE). The CE of a lottery is the value whose utility is the same as the utility of the lottery

$$u(v) = u(\tilde{v}) \tag{3.1}$$

Thus, if we have a lottery whose utility is x, its certain equivalent $\tilde{v}$ is given by

$$\tilde{v} = u^{-1}(x) \tag{3.2}$$

where u^{-1} is the inverse of the utility function u.

We will restrict ourselves to dealing with a decision maker who subscribes to what is known as the delta property. If all prizes in a lottery are increased by the same amount Δ, his certain equivalent for the lottery increases by Δ. Such a decision maker possesses a utility function which is either linear or exponential. The linear case implies risk indifference, so we will work with exponential utility functions

$$u(v) = -(\text{sgn } \gamma)\, e^{-\gamma v} \tag{3.3}$$

$$u^{-1}(x) = -\frac{1}{\gamma} \ln [(-\text{sgn } \gamma)\, x] \tag{3.4}$$

where γ is the risk aversion coefficient. Risk averters have a positive γ, while risk preferers have a negative γ. (sgn γ) denotes the sign of γ. An important implication of the exponential utility function is the following:

$$u(v+\Delta) = -(\operatorname{sgn}\gamma)\, e^{-\gamma(v+\Delta)}$$

i.e.,
$$= -(\operatorname{sgn}\gamma)\, e^{-\gamma v}\, e^{-\gamma\Delta} \qquad (3.5)$$

$$u(v+\Delta) = e^{-\gamma\Delta}\, u(v)$$

Adding a constant Δ to all lottery prizes causes their utilities to be multiplied by $e^{-\gamma\Delta}$.

Now we are in a position to analyze the Markov Decision Process for a decision maker possessing the Δ property, i.e., an exponential utility function given by (3.3) and (3.4). As usual, we first consider the limited time horizon and then let n tend to ∞. Given a certain policy (i.e., a probability transition matrix and associated reward matrix), the process will generate a total reward $v_i(n+1)$ if it is in state i and is allowed to continue for $n+1$ transitions. This uncertain reward has a certain equivalent $\tilde{v}_i(n+1)$. The CE is that amount the decision maker would be willing to take for certain instead of receiving the uncertain reward generated by the Markov process. *It can be shown [6] that this* CE is given by

$$u\left(\tilde{v}_i(n+1)\right) = \sum_{j=1}^{N} p_{ij}\, u\left[r_{ij} + \tilde{v}_j(n)\right] \qquad (3.6)$$

Using the property given in (3.5), we can reduce (3.6) to

$$u\left(\tilde{v}_i(n+1)\right) = \sum_{j=1}^{N} p_{ij}\, e^{-\gamma r_{ij}}\, u\left(\tilde{v}_j(n)\right) \qquad (3.7)$$

If we define the utility of being in state j, with n transitions remaining, as $u_j(n)$,

$$u_j(n) = u\left(\tilde{v}_j(n)\right) = -(\operatorname{sgn}\gamma)\, e^{-\gamma\tilde{v}_j(n)} \qquad (3.8)$$

Then we can write (3.7) simply as

$$u_i(n+1) = \sum_{j=1}^{N} p_{ij}\, e^{-\gamma r_{ij}}\, u_j(n) \tag{3.9}$$

In the case of risk aversion (positive γ), the term $e^{-\gamma r_{ij}}$ is the negative utility, or the "disutility," of the reward r_{ij}. The term "disutility" will be retained regardless of the sign of γ. If we define the "disutility contribution" of the transition from i to j as

$$q_{ij} = p_{ij}\, e^{-\gamma r_{ij}} \tag{3.10}$$

then we have a disutility contribution matrix Q with elements q_{ij} which are nonnegative. In this case, (3.9) becomes

$$u_i(n+1) = \sum_{j=1}^{N} q_{ij} u_j(n) \tag{3.11}$$

This is the recursive relation for computing successive utilities of the process. To find the certain equivalents, we refer to the definition of the utilities given by (3.8) and use (3.4) to get

$$\tilde{u}_i(n) = -\frac{1}{\gamma} \ln \left[-(\operatorname{sgn} \gamma)\, u_i(n) \right] \tag{3.12}$$

To see what happens when we allow the time horizon to be infinite, we first write (3.11) in vector form. Defining $U(n)$ as the vector whose components are $u_i(n)$, we can write (3.11) as

$$U(n+1) = Q \cdot U(n)$$

This gives

$$U(n) = Q^n \cdot U(0) \tag{3.13}$$

If the Markov transition probability matrix P is irreducible and acyclic, then Q is irreducible and primitive.* In this case, it can be shown that

$$\lim_{n \to \infty} Q^n \cdot U(0) = \lim_{n \to \infty} \left(\frac{1}{\lambda^n}\right) U(n) = k \cdot U \tag{3.14}$$

where λ is the largest eigenvalue of Q, and U is the corresponding eigenvector with k chosen such that $u_N = -(\text{sgn } \gamma)$. Thus, for large n the utility of any state is multiplied by λ at each successive stage. Equations (3.12) and (3.14) may be used to show [6] that the asymptotic form of the certain equivalent can be written as

$$\tilde{v}_i(n) = n\tilde{g} + \tilde{v}_i + c \tag{3.15}$$

where the "certain equivalent gain" $\tilde{g}$ of the process is given by

$$\tilde{g} = -\frac{1}{\gamma} \ln \lambda \tag{3.16}$$

---

*A reducible matrix A is one for which a permutation exists to place it in the form

$$\begin{bmatrix} B & 0 \\ C & D \end{bmatrix}$$

where B and D are square matrices. Otherwise, A is irreducible. An irreducible P is one in which all states communicate. A matrix A is primitive if some power of A has all elements positive. A primitive P is called acyclic.

$\tilde{g}$ is a property of the policy under consideration, and we will be seeking that policy which maximizes it. To compute $\tilde{g}$ for a given policy, we divide (3.11) by λ^n, let $n \to \infty$, and then use (3.14) to get

$$\sum_{j=1}^{N} q_{ij} u_j = \lambda u_i \qquad i = 1,2, \ldots, N \tag{3.17}$$

(3.17) has to be solved for the largest eigenvalue λ of Q, which makes (3.17) of rank $N-1$, i.e., a redundant equation exists. This is overcome by setting $u_N = -(\text{sgn } \gamma)$, which makes $\tilde{v}_N = 0$. However, choosing a value for u_N is not completely arbitrary. That value has to have the opposite sign of γ (more about this in Section C). As soon as we have λ, (3.16) gives us $\tilde{g}$ for the policy under consideration. This phase of the algorithm is called policy evaluation (PE). We need a policy improvement (PI) phase. As in the risk-indifferent case, test quantities are defined, and improving the policy reduces to maximizing the test quantities in each state. The utilities replace the relative values, and the disutility contributions replace the probabilities. In other words, we select an alternative k in each state i such that

$$u_i^k = \max_{k=1,2,\ldots,K_i} \left[\sum_{j=1}^{N} q_{ij}^k u_j \right] \tag{3.18}$$

This is the equivalent of (2.57). The immediate expected rewards are not explicitly present because they are included in the utilities. The algorithm terminates when PI yields the same policy that we entered it with.

C. Markov Decision Processes with Policy Constraints

From the discussion of the previous section, we can view the risk sensitive Markov Decision Process as follows.

Each policy P, made up of an alternative in each state, has associated with it a disutility contribution matrix Q. The maximal eigenvalue λ^M of Q determines the certain equivalent gain of the policy $\tilde{g} = -1/\gamma \ln \lambda^M$. Thus, we are confronted with the task of selecting that Q which yields the largest value of $\tilde{g}$. The sign of γ (type of attitude towards risk) determines our objective. For a risk prefering decision maker, γ is negative. Consequently, maximizing $\tilde{g}$ is the same as maximizing λ^M. For a risk averse decision maker, however, γ is positive, and it is the smallest λ^M which gives the largest $\tilde{g}$.

We have thus ascertained that our objective function is λ^M, the maximal eigenvalue of Q, whence our constraints are those defining the eigenvalue problem that yields λ^M. To gain more insight into this, we reconsider the risk-indifferent case. If we write the constraints defining the limiting state probabilities Π for a transition probability matrix P in vector form, we get

$$\Pi^T \cdot P = \Pi^T \tag{3.19}$$

It is then apparent that Π is a "left eigenvector" of P (or eigenvector of its transpose) with a corresponding eigenvalue of unity. But that eigenvalue is the maximal eigenvalue of P by virtue of it being a stochastic matrix. This holds for any policy. Therefore, in the risk-indifferent case, all maximal eigenvalues are unity. The corresponding left eigenvectors then define the objective function through

$$g = \sum_{i=1}^{N} \pi_i q_i$$

In the risk-sensitive case, we deal with the Q matrices. Their maximal eigenvalues differ and, solving the eigenvalue problem, yields the objective function through the eigenvalue rather than the eigenvector. To get the constraints, then, we define a vector Z, with components z_i, as the left eigenvector of Q. Thus, Z is the counterpart of Π (more about that later), and it should satisfy

$$Z^T \cdot Q = \lambda^M \cdot Z^T \tag{3.20}$$

(3.20) is the counterpart of (3.19), i.e., for a given policy. To take alternative selection into account, we use the d_i^k on the rows of Q, as we did in the risk-indifferent case. We will first concentrate on the risk prefering case, i.e., $\gamma < 0$. *Here, we are dealing with the* constrained maximization problem:

$$\max \lambda^M \tag{3.21}$$

subject to

$$\sum_{i=1}^{N} z_i \sum_{k=1}^{K_i} q_{ij}^k d_j^k - \lambda^M z_j = 0 \qquad j = 1,2, \ldots, N \tag{3.22}$$

$$\sum_{k=1}^{K_i} d_i^k - 1 = 0 \qquad i = 1,2, \ldots, N \tag{3.23}$$

$$\left.\begin{aligned} C_\ell\left(d_i^k\right) \leq 0 \qquad & \ell = 1,2, \ldots, m \qquad (i,k) \in S_1 \\ C_p\left(d_i^k\right) = 0 \qquad & p = 1,2, \ldots, q \qquad (i,k) \in S_2 \end{aligned}\right\} \tag{3.24}$$

where λ^M is the largest positive number satisfying (3.22) and S_1 and S_2 are subsets of $S = \{(i,k)\}$ the set of all (i,k) pairs defining each alternative in each state.

The constraints (3.23) and (3.24) are those we previously encountered as (2.37) and (2.46). The constraints (3.22) are not linear, whence there is no LP equivalent of (3.21) through (3.24). Consequently, we cannot apply the LP result which guarantees that the d_i^k will turn out to have zero-unity values. Rather, when solving the problem, we will restrict ourselves to those values. While other values satisfying (3.23) might give larger values for λ^M, our objective function, we reject them. The reason is that they represent "randomized strategies," a concept which is meaningless in our context.

As noted before, a set of d_i^k describing a policy P determines a disutility contribution matrix $Q(P)$ and its associated left eigenvector Z defined by

$$Z^T \cdot Q(P) = \lambda^M \cdot Z^T \tag{3.25}$$

For an irreducible primitive matrix Q, the components of the maximal eigenvector all have the same sign. They are either all positive or negative. (The transpose of an irreducible primitive matrix is also irreducible and primitive.)

As in Chapter II, we first concentrate on (3.21) through (3.23), i.e., unconstrained policies. We showed that the maximizer of the

constrained problem is a critical point of the corresponding Lagrangian. Here, we have $2N$ Lagrange multipliers corresponding to (3.22) and (3.23). The first N of those, associated with (3.22), we denote by $u_1, u_2, \ldots, u_N$. The other N ones, we denote by β_i. Consequently, our Lagrangian is

$$L = \lambda^M + \sum_{j=1}^{N} \left[\sum_{i=1}^{N} z_i \sum_{k=1}^{K_i} q_{ij}^k d_i^k - \lambda^M z_j \right] u_j + \sum_{i=1}^{N} \beta_i \left[\sum_{i=1}^{N} d_i^k - 1 \right] \qquad (3.26)$$

To maximize our constrained function, we seek critical points of L. As before, we only set certain partial derivatives to zero to evaluate a policy, then we try to change the policy so as to maximize L (because the values of the constrained function and L are identical whenever a policy is evaluated).

Setting the partial derivatives of L w.r.t. the z_i and λ^M equal to zero, we get

$$\frac{\partial L}{\partial z_i} = \sum_{j=1}^{N} u_j \sum_{k=1}^{K_i} q_{ij}^k d_i^k - \lambda^M u_i = 0 \qquad i = 1,2, \ldots, N \qquad (3.27)$$

$$\frac{\partial L}{\partial \lambda^M} = 1 - \sum_{j=1}^{N} z_j u_j = 0 \qquad (3.28)$$

For a given policy P, the d_i^k are all zero or unity with only one d_i^k equal unity in each state. Thus, in (3.27), the summation over k reduces to one element for each i. This defines the $Q(P)$ of the policy under consideration. Defining U as the vector whose components are u_i, (3.27) and (3.28) can be written as

$$Q(P) \cdot U = \lambda^M U \tag{3.29}$$

$$Z^T \cdot U = 1 \tag{3.30}$$

Differentiating L w.r.t. the u_i and equating to zero gives us the original constraints (3.25)

$$Z^T \cdot Q(P) = \lambda^M \cdot Z^T \tag{3.25}$$

The system (3.29) is the PE phase of the PE-PI algorithm. It is an eigenvalue problem, whence the rank of the system is $N-1$. At this point, there is nothing to indicate that setting u_N to some particular value has any significance. We know, however, that U, as well as Z, has components which are either all positive or all negative. Equation (3.30) tells us that Z and U have the same signs. This will become significant in the PI phase. What concerns us here is the relationship of the solutions of (3.25) and (3.29).

In matrix form, (3.29) can be written as

$$\begin{bmatrix} q_{11} - \lambda^M & q_{12} & \cdots & q_{1N} \\ q_{21} & q_{22} - \lambda^M & \cdots & q_{2N} \\ \cdots & \cdots & \cdots & \cdots \\ q_{N1} & q_{N2} & \cdots & q_{NN} - \lambda^M \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ \cdot \\ \cdot \\ \cdot \\ u_N \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \cdot \\ \cdot \\ \cdot \\ 0 \end{bmatrix}$$

This, as we noted earlier, is a system of rank $N-1$. If we set $u_N = a$, say, we get

$$\begin{bmatrix} q_{11} - \lambda^M & q_{12} & \cdots & q_{1,N-1} \\ q_{21} & q_{22} - \lambda^M & \cdots & q_{2,N-1} \\ \cdots & \cdots & \cdots & \cdots \\ q_{N-1,1} & q_{N-1,2} & \cdots & q_{N-1,N-1} - \lambda^M \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ \cdot \\ \cdot \\ \cdot \\ u_{N-1} \end{bmatrix} = -a \begin{bmatrix} q_{1N} \\ q_{2N} \\ \cdot \\ \cdot \\ \cdot \\ q_{N-1,N} \end{bmatrix}$$

(3.31)

where we dropped the last equation. The resulting system can be written as

$$A \cdot \underline{U} = V$$

where A is the matrix on the L.H.S. of (3.31), and V is the vector on the R.H.S. $\underline{U}$ is the $N-1$ vector composed of the first $N-1$ components of U. Hence,

$$\underline{U} = A^{-1} \cdot V \tag{3.32}$$

Now we write (3.25) in matrix form:

$$\begin{bmatrix} q_{11} - \lambda^M & q_{21} & \cdots & q_{N1} \\ q_{12} & q_{22} - \lambda^M & \cdots & q_{N2} \\ \cdots & \cdots & \cdots & \cdots \\ q_{1N} & q_{2N} & \cdots & q_{NN} - \lambda^M \end{bmatrix} \cdot \begin{bmatrix} z_1 \\ z_2 \\ \cdot \\ \cdot \\ \cdot \\ z_N \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \cdot \\ \cdot \\ \cdot \\ 0 \end{bmatrix}$$

This being a system of rank $N-1$, we can set $z_N = b$, say, to get

$$\begin{bmatrix} q_{11} - \lambda^M & q_{21} & \cdots & q_{N-1,1} \\ q_{12} & q_{22} - \lambda^M & \cdots & q_{N-1,2} \\ \cdots & \cdots & \cdots & \cdots \\ q_{1N} & q_{2N} & \cdots & q_{N-1,N-1} - \lambda^M \end{bmatrix} \cdot \begin{bmatrix} z_1 \\ z_2 \\ \cdot \\ \cdot \\ \cdot \\ z_{N-1} \end{bmatrix} = -b \begin{bmatrix} q_{N1} \\ q_{N2} \\ \cdot \\ \cdot \\ \cdot \\ q_{N,N-1} \end{bmatrix} \tag{3.33}$$

Denoting the $N-1$ vector whose components are $z_1,\ldots,z_{N-1}$ by $\underline{Z}$ and the R.H.S. vector of (3.33) by W, we get

$$A^T \cdot \underline{Z} = W$$

where A is the same matrix as in (3.31). Thus,

$$\underline{Z} = \left(A^T\right)^{-1} \cdot W \tag{3.34}$$

And thus solving (3.32) implies that (3.34) is solved. All we need to get Z, once we have U, is to transpose the inverse we already have and multiply it by the transpose of the last row of A (and the value of z_N). This is significant for the case of policy constraints.

Now we interpret the Lagrange multipliers u_i and the z_i. The Z plays here the role that the Π plays in the risk insensitive case. Note that the constraints defining the limiting state probabilities are

$$\Pi^T \cdot P = \Pi^T$$

$$\Pi^T \cdot \underline{1} = 1$$

where $\underline{1}$ is the vector whose components are all unity. That vector is the eigenvector of P corresponding to Π:

$$P \cdot \underline{1} = 1$$

Moreover, the common eigenvalue is unity, the maximal eigenvalue for any transition probability matrix P. Hence, (3.25) represents the "equilibrium of disutility contribution flows" in the limiting case, exactly like the constraints on the limiting state probabilities did for the probabilistic flows. The difference is that here the "outflow" is multiplied by λ^M, correspodning to the fact that, in the limit, utilities are multiplied by λ^M every transition. The u_i, being Lagrange multipliers, give us the cost of violating the constraints per unit of disequilibrium just as the v's were in the risk-insensitive case.

Now we proceed to PI. Once we have evaluated a policy, we want to improve it. We make maximizing the Lagrangian our objective (providing the guarantee of improvement outside the Lagrangian framework, as before). To do that, we rewrite L to bring out the dependence on d_i^k.

$$L = \sum_{i=1}^{N} z_i \sum_{j=1}^{N} u_j \sum_{k=1}^{K_i} q_{ij}^k d_i^k + \lambda^M - \lambda^M \sum_{j=1}^{N} z_j u_j + \sum_{i=1}^{N} \beta_i \left[\sum_{k=1}^{K_i} d_i^k - 1 \right]$$

For the given policy $\sum_{j=1}^{N} z_j u_j = 1$ (from (3.30)) and for any policy $\sum_{k=1}^{K_i} d_i^k = 1$, whence we only have to contend with the first term:

$$\sum_{i=1}^{N} z_i \sum_{j=1}^{N} u_j \sum_{k=1}^{K_i} q_{ij}^k d_i^k \tag{3.35}$$

(3.35) is the inner product of two vectors. The first vector Z is constant. The second is policy dependent. Once a policy P is selected, it defines a vector T(P) of "test quantities"

$$t_i^k = \sum_{j=1}^{N} q_{ij}^k u_j \tag{3.36}$$

where the q_{ij}^k is the corresponding element of the Q selected by policy P. The sign of t_i^k is the same as that of U, and hence also Z. Thus, if Z and U are chosen positive ($u_N = -(\text{sgn } \gamma)$ because we are dealing with the case $\gamma < 0$), then, to maximize L, we have to select that feasible T(P) with largest components. Otherwise, we would have to select that one with the smallest components (if Z and U are negative). In other words, if the sign of u_N is the same as that of γ, the test quantity has to be minimized in order to maximize the Lagrangian. We will assume hereafter that in PE we take $u_N = z_N = -(\text{sgn } \gamma)$.

In the absence of policy constraints, we can select

$$t_i = \max_{k=1,2,\ldots,K_i} \left[\sum_{j=1}^{N} q_{ij}^k u_j \right] \tag{3.37}$$

Since the z_i are nonnegative, this choice maximizes the Lagrangian, and we have the PI phase of Section B. If we have policy constraints, however, such a policy might not be feasible. In this case, we have to consider (3.35) as a whole. In essence, we redefine the components of T to be

$$t_i = z_i \sum_{j=1}^{N} q_{ij}^k u_j \tag{3.38}$$

where the policy P determines the alternative in each state

$$P(i) = k$$

The Lagrangian then becomes merely the sum of the components of $T(P)$. Here, the original test quantities have been weighted by the corresponding z_i, the variables controlling the equilibrium of disutility contribution flows. Whereas, in the risk-insensitive case, weighting test quantities by the limiting state probabilities was intuitive (the process spends π_i of the time in state i, on the average), this weighting cannot be intuitively derived in the risk-sensitive case. Z encodes the limiting behavior from both the probabilistic and risk attitude aspects.

Note that, if the test quantities are defined by (3.38), i.e., we multiply by z_i, then, when maximizing the Lagrangian, we do not have to worry about the signs of Z and U. They cancel out. Thus, in the absence of policy constraints, we can set $u_N = (\text{sgn } \gamma) = -1$ but take care to multiply the test quantities by -1 before maximizing in each state. This is just another way of saying that u_N has to be positive when γ is negative.

In other words, (3.29) is not really a "free" eigenvalue problem, in the sense that we are not free to choose any value for one of the u_i. The values have to be of different sign than γ in the absence of policy constraints in order to implement the PI of Section B as is. Otherwise, the test quantity has to be minimized. It is the realization that PI is maximization of the Lagrangian which led us to detect this dependence on sign γ. As mentioned, this dependence can be removed if the test quantities are multiplied by the z_i. Since this is what we

do when policy constraints are present, we need not worry about signs. We will select $z_N = u_N = -(\text{sgn } \gamma)$ for consistency. (Either Z or U has then to be normalized to satisfy (3.30).) For constrained policies, we maximize the Lagrangian by BB as before. That BB does maximize the Lagrangian has already been proved. The proof still applies because the Lagrangian was only assumed to be the sum of the components of T(P) (which it still is) without any dependence on how those components were obtained.

For the case of risk aversion, positive γ, we previously mentioned that we need to minimize λ^M in order to maximize $\tilde{g}$. In this case, our constrained problem is the same as (3.21) through (3.24), except that (3.21) is replaced by $\min \lambda^M$. What applies to constrained maximization applies to constrained minimization, as far as the Lagrange multiplier rule is concerned.

Thus, the minimizer of the constrained problem is still a critical point of the Lagrangian. Hence, the PE phase is the same. When we get down to PI, however, we would like to select P so as to minimize the Lagrangian. In the absence of policy constraints, setting $u_N = -(\text{sgn } \gamma)$ makes U and Z both negative. Thus, to minimize the inner product, the variable vector composed of test quantities has to be <u>maximized</u> (because it will be multiplied by a negative vector, and we want to minimize the result). Hence, setting $u_N = -(\text{sgn } \gamma)$ in the unconstrained policies case, and then maximizing the resultant test quantities, actually <u>minimizes</u> the Lagrangian. This is what is really sought here. For constrained policies, we merely minimize the Lagrangian without worrying about signs, because we multiply the test quantities by z_i. A better

approach would be to take the sign of γ into consideration, explicitly. This could be done by redefining the test quantities as

$$t_i^k = -(\text{sgn } \gamma)\, z_i \sum_{j=1}^{N} q_{ij}^k u_j \tag{3.39}$$

For a risk preferrer ($\gamma < 0$), this reduces to (3.38), whence all the previous applies. For $\gamma > 0$, (3.39) effectively multiplies the Lagrangian by -1. In other words, the Lagrangian defined by (3.39) is the negative of that defined by (3.38). Since we want to minimize the latter, we can maximize the former. Consequently, PI becomes maximization, irrespective of the sign of γ if we define the test quantities by (3.39).

To summarize then, setting $u_N = -(\text{sgn } \gamma)$ makes PI maximize test quantities in the absence of policy constraints. While this involves dependence on the sign of γ, it automatically takes into account the fact in one case we want to maximize L and in the other minimize it. When policy constraints are present, we have to explicitly take the sign of γ into account by defining the test quantities as in (3.39), whence we always maximize L in the PI.

D. Development and Convergence of the Algorithm

As in Chapter II, the maximization of L does not, per se, guarantee an improvement in $\tilde{g}$. We have to provide that guarantee outside the Lagrangian framework. We will introduce a sufficient condition for improving $\tilde{g}$, based on a condition derived by Howard and Matheson [6]. If we have a policy A, then the test quantity corresponding to the alternative selected by any other policy B in state i is defined as

$$t_i(B) = -(\text{sgn } \gamma)\, z_i^A \sum_{j=1}^{N} q_{ij}^B u_j^A \tag{3.40}$$

where z_i^A and u_j^A are the values obtained by the PE for policy A. We will define Δ_i's analogous to the γ_i's of the risk-indifferent case by

$$\Delta_i(B,A) = t_i(B) - t_i(A) \tag{3.41}$$

where the t_i are defined by (3.40).

Proposition 3.1.

Given a policy A for which PE has been performed and any other policy B, a sufficient condition for $\tilde{g}^B > \tilde{g}^A$ is

$$\Delta_i(B,A) \geq 0 \qquad i = 1,2, \ldots, N \tag{3.42}$$

with inequality holding for at least one state i.

Proof.

We note that since $\tilde{g} = -1/\gamma \ln \lambda$, where λ is the maximal eigenvalue of Q, then we need to prove that $\lambda^B > \lambda^A$ for $\gamma < 0$ and vice versa.

An important result from matrix theory is that, if Q is a nonnegative irreducible matrix with maximal eigenvalue λ and x is a vector with positive components x_i, then

$$\min_i \frac{\sum_{j=1}^{N} q_{ij} x_j}{x_i} \leq \lambda \leq \max_i \frac{\sum_{j=1}^{N} q_{ij} x_j}{x_i} \tag{3.43}$$

with equality holding if and only if x is an eigenvector of Q.

Now we consider the implications of (3.42). By virtue of (3.41), we can write

$$t_i(B) \geq t_i(A)$$

with inequality holding in at least one state. Using (3.40), this reduces to

$$-(\text{sgn } \gamma)\, z_i^A \sum_{j=1}^{N} q_{ij}^B u_j^A \geq -(\text{sgn } \gamma)\, z_i^A \sum_{j=1}^{N} q_{ij}^A u_j^A$$

Since, from PE, z_i and γ have different signs, the product $-(\text{sgn } \gamma)\, z_i$ is always positive, and we can divide both sides of the inequality by that quantity without reversing its sense. Also from PE, U is an eigenvector of Q^A, whence

$$\sum_{j=1}^{N} q_{ij}^B u_j^A \geq \lambda^A u_i^A \tag{3.44}$$

with inequality holding in at least one state i. If U is also an eigenvector of Q^B, (3.44) reduces to

$$\lambda^B u_i^A \geq \lambda^A u_i^A$$

with

$$\lambda^B u_i^A > \lambda^A u_i^A \qquad \text{for some } i$$

For

$$\gamma < 0, \qquad u_i > 0 \quad \text{and} \quad \lambda^B > \lambda^A$$

For

$$\gamma > 0, \qquad u_i < 0 \quad \text{and} \quad \lambda^B < \lambda^A$$

If U is not an eigenvector of Q^B, we consider $\gamma < 0$ first, then $\gamma > 0$.

For $\gamma < 0$, $u_j > 0$, and (3.44) can be rewritten as

$$\frac{\sum_{j=1}^{N} q_{ij}^B u_j^A}{u_i^A} \geq \lambda^A$$

Since this holds for all i, then it is certainly true that

$$\min_i \sum_{j=1}^{N} \frac{q_{ij}^B u_j^A}{u_i^A} \geq \lambda^A$$

Applying (3.43) to the L.H.S. of this inequality, noting that U is not an eigenvector of Q^B, we get

$$\lambda^B > \lambda^A$$

For $\gamma > 0$, $u_i < 0$ and (3.44) can be rewritten as

$$\sum_{j=1}^{N} q_{ij}^B \left(-|u_j^A|\right) \geq \lambda^A \left(-|u_i^A|\right)$$

Hence,

$$\frac{\sum_{j=1}^{N} q_{ij}^B |u_j^A|}{|u_i^A|} \leq \lambda^A$$

Since this holds for all i, it is certainly true that

$$\max_i \frac{\sum_{j=1}^{N} q_{ij}^B |u_j^A|}{|u_i^A|} \leq \lambda^A$$

Applying (3.43) to the L.H.S. with U not an eigenvector of Q^B gives $\lambda^B < \lambda^A$.

We thus have a sufficient condition for improving $\tilde{g}$. The foregoing proposition states, in effect, that "improving" the test quantity, as defined by (3.40), in at least one state suffices to improve $\tilde{g}$. Thus, when we enter BB, we will discard alternatives for which $\Delta_i < 0$. If BB yields a policy different to the one we entered with, it is automatically an improvement. Otherwise, the policy on which BB converges maximizes $\tilde{g}$ over the set of feasible policies that differ with it in exactly one "coupled" state.

The previous proposition also guarantees that if, for a policy A, each test quantity is maximum in its state, then policy A is optimum. The proof is identical, except that equality is allowed to hold in all states. Finally, to get an initial feasible policy, we maximize $-(\text{sgn } \gamma) \sum_{j=1}^{N} q_{ij}$ over the feasible policy set by BB. Thus, the algorithm can be schematically represented in Figure 3.1.

The convergence proof is identical to that of Chapter II. If we exit the algorithm at E2, that policy is the overall optimum. No policy can have a better $\tilde{g}$. (This is the policy PE-PI would converge to. In other words, the policy constraints have not altered the optimum policy.)

If we exit the algorithm at E3, then any feasible policy has become infeasible by virtue of (2.71) type constraints. This means that it belongs to a subset over which we maximized $\tilde{g}$, whence it cannot have a better $\tilde{g}$ than the one we obtained.

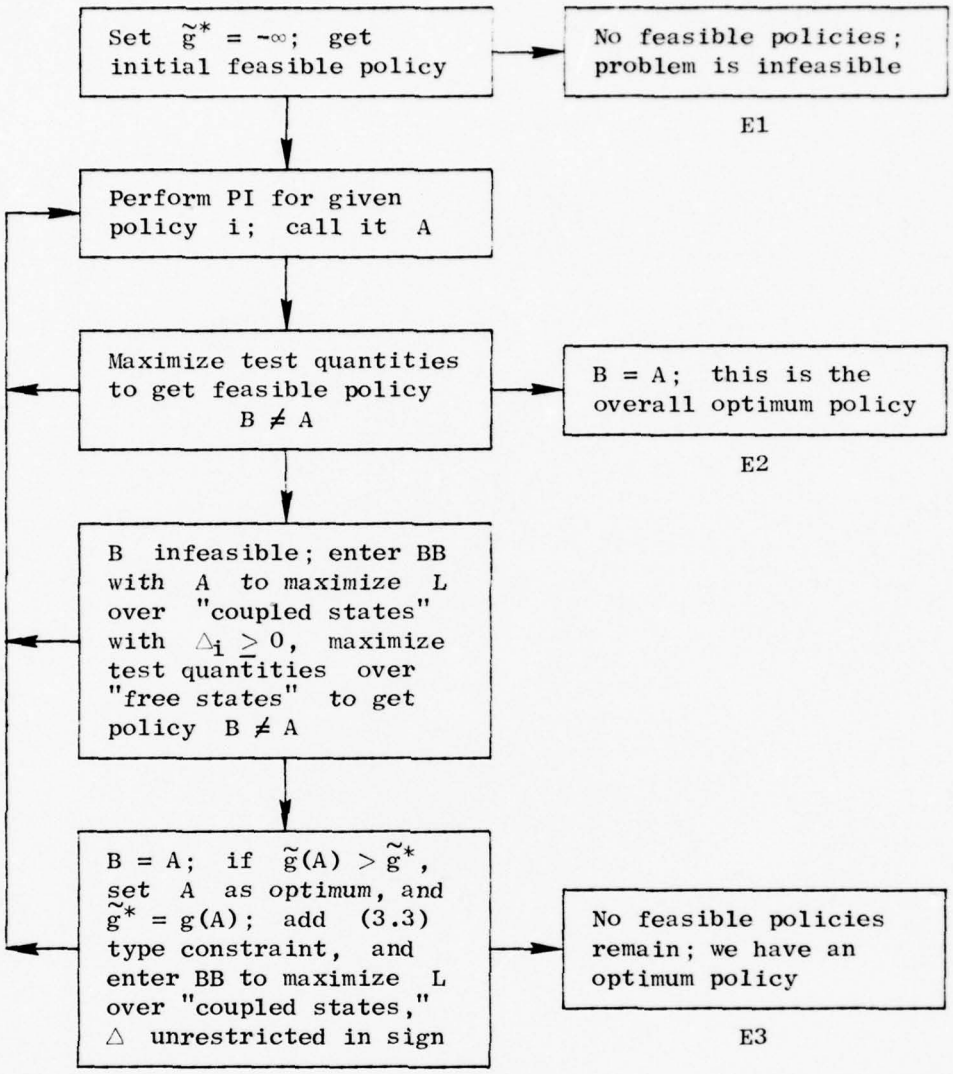


Fig. 3.1. ALGORITHM FOR RISK-SENSITIVE CASE.

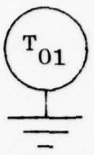
E. The Example

We will solve the same example we solved in Chapter II, introducing a risk averse coefficient $\gamma = 0.01$. We rewrite the original policy constraints (2.72) and (2.73) as

$$d_1^1 - d_2^1 = 0 \tag{3.45}$$

$$d_1^2 + d_2^2 \leq 1 \qquad (3.46)$$

To get an initial feasible policy, we enter BB to maximize $-\sum_{j=1}^{N} q_{ij}$ over the feasible policy set. The BB tree:



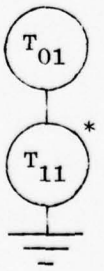
$T_{01} = -(0.923, 0.852, 0.933)$ $\qquad$ $P_{01} = (1,1,1)$

Performing PE for the feasible policy (1,1,1) results in:

State	Ordered Test Quantities		
1	$t_1^1 = -0.362$	$t_1^2 = -0.369$	$t^3 = -0.380$
2	$t_2^2 = -0.173$	$t_2^1 = -0.181$	
3	$t_3^2 = -0.367$	$t_3^1 = -0.369$	$t_3^3 = -0.381$

$A = (1,1,1)$ $\qquad$ $\tilde{g}(A) = 9.19$

Regular PI gives (1,2,2), an infeasible policy. Maximizing the free state gives P(3) = 2, and the BB tree is:



$T_{01} = -(0.362, 0.173)$ $\qquad$ $P_{01} = (1,2)$

$T_{11} = -(0.362, 0.181)$ $\qquad$ $P_{11} = (1,1)$

Thus, we get policy B = (1,1,2), different to the one we started with. Performing PE for (1,1,2) gives:

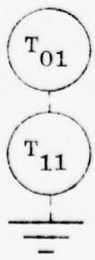
State	Ordered Test Quantities		
1	$t_1^1 = -0.357$	$t_1^2 = -0.364$	$t_1^3 = -0.374$
2	$t_2^2 = -0.273$	$t_2^1 = -0.286$	
3	$t_3^2 = -0.268$	$t_3^1 = -0.269$	$t_3^3 = -0.278$

$A = (1,1,2)$ $\quad \tilde{g}(A) = 9.34$

Regular PE yields (1,2,2) which violates (3.45), i.e., is infeasible. Hence, we maximize in 3 to get P(3) = 2, and we enter BB with $\Delta_i \geq 0$, i.e.,

State	Ordered Test Quantities	
1	t_1^1	
2	t_2^2	t_2^1

The BB tree is:



$T_{01} = -(0.357,0.273)$ $\quad P_{01} = (1,2)$

$T_{11} = -(0.357,0.286)$ $\quad P_{11} = (1,1)$

Thus, we get policy (1,1,2), the same one we entered with. Hence, our optimum policy P_o and $\tilde{g}^*$ so far are

$$P_o = (1,1,2) \qquad \tilde{g}^* = 9.34$$

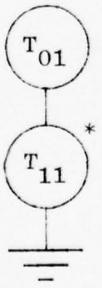
We add the constraint

$$d_1^1 + d_2^1 \leq 0 \tag{3.47}$$

And we enter BB with no restrictions on Δ_i, i.e.,

State	Ordered Test Quantities		
1	t_1^1	t_1^2	t_1^3
2	t_2^2	t_2^1	

The BB tree is:



$$T_{01} = -(0.357,0.273) \qquad P_{01} = (1,2)$$

$$T_{11} = -(0.374,0.273) \qquad P_{11} = (3,2)$$

Hence, we have a policy (3,2,2) for which we perform PE:

State	Ordered Test Quantities		
1	$t_1^2 = -0.091$	$t_1^1 = -0.096$	$t_1^3 = -0.099$
2	$t_2^2 = -0.656$	$t_2^1 = -0.756$	
3	$t_3^2 = -0.128$	$t_3^1 = -0.134$	$t_3^3 = -0.147$

$A = (3,2,2)$ $\quad \tilde{g}(A) = 12.40$

Regular PE yields an infeasible policy. Thus, we maximize in 3 and enter BB with:

State	Ordered Test Quantities		
1	t_1^2	t_1^1	t_1^3
2	t_2^2		

The BB tree:

$T_{01} = -(0.091,0.656)$ $\quad P_{01} = (2,2)$

$T_{11} = -(0.099,0.656)$ $\quad P_{11} = (3,2)$

And we got (3,2,2) the policy we entered with. Since its $\tilde{g}$ is greater than $\tilde{g}^*$, we update:

$$P_o = (3,2,2) \qquad \tilde{g}^* = 12.40$$

We add the constraint

$$d_1^3 + d_2^2 \leq 0 \tag{3.48}$$

And we enter BB with:

State	Ordered Test Quantities		
1	t_1^2	t_1^1	t_1^3
	|	‖	
2	t_2^2	t_2^1	

The BB tree:

$$T_{01} = -(0.091,0.656) \qquad P_{01} = (2,2)$$

No feasible policy exists in the tree, i.e., the problem has become infeasible. Thus, our optimum policy and $\tilde{g}$ are

$$P_o = (3,2,2) \qquad \tilde{g}^* = 12.40$$

This is the same optimum policy as in Chapter II, the risk-indifferent case. Thus, a small risk-aversion coefficient of 0.01 does not change the optimum policy. The negligible effect of such a small coefficient may also be detected by regarding the risk premium, the difference between expected value and expected utility decision making:

$$g^* - \tilde{g}^* = 12.774 - 12.40 = 0.374$$

This, however, is not our concern here. Our main objective is to have an algorithm which works for both risk-indifferent and risk-sensitive cases. This objective has been achieved.

Chapter IV

SENSITIVITY OF OPTIMAL POLICY TO CONSTRAINTS

In this chapter, we investigate the effect of policy constraints on the optimal policy and how much a rational decision maker would be willing to pay in order to remove one or more constraints.

Our point of departure is the absence of policy constraints. In this case, no policy can yield a higher gain (or certain equivalent gain) than the policy P_o arrived at by Howard's VD-PI (or PE-PI) algorithm. When policy constraints are introduced, policy P_o might, or might not, become infeasible. If the set of policy constraints does not make P_o infeasible, it is still the optimal policy, and the constraints are merely a red herring. In other words, we can tell the decision maker that, in this case, his concern about policy constraints is much ado about nothing. We define such an optimal policy as a "constraint-indifferent" optimal policy, and all constraints are worthless, in the sense that the decision maker has nothing to gain by removing any of them. Moreover, we do not have to solve the problem in the absence of policy constraints to detect constraint-indifference. The algorithm we developed has the ability to detect that, as we showed in the convergence proofs. If we exit the algorithm from E2, we have a constraint-indifferent optimal policy. The distinguishing feature of the E2 exit is that, for the selected policy, each alternative maximizes the test quantity in its state. Therefore, any other policy results in nonpositive γ_i, whence the gain cannot increase.

If we exit the algorithm from E3, we have what we define as a "constraint-sensitive" optimal policy. Here, the policy having the highest

possible gain is infeasible. The constraints have affected the optimal policy. If we look at the table of ordered test quantities for the last iteration, there will be at least one state in which the selected alternative does not maximize the test quantity (otherwise, we would have exited from E2). For such states, the only thing that prevented BB from maximizing the test quantities, is the feasibility. Thus, it might be worthwhile to pay for removing some constraints. There is, however, a subset of constraints (possibly the null set) which do not affect the optimal policy and can be determined from the final (i.e., last iteration) table of ordered test quantities. Assume that the coupled states are numbered $1,2,\ldots,M \leq N$, and that the alternatives selected by the optimal policy in those states are $a,b,c,\ldots,m$, respectively.

We start by listing the table of ordered test quantities for the coupled states of the optimal policy. Without loss of generality, we have renumbered the alternatives in each state according to the descending order of their test quantities.

State	Ordered Test Quantities
1	$t_1^1 \quad t_1^2 \quad \ldots \quad t_1^a \;\vert\; t_1^{a+1} \quad \ldots\ldots \quad t_1^{K_1}$
2	$t_2^1 \quad t_2^2 \quad \ldots\ldots\ldots \quad t_2^b \;\vert\; t_2^{b+1} \quad \ldots\ldots \quad t_2^{K_2}$
.	
.	
.	
M	$t_M^1 \quad t_M^2 \quad \ldots \quad t_M^m \;\vert\; t_M^{m+1} \quad \ldots\ldots \quad t_M^{K_M}$

The line we have drawn through the table is the line representing the restriction $\gamma_i \geq 0$. No alternatives to the right of this line are

considered in BB. Our claim is that, if the d_i^k involved in a constraint are all to the right of that line, the optimal policy is not affected by that constraint. This is due to the fact that only $\gamma_i > 0$ can yield a better policy, and all such policies (to the left of the line) are infeasible (otherwise, we would have converged to the best).

Finally, the alternatives selected by the optimal policy in the coupled states are given by

$$P(1) = a \qquad P(2) = b, \ldots, P(M) = m$$

Each policy constraint $C_p(d_i^k) \leq 0$ (or $= 0$) involves d_i^k such that $i \in \{1,2,\ldots,M\}$. Now consider the subset of constraints

$$C = \left\{ C_p\left(d_i^k\right) : k > P(i) \right\}$$

Then C is composed of constraints which do not affect the optimal policy, i.e., if they are discarded, the solution would not change. The proof is simple. If a constraint involves d_i^k such that $k > P(i)$ for all i, then the extra feasible policies resulting from discarding that constraint have negative γ_i's (to the right of the line), whence their gain is inferior to the policy we already obtained. Consequently, the algorithm would not converge to any of them. It would still converge to the same optimal policy.

Thus, when the algorithm terminates, we can immediately determine whether or not there are worthless constraints. If we exit from E2, we have a constraint-indifferent optimal policy. If we exit from E3, we can look at the table of ordered test quantities and detect those constraints that the decision maker need not have concerned himself with.

At this point, he should not be willing to pay for removing any of them; he would not gain anything. As for the remainder of the constraints, some of them might also be worthless, but some of them definitively are not. To discover the worth of any single constraint, we can remove it and solve the problem again, starting with the optimal policy as our initial feasible policy. This we do for the taxicab example in the risk-indifferent case. There, our policy constraints were

$$d_1^1 - d_2^1 = 0 \tag{4.1}$$

$$d_1^2 + d_2^2 \leq 1 \tag{4.2}$$

We converged to policy $P_o = (3,2,2)$ and gain $g \approx 12.77$.

State	Ordered Test Quantities		
1	$t_1^2 = 1.04$	$t_1^1 = 0.58$	$t_1^3 = 0.31$
2	$t_2^2 = 20.69$	$t_2^1 = 9.09$	
3	$t_3^2 = 1.51$	$t_3^1 = 0.95$	$t_3^3 = -0.18$

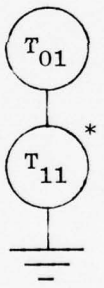
$P_o = (3,2,2)$ $\quad g = 12.77$

Here, we have drawn the $\gamma_i \geq 0$ line and the constraints. Neither constraint involves d_i^k which are all to the right of that line. Thus, we do not have, as yet, any worthless constraints. Let us see what happens if we discard (4.1). This means that the union drop the restriction of using its facilities but still allows only one taxi-cab

stand to be used. Our initial feasible solution is (3,2,2). We note that no improvement can be made in the free state, and maximizing over the coupled ones results in the infeasible policy (2,2,2). Hence, we enter BB with the next table. (We shuffled the states around to have the one with least alternatives as the first component. This reduces the BB computations.)

State	Ordered Test Quantities		
2	t_2^2		
	|		
1	t_1^2	t_1^1	t_1^3

The BB tree is:



T_{01} = (1.04,20.69) P_{01} = (2,2)

T_{11} = (0.58,20.69) P_{11} = (1,2)

Hence, we move to (1,2,2), a feasible policy different to the one we entered with. Performing VD gives us

State	Ordered Test Quantities		
1	$t_1^2 = 1.47$	$t_1^1 = 1.12$	$t_1^3 = 0.59$
2	$t_2^2 = 20.48$	$t_2^1 = 11.08$	
3	$t_3^2 = 1.2$	$t_3^1 = 0.84$	$t_3^3 = 0.22$

$$P = (1,2,2) \qquad g = 13.15$$

No improvement is possible via maximizing test quantities, so we enter BB with

State	Ordered Test Quantities	
2	t_2^2	
1	t_1^2	t_1^1

The BB tree is identical to the previous one, i.e., we converge to (1,2,2). Therefore, we introduce the constraint

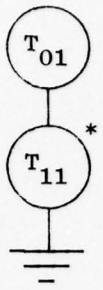
$$d_1^1 + d_2^2 \leq 0$$

and we set our optimal policy so far as

$$P_o = (1,2,2) \qquad g^* = 13.15$$

State	Ordered Test Quantities		
2	t_2^2	t_2^1	
1	t_1^2	t_2^1	t_2^3

The BB tree is given:



$T_{01} = (1.47, 20.48)$ $\quad P_{01} = (2,2)$

$T_{11} = (1.47, 11.08)$ $\quad P_{11} = (2,1)$

The VD for (2,1,2) gives

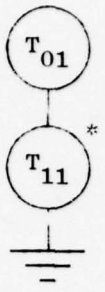
State	Ordered Test Quantities		
1	$t_1^1 = 2.44$	$t_1^2 = 2.05$	$t_1^3 = 1.29$
2	$t_2^2 = 8.87$	$t_2^1 = 6.61$	
3	$t_3^2 = 2.66$	$t_3^1 = 2.53$	$t_3^3 = 1.22$

$P = (2,1,2)$ $\quad g = 8.81$

No improvement is possible, so we enter BB with

State	Ordered Test Quantities		
2	t_2^2	t_2^1	
1	t_1^1	t_1^2	t_1^3

The BB tree is:



$$T_{01} = (2.44,8.87) \qquad P_{01} = (1,2)$$

$$T_{11} = (2.05,6.61) \qquad P_{11} = (2,1)$$

And we have converged to $(2,1,2)$. Since $g < g^*$, our previous optimal policy is still optimal so far. We introduce the constraint

$$d_1^2 + d_2^1 \leq 0 \tag{4.4}$$

and enter BB with

State	Ordered Test Quantities
2	t_2^2 t_2^1
1	t_1^1 t_1^2 t_1^3

The BB tree is:

$$T_{01} = (2.44,8.87) \qquad P_{01} = (1,2)$$

The only node in the tree is infeasible, and no branching is possible, i.e., an infeasible problem. Thus, we have exhausted the set of feasible policies and exited from E3. Our necessarily constraint-sensitive optimal policy is

$$P = (1,2,2) \qquad g = 13.15$$

The difference in gains between this optimal policy and that in the presence of (4.1) is

$$\Delta g_1 = 13.15 - 12.77 = 0.38$$

Therefore, it is not worth more than 0.38 units per transition for the taxi-cab driver to try removing (4.1). For instance, if there were a proposition in the union for raising the dues in return for abolishing the rule that forces a member to use union facilities, he should not vote for it if the increase in dues averages more than 0.38 per trip. Otherwise, he votes for the proposition. He stands to gain by changing his optimal policy under the new rules of the game. Note that there still is a more gainful policy if (4.2) were also discarded (the constraint-indifferent policy; here, it would be constraint-indifferent by default because there would be no constraints). We know about the existence of that policy from the fact that we did not exit from E2.

We could discard (4.2) and start with the policy (1,2,2) to get the optimal one, thus computing how much that constraint is worth in the absence of (4.1). A more interesting thing happens, however, if we retain (4.1) and discard (4.2). In other words, if, after solving the problem in the presence of both (4.1) and (4.2) we want to know how much

(4.2) is worth, we would start with policy (3,2,2) and only (4.1) as a constraint. We repeat the policy's table here.

State	Ordered Test Quantities		
1	$t_1^2 = 1.04$	$t_1^1 = 0.58$ ‖	$t_1^3 = 0.31$
2	$t_2^2 = 20.69$	$t_2^1 = 9.09$	
3	$t_3^2 = 1.51$	$t_3^1 = 0.95$	$t_3^3 = -0.18$

$P_o = (3,2,2)$ $\quad g = 12.77$

Here, no improvement can be made in the free state, but we can maximize over the coupled states to get:

State	Ordered Test Quantities		
1	$t_1^2 = 0.82$	$t_1^1 = 0.71$ ‖	$t_1^3 = 0.37$
2	$t_2^2 = 22.29$	$t_2^1 = 13.21$	
3	$t_3^2 = 1.01$	$t_3^1 = 0.75$	$t_3^3 = 0.33$

$P = (2,2,2)$ $\quad g = 13.34$

Here, the test quantities are all maximum in their states under this policy, and we exit from E2. Hence, (2,2,2) is the best policy he can ever implement. It is also constraint-indifferent. In other words, it is only the rule that only one union taxicab stand can be used that need concern him. Its value is

$$\Delta g_2 \approx 13.34 - 12.77 = 0.57$$

Note, however, that, in the absence of constraint (4.1), constraint (4.2) is worth

$$\Delta g_3 \approx 13.34 - 13.15 = 0.19$$

This is arrived at by the fact that, in the absence of (4.1), we converged to a gain of 13.15. Had we then discarded (4.2) and started with the then optimal policy (1,2,2), we would have converged to (2,2,2). The values naturally turn out to be additive. (Otherwise, we would have a "money pump" situation). In other words, a constraint does not usually have a value independent of other constraints. There could be, however, a constraint (or group of constraints) that renders the others worthless. In our example, constraint (4.2) was such a constraint. If that constraint representing the one-stand-only rule were removed, the other constraint is worthless. To achieve that, a value of 0.57 per transition is an upper bound. If only (4.1) were removed at a cost of 0.38 per transition, then removing (4.2) would be worth 0.19. Thus, to get the constraint-indifferent optimal policy, our friend could do one of two things. He could expend up to 0.38 per transition to remove the use-the-union-facilities rule and up to 0.19 per transition to remove the one-stand-only rule for a total of 0.57 per transition. Alternatively, he could expend up to 0.57 per transition to remove the one-stand-only rule and not bother about the first rule. In either case, there is an (identical) upper bound on how much he should be willing to pay to achieve that constraint-indifferent optimal policy.

In general, it is not that simple to discover combinations that achieve the constraint-indifferent optimal policy. The fact is, however, that no matter how it can be achieved, there is a unique upper bound on the value of doing so, namely the difference in gain between the constraint-indifferent optimal policy and the constraint-sensitive optimal policy given by the algorithm when all constraints are taken into consideration. If the decision maker is interested in the aforementioned upper bound, we could start discarding constraints, one at a time, and solving the problem starting with the latest constraint-sensitive optimal policy as an initial feasible policy until we eventually get the constraint-indifferent optimal policy. We can give him two things here. First, the breakdown of the upper bound between constraints. Secondly, we tell him that, no matter what, the constraint-indifferent optimal policy is the best he can ever hope to achieve for the given problem.

If the decision maker is interested only in specific constraints, or groups thereof, we can compute the worth of such constraints by the afore mentioned technique; it being understood that the worth we compute of specific constraints is subject to the presence of the remainder of the constraints.

The problem of determining specifically which constraint, or group of constraints, render the rest worthless (i.e., discarding them results in a constraint-indifferent optimal policy), or even the minimal such group, is an essentially combinatorial problem worthy of research.

Chapter V

MODIFICATIONS FOR PROBLEMS WITH A LARGE NUMBER OF STATES

A. Introduction

As explained earlier, for a constraint-sensitive optimal, we have to exhaust the feasible policy set. That set becomes very large as the number of states increases. At the same time, the rate at which it is exhausted is slow. This we discuss in further detail in Section B. In Section C, we introduce the idea of partitions, i.e., dividing the coupled states into groups which have no intergroup couplings. We show how partitioning increases the rate of exhausting the feasible policy set, discuss the problems introduced by partitioning, and how to overcome them. In Section D, we elaborate on how to deal with transient states if we have a single trapping state (whence all policies have the same gain). In Section E, we discuss a modification which might further speed up the algorithm. Finally, in Section F, we introduce various constraints to Howard's baseball problem [4] and give the computational results.

B. Of Dimensions and Exhaustion

As the "dimension" of the problem, i.e., the number of states increases, the number of policies increases multiplicatively. This is because the latter is given by $\prod_{i=1}^{N} K_i$. Thus, adding one state with 3 alternatives, say, increases the number of policies by a multiplicative factor of 3. While it is true that the constraints eliminate some of these policies, we can still consider that, in general, we have a fairly

large number of feasible policies to consider. In the case of a constraint-sensitive optimal policy, we have to exhaust the feasible policy set (exit E3 in the algorithm). As we have seen, whenever branch and bound converges on a policy, that policy maximizes the gain over the subset of policies differing with it in exactly one coupled state. That subset is discarded, and exhaustion occurs when the union of such subsets covers the feasible policy set. Now, if we consider the number of policies in any one of the afore-mentioned subsets, we find that it is given by $\sum_{i=1}^{N} K_i - N + 1$. (Here, we assume, for illustrative purposes, that all the N states are coupled.) Of this number of policies, some might already be infeasible, either due to the original constraints or type (2.71) constraints. Hence, the exhaustion process is, at best, additive in nature; whereas the set to be exhausted has a size whose nature is multiplicative. In other words, the feasible set is being exhausted at too slow a rate. It thus appears that, in the absence of additional structural properties, the computational cost of the algorithm would be prohibitive for problems with a large number of states.

C. Of Partitions

For practical problems with a large number of states, it might very well turn out that the coupled states can be divided into groups having no inter-group coupling of alternatives. (We will later give examples.) Those groups we term "partitions." We thus define a partition as a group of coupled states in which no alternative in any state appears in a constraint involving alternatives in any state not in the partition. Formally: *Let the m^{th} constraint* (C_m) be defined by the set of ordered pairs (i,k) referring to the d_i^k's involved in the constraint, i.e.,

$$C_m = \left\{\left[i_1(m),k_1(m)\right], \ldots, \left[i_\ell(m),k_\ell(m)\right]\right\}$$

e.g., for $d_1^2 + d_3^1 + d_5^2 \geq 1$,

$$C = \{(1,2),(3,1),(5,2)\}$$

Let the r^{th} partition be defined by its constituent states:

$$P_r = \left\{i_1(r), \ldots, i_N(r)\right\}$$

Then our definition becomes

$$i \in P_r \Longleftrightarrow \left\{(i,k) \in C_m \Rightarrow i_j(m) \in P_r \ \forall\left[i_j(m),k_j(m)\right] \in C_m\right\}$$

We have thus characterized the states further. First, we divide them into free and coupled states. Then the coupled states are further decomposed into partitions (whence a partition may be regarded as a "generalized free state"). The advantage of partitioning is explained by the following proposition.

Proposition 5.1.

Let the maximization of the Lagrangian over the set of feasible policies subject to $\gamma_i \geq 0$ yield a policy P for which all $\gamma_i = 0$ (i.e., branch and bound converged to P). Then P maximizes the gain over the set of all feasible policies that differ with P in at most one state in each partition.

Proof.

Let P' be a feasible policy differing with P in at most one state in each partition. First, recall that

$$g(P') - g(P) = \Delta g = \sum_{i=1}^{N} \pi_i(P')\, \gamma_i \tag{2.66}$$

Now consider each partition. If P' is identical with P throughout a partition, then $\gamma_i = 0$ for all states in that partition. If P' differs with P in exactly one state in a partition, then $\gamma_i < 0$ for that state. (Otherwise, the Lagrangian could have been improved, and that did not happen.) As for the free state, $\gamma_i = 0$. (The alternatives there maximize the test quantities.) Therefore, $\gamma_i \leq 0$ for all states, whence $\Delta g \leq 0$ and P cannot be inferior, in gain, to P'.

The importance of the previous result is that, with partitions, the rate of exhausting the feasible set is greatly enhanced. The number of policies in a subset discarded by a policy P has a rough estimate of $\prod_{r=1}^{R} [\sum_{i \in P_r} K_i - N_r + 1]$, where R is the number of partitions and N_r is the number of states in partition P_r. Actually, the previous estimate is on the high side because we have to subtract from it those combinations that cause a change in more than one partition. However, that estimate serves to illustrate the fact that the rate of exhaustion is better than additive, a definite improvement on the case where no partitions exist. This contention has been borne out in the computational results.

The introduction of partitions complicates matters slightly. First, we have the problem of additional (2.71) type constraints, to implement

discarding subsets over which we maximize the gain. We could add a constraint for each partition. In this case, however, we would have to consider combinations of them to detect whether or not a given policy belongs to a discarded set. This problem is resolved by applying the result of the proposition directly without resort to formal constraints of type (2.71). Every time branch and bound converges to a policy; that policy is retained in lieu of a constraint. Then, given any policy, it is compared to the retained policies. If the given policy differs with a retained policy by, at most, one alternative in each partition, that given policy belongs to a set over which we have maximized. It is ignored, i.e., considered infeasible.

The second problem that arises is how to detect that the set of feasible policies has been exhausted. Whenever BB converges to a necessarily feasible policy P, we enter BB with no restriction on γ_i for each partition in turn, all other partitions being held constant (i.e., no change in alternatives). If BB finds the problem infeasible for one partition, this does not imply exhaustion. Only if all partitions yield an infeasible problem, is the feasible policy set exhausted. We have, however, to prove this assertion.

Assume that for policy P, all partitions yield an infeasible problem. Assume that there exists a feasible policy P' whose gain is greater than the optimum obtained so far. This implies that P' does not belong to any set over which we have maximized the gain. This, in turn, implies that for each retained policy R, there exists at least one partition in which R and P' differ in more than one state. This, of course, applies to P. Now consider P'. There exists a partition

P_i and a retained policy R_1 such that P' and R_1 differ in more than one state in P_i. (Otherwise, P' would belong to a discarded set.) Now consider a policy L_1 which is identical to R_1 in all states except those in P_i and identical to P' in P_i. L_1 differs with R_1 in more than one state in P_i, whence it does not belong to the set discarded by R_1. But BB did not yield us L_1.

Therefore, there must exist a retained policy R_2 defining a discarded set to which L_1 belongs. In partition P_i, L_1 and R_2, and therefore P' and R_2, differ in at most one state. Thus, there exists a partition P_j in which P' and R_2 differ in more than one state. Now consider policy L_2 identical to R_2 in all partitions except P_j and identical to P' in P_j. Thus, L_2 and P' are identical partitions P_i and P_j. BB did not yield this policy, whence it belongs to a discarded set defined by a retained policy R_3. Continuing in this manner, we finally form P' and find that it belongs to a subset over which we have maximized, whence its gain cannot exceed that already obtained. Thus, we have exhausted the feasible policy set.

D. Of Trapping States

In some problems, such as Howard's baseball example, there is one trapping state, whence all others are transient. In this case, all policies have the same gain. Howard shows, however, that his VD-PI algorithm improves the relative values every iteration [4]. Since we have retained Howard's sufficient condition $\gamma_i \geq 0$, this still applies. However, when we have a constraint-sensitive optimal, we are maximizing over subsets. Whenever we detect such a maximum, we compare its gain

with the current optimum. If the gains are equal, as they are here, we need a criterion for selecting an optimum. We adopt the criterion developed by Nesbitt [10] for such cases, where $\sum_{i=1}^{N=1} \beta_i v_i$ is optimized, β_i's being the given initial state probabilities. This maximizes the asymptotic expected reward.

E. Of Speeding Up the Algorithm

As mentioned earlier, a computationally burdensome approach to the constrained Markov Decision Processes problem would be to obtain the unconstrained optimum by Howard's VD-PI algorithm, then proceed backwards in the policy ordering, checking feasibility. Our algorithm attacks the problem directly by only considering feasible policies and exhausting the feasible policy set. We can also obtain constraint-indifferent optimum without having to exhaust the set. A moddle-of-the-ground approach would be to obtain the unconstrained optimum by VD-PI, then start our algorithm from there. The initial feasible policy is obtained by branch and bound with no restrictions on γ_i, from the table of test quantities rather than from the q_i^k. While we still would have to exhaust the feasible policy set for a constraint sensitive optimum, we have a better chance of starting at a policy having a high gain. While it is true that we perform extra VD's in the beginning, it is hoped that this will be offset by performing less BB iterations later on.

F. Baseball Example and Computational Results

All the above considerations were applied to Howard's baseball example [4] with constraints imposed on the policies, the constraints

being various bright ideas imposed on the manager by the club's eccentric owner.

Table 5.1 gives the results for a simple constraint:

$$d_2^1 + d_3^2 \leq 1 \tag{5.1}$$

This yields a constraint-indifferent optimal policy. Another simple constraint yields the results given in Table 5.2:

$$d_2^1 + d_3^1 \leq 1 \tag{5.2}$$

In both cases, we have only one partition.

Encouraged by the results his team achieves, the owner wants to try new ideas. The manager knows better but wants to keep his job, so he compromises. They agree that, irrespective of how many men are out, the owner's ideas are to be carried out only if there is a man on third base and the bases are not loaded. With no one out, the owner wants a hit decision in at least 2 of the 3 possible situations. Likewise, with one man out and two men out. This translates immediately to the following constraints:

$$d_5^1 + d_6^1 + d_7^1 \geq 2 \tag{5.3}$$

$$d_{13}^1 + d_{14}^1 + d_{15}^1 \geq 2 \tag{5.4}$$

$$d_{21}^1 + d_{22}^1 + d_{23}^1 \geq 2 \tag{5.5}$$

Table 5.1

BASEBALL PROBLEM
(Transient--One Constraint)

Number of policy constraints	= 1
Number of feasible policies	= 2.2×10^{10}
Number of iterations	= 2
Optimal gain	= 0.0
Optimal policy type	= constraint-indifferent

State	Decision	Value v_i
1	Hit	0.81
2	Hit	1.24
3	Hit	1.32
4	Hit	1.88
5	Hit	1.56
6	Hit	2.07
7	Hit	2.16
8	Hit	2.73
9	Hit	0.45
10	Hit	0.77
11	Hit	0.87
12	Hit	1.23
13	Hit	1.10
14	Hit	1.44
15	Hit	1.53
16	Hit	1.95
17	Hit	0.17
18	Hit	0.33
19	Hit	0.39
20	Hit	0.58
21	Hit	0.50
22	Hit	0.67
23	Hit	0.73
24	Hit	0.98
25	Trapped	0.0

Table 5.2

BASEBALL PROBLEM
(Transient--One Constraint)

Number of policy constraints = 1

Number of feasible policies = 2.2×10^{10}

Number of iterations = 5

Optimal gain = 0.0

Optimal policy type = constraint-sensitive

State	Decision	Value v_i
1	Hit	0.78
2	Hit	1.24
3	Bunt	1.01
4	Hit	1.88
5	Hit	1.53
6	Hit	2.06
7	Hit	2.14
8	Hit	2.73
9	Hit	0.45
10	Hit	0.77
11	Hit	0.87
12	Hit	1.23
13	Hit	1.10
14	Hit	1.44
15	Hit	1.53
16	Hit	1.95
17	Hit	0.17
18	Hit	0.33
19	Hit	0.39
20	Hit	0.58
21	Hit	0.50
22	Hit	0.67
23	Hit	0.73
24	Hit	0.98
25	Trapped	0.0

That is not the whole story, however. The owner has additional bright ideas. If the manager decides to hit with a man on first base, then he has to hit with a man on second. The corresponding constraints are:

$$d_6^1 + d_7^2 \leq 1 \tag{5.6}$$

$$d_6^1 + d_7^3 \leq 1 \tag{5.7}$$

$$d_{14}^1 + d_{15}^2 \leq 1 \tag{5.8}$$

$$d_{14}^1 + d_{15}^3 \leq 1 \tag{5.9}$$

$$d_{22}^1 + d_{23}^2 \leq 1 \tag{5.10}$$

$$d_{22}^1 + d_{23}^3 \leq 1 \tag{5.11}$$

Finally, the manager cannot decide to hit with a man on second base unless he decides the same with one on first and second. This translates into:

$$d_5^2 + d_7^1 \leq 1 \tag{5.12}$$

$$d_5^3 + d_7^1 \leq 1 \tag{5.13}$$

$$d_{13}^2 + d_{15}^1 \leq 1 \tag{5.14}$$

$$d_{13}^3 + d_{15}^1 \leq 1 \tag{5.15}$$

$$d_{21}^2 + d_{23}^1 \leq 1 \tag{5.16}$$

$$d_{21}^3 + d_{23}^1 \leq 1 \tag{5.17}$$

Constraints (5.3) through (5.17) decompose into three natural partitions corresponding to how many men are out. Table 5.3 gives the results under these conditions. Note that the optimal is constraint-indifferent. In other words, all the fuss the owner made is really irrelevant. The manager does exactly what he would have done without any interference from the owner. However, neither of them realizes that, and the team continues to win, which makes the owner come up with even more ideas. The manager salvages freedom of action only if the bases are loaded or nobody is on. In addition to the previous, the owner imposes restrictions whenever nobody is on third. He wants the decision to be a steal in at least two of every three possible situations. This leads to:

$$d_2^3 + d_3^3 + d_4^3 \geq 2 \tag{5.18}$$

$$d_{10}^3 + d_{11}^3 + d_{12}^3 \geq 2 \tag{5.19}$$

$$d_{18}^3 + d_{19}^3 + d_{20}^3 \geq 2 \tag{5.20}$$

Moreover, if he decides to steal second with one man on, he cannot hit or bunt (i.e., must steal third) with two men on:

$$d_2^3 + d_4^1 \leq 1 \tag{5.21}$$

$$d_2^3 + d_4^2 \leq 1 \tag{5.22}$$

$$d_{10}^3 + d_{12}^1 \leq 1 \tag{5.23}$$

$$d_{10}^3 + d_{12}^2 \leq 1 \tag{5.24}$$

Table 5.3

BASEBALL PROBLEM
(Transient--Fifteen Constraints)

Number of policy constraints	= 15
Number of feasible policies	= 3.4×10^7
Number of iterations	= 2
Optimal gain	= 0.0
Optimal policy type	= constraint-indifferent

State	Decision	Value v_i
1	Hit	0.81
2	Hit	1.24
3	Hit	1.32
4	Hit	1.88
5	Hit	1.56
6	Hit	2.07
7	Hit	2.16
8	Hit	2.73
9	Hit	0.45
10	Hit	0.77
11	Hit	0.87
12	Hit	1.23
13	Hit	1.10
14	Hit	1.44
15	Hit	1.53
16	Hit	1.95
17	Hit	0.17
18	Hit	0.33
19	Hit	0.39
20	Hit	0.58
21	Hit	0.50
22	Hit	0.67
23	Hit	0.73
24	Hit	0.98
25	Trapped	0.0

$$d_{18}^{3} + d_{20}^{1} \leq 1 \tag{5.25}$$

$$d_{18}^{3} + d_{20}^{2} \leq 1 \tag{5.26}$$

With less than two men out, he cannot decide to steal with two men on if he decides to hit or bunt with a man on second:

$$d_{3}^{1} + d_{4}^{3} \leq 1 \tag{5.27}$$

$$d_{3}^{2} + d_{4}^{3} \leq 1 \tag{5.28}$$

$$d_{11}^{1} + d_{12}^{3} \leq 1 \tag{5.29}$$

$$d_{11}^{2} + d_{12}^{3} \leq 1 \tag{5.30}$$

With two men out, the rule changes to not steal with two men on if hit or bunt with a man on first:

$$d_{18}^{1} + d_{20}^{3} \leq 1 \tag{5.31}$$

$$d_{18}^{2} + d_{20}^{3} \leq 1 \tag{5.32}$$

Table 5.4 gives the results of the problem subject to constraints (5.3) through (5.32).

In all the previous, we retained the original structure of the problem, namely a single trapping state (state 25, three men out). To select among policies, we used an initial state probability distribution $\beta_1 = 1$, $\beta_i = 0$ for $i \neq 1$. This means always starting in state 1 (no men out, no men on). The same problems were run with $\beta_i = 1/24$ (equally likely

Table 5.4

BASEBALL PROBLEM
(Transient--Thirty Constraints)

Number of policy constraints = 30

Number of feasible policies = 4.6×10^4

Number of iterations = 2

Optimal gain = 0.0

Optimal policy type = constraint-sensitive

State	Decision	Value v_i
1	Hit	0.62
2	Hit	0.93
3	Steal 3	0.87
4	Steal 3	1.20
5	Hit	1.39
6	Hit	1.83
7	Hit	2.01
8	Hit	2.60
9	Hit	0.35
10	Hit	0.56
11	Steal 3	0.57
12	Steal 3	0.75
13	Hit	1.01
14	Hit	1.32
15	Hit	1.47
16	Hit	1.89
17	Hit	0.12
18	Steal 2	0.17
19	Hit	0.35
20	Steal 3	0.31
21	Hit	0.47
22	Hit	0.63
23	Hit	0.72
24	Hit	0.98
25	Trapped	0.0

to start anywhere), and the results were identical. To test the algorithm on a problem with many states that are recurrent, we changed the P_{ij} of the trapping state. We made state 25 return to state 1 (i.e., a new inning) with probability 1. Tables 5.5, 5.6, and 5.7 give the results of the recurrent problems subject to constraints (5.2), (5.3) through (5.17), and (5.3) through (5.32), respectively. Finally, we ran the algorithm in the manner described in Section E for both the recurrent and transient baseball problems, and a slight improvement in execution time was noticed. Tables 5.8 and 5.9 give the results of the two problems, respectively.

The computational results obtained indicate to us that we have a computationally efficient algorithm for Markov Decision Processes with constraints when the number of states is large.

Table 5.5

BASEBALL PROBLEM
(Recurrent--One Constraint)

Number of policy constraints = 1

Number of feasible policies = 2.2×10^{10}

Number of iterations = 4

Optimal gain = 0.1373

Optimal policy type = constraint-sensitive

State	Decision	Value v_i
1	Hit	0.13
2	Hit	0.62
3	Bunt	0.45
4	Hit	1.26
5	Hit	0.91
6	Hit	1.44
7	Hit	1.52
8	Hit	2.11
9	Hit	0.01
10	Hit	0.35
11	Hit	0.43
12	Hit	0.81
13	Hit	0.68
14	Hit	1.02
15	Hit	1.11
16	Hit	1.53
17	Hit	-0.05
18	Hit	0.11
19	Hit	0.17
20	Hit	0.36
21	Hit	0.27
22	Hit	0.45
23	Hit	0.51
24	Hit	0.76
25	New Inning	0.0

Table 5.6

BASEBALL PROBLEM
(Recurrent--Fifteen Constraints)

Number of policy constraints = 15

Number of feasible policies = 3.4×10^7

Number of iterations = 2

Optimal gain = 0.1406

Optimal policy type = constraint-indifferent

State	Decision	Value v_i
1	Hit	0.14
2	Hit	0.61
3	Hit	0.68
4	Hit	1.24
5	Hit	0.91
6	Hit	1.43
7	Hit	1.52
8	Hit	2.09
9	Hit	0.001
10	Hit	0.34
11	Hit	0.42
12	Hit	0.80
13	Hit	0.67
14	Hit	1.01
15	Hit	1.10
16	Hit	1.32
17	Hit	-0.06
18	Hit	0.10
19	Hit	0.16
20	Hit	0.35
21	Hit	0.27
22	Hit	0.44
23	Hit	0.50
24	Hit	0.75
25	New Inning	0.0

Table 5.7

BASEBALL PROBLEM
(Recurrent--Thirty Constraints)

Number of policy constraints = 30

Number of feasible policies = 4.6×10^4

Number of iterations = 2

Optimal gain = 0.1017

Optimal policy type = constraint-sensitive

State	Decision	Value v_i
1	Hit	0.10
2	Hit	0.42
3	Steal 3	0.31
4	Steal 3	0.64
5	Hit	0.89
6	Hit	1.33
7	Hit	1.52
8	Hit	2.11
9	Hit	-0.004
10	Hit	0.21
11	Steal 3	0.18
12	Steal 3	0.34
13	Hit	0.68
14	Hit	0.98
15	Hit	1.14
16	Hit	1.57
17	Hit	-0.05
18	Steal 2	-0.03
19	Hit	0.18
20	Steal 3	0.10
21	Hit	0.29
22	Hit	0.45
23	Hit	0.55
24	Hit	0.81
25	New Inning	0.0

Table 5.8

SPEEDING UP THE ALGORITHM--BASEBALL PROBLEM
(Recurrent--Thirty Constraints)

Number of policy constraints = 30

Number of feasible policies = 4.6×10^4

Number of iterations = 3

Optimal gain = 0.1017

Optimal policy type = constraint-sensitive

State	Decision	Value v_i
1	Hit	0.10
2	Hit	0.42
3	Steal 3	0.31
4	Steal 3	0.64
5	Hit	0.89
6	Hit	1.33
7	Hit	1.52
8	Hit	2.11
9	Hit	-0.004
10	Hit	0.21
11	Steal 3	0.18
12	Steal 3	0.34
13	Hit	0.68
14	Hit	0.98
15	Hit	1.14
16	Hit	1.57
17	Hit	-0.05
18	Steal 2	-0.03
19	Hit	0.18
20	Steal 3	0.10
21	Hit	0.29
22	Hit	0.45
23	Hit	0.55
24	Hit	0.81
25	New Inning	0.0

Table 5.9

SPEEDING UP THE ALGORITHM--BASEBALL PROBLEM
(Transient--Thirty Constraints)

Number of policy constraints = 30

Number of feasible policies = 4.6×10^4

Number of iterations = 3

Optimal gain = 0.0

Optimal policy type = constraint-sensitive

State	Decision	Value v_i
1	Hit	0.62
2	Hit	0.93
3	Steal 3	0.87
4	Steal 3	1.20
5	Hit	1.39
6	Hit	1.83
7	Hit	2.01
8	Hit	2.60
9	Hit	0.35
10	Hit	0.56
11	Steal 3	0.57
12	Steal 3	0.75
13	Hit	1.01
14	Hit	1.32
15	Hit	1.47
16	Hit	1.89
17	Hit	0.12
18	Steal 2	0.17
19	Hit	0.35
20	Steal 3	0.31
21	Hit	0.47
22	Hit	0.63
23	Hit	0.72
24	Hit	0.98
25	Trapped	0.0

Chapter VI

CONCLUSIONS AND SUGGESTIONS FOR FUTURE RESEARCH

An important limitation of the Markov Decision Process as a model for practical problems has been overcome. The ability to deal with policy constraints extends the applicability of the model, as we saw in the taxicab and baseball examples. Although the policy constraints there were immediately translated into algebraic form, we also have the ability to express complicated constraints via the algebra of events. We showed, however, that the resultant constraints might not be of the simplest form possible. An area worthy of future research would be to try and devise a procedure which yields simple algebraic expressions. The algorithm we developed could be used to order the policies according to gain by successively making the optimal policy infeasible. This would involve more computational effort than Nesbitt's [10] procedure. However, it orders risk-sensitive policies for which no method for ordering has yet been devised. An interesting research would be to seek a unified approach to both the ordering and the policy constraint problems.

Another area worthy of further research is sensitivity analysis. As we pointed out, the values of the constraints are interdependent. To discover which constraints, or group of constraints, are responsible for constraint-sensitivity of the optimal policy, it seems fruitless to try using our algorithm repeatedly in an effort to exhaust all possible constraint combinations. The mere bookkeeping required is mind-boggling. Investigating the structural interaction between the coupled states during maximization of the Lagrangian would probably be a better approach.

REFERENCES

1. H. Everett, III, "Generalized Lagrange Multiplier Method for Solving Problems of Optimum Allocation of Resources," Operations Research, Vol. 11, No. 3, May-June 1963, pp. 101-111.

2. W. C. Healy, Jr., "Multiple Choice Programming," Operations Research, Vol. 12, 1964, pp. 122-138.

3. F. S. Hillier and G. J. Liebermann, Introduction to Operations Research, Holdenday, San Francisco, 1967.

4. R. A. Howard, Dynamic Programming and Markov Processes, The M.I.T. Press, Cambridge, 1960.

5. R. A. Howard, Dynamic Probabilistic Systems, 2 volumes, Wiley, New York, 1971.

6. R. A. Howard and J. E. Matheson, "Risk Sensitive Markov Decision Processes," Management Science, Vol. 18, No. 7, March 1972, pp. 356-369.

7. E. L. Lawler and D. E. Wood, "Branch-and-Bound Methods: A Survey," Operations Research, Vol. 14, 1966, pp. 669-719.

8. D. G. Luenberger, Introduction to Linear and Non-Linear Programming, Addison-Wesley, Reading, Mass., 1973.

9. Mine and Osaki, Markovian Decision Processes, American Elsevier Publishing Co., New York, 1970.

10. D. M. Nesbitt, "Policy Ordering in Semi-Markov Decision Processes," Ph.D. Dissertation, Engineering-Economic Systems Department, Stanford University, March 1975.

DISTRIBUTION LIST

Director (2 copies)
Advanced Research Projects
Agency
Attention: Program Management
1400 Wilson Boulevard
Arlington, VA 22209

Administrator, Defense
Documentation Center
(12 copies)
Attention: DDT-TC
Cameron Station
Alexandria, VA 22314

Office of Naval Research
(3 copies)
Attention: Code 455
800 North Quincy Street
Arlington, VA 22217

Defense Contract Administrative
Services Management Area
Attention: Mr. K. Gerasim
300 Joppa Road
Towson, MD 21204

Director (6 copies)
Naval Research Laboratory
Attention: Code 2627
Washington, DC 20375

Director (6 copies)
Naval Research Laboratory
Attention: Code 2629
Washington, DC 20375

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: EES-DA-76-3	2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE (and Subtitle): "Markov Decision Processes with Policy Constraints"	5. TYPE OF REPORT & PERIOD COVERED: Technical rept.
6. PERFORMING ORG. REPORT NUMBER: EES DA-76-3	
7. AUTHOR(s): John Nafeh	8. CONTRACT OR GRANT NUMBER(s): 75-030-0713 / NSF-GK-36491
9. PERFORMING ORGANIZATION NAME AND ADDRESS: The Board of Trustees of the Leland Stanford Junior University, c/o Office of Research Administrator, Encina Hall, Stanford, California 94305	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS: Advanced Research Projects Agency, Human Resources Research Office, Arlington, Va. 22209 (Sub-contract of Decisions & Designs, Inc., McLean, Va. 22101)	12. REPORT DATE: April 1976
	13. NUMBER OF PAGES: 170 (172 pp)
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office): Engineering Psychology Programs, Code 455, Office of Naval Research, 800 No. Quincy Street, Arlington, Va. 2217	15. SECURITY CLASS. (of this report): Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

OPTIMAL POLICY	RISK-SENSITIVE	CONSTRAINT-INDIFFERENT
RISK-INDIFFERENT	POLICY CONSTRAINTS	CONSTRAINT-SENSITIVE

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This work is concerned with Markov Decision Processes with policy constraints. The selection of an optimum stationary policy for such processes, in the absence of policy constraints, is a problem which has received a great deal of attention, and has been satisfactorily solved. Relatively little attention has been given to the case when policy constraints are present → next page

(continued)

408 566

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

Block 20 (Continued)

or to the formulation of such constraints. Optimum policy sensitivity analysis is also a subject in which little has been achieved.

Towards those ends, this work makes three major contributions. First, policy constraints are formulated and categorized. Secondly, a computationally efficient iterative algorithm is developed for selecting the optimum policy for completely ergodic, infinite time horizon Markov Decision Processes with policy constraints for both the risk-indifferent and risk-sensitive cases. Finally, the sensitivity of optimum policies to the policy constraints is analyzed by using the algorithm to compute the value of removing a constraint or a group of constraints.