A STOPPING RULE FOR FORECAST HORIZONS IN NONHOMOGENEOUS MARKOV DECISION PROCESSES

James C. Bean, Wallace J. Hopp,
and Izak Duenyas

Technical Report 89-21

Department of Industrial and Operations Engineering
University of Michigan, Ann Arbor, Michigan 48109-2117

June 1989
Revised June 1991

This work was supported in part by the National Science Foundation under Grants ECS-8700836, DDM-9018515 and ECS-8619732

A Stopping Rule for Forecast Horizons in Nonhomogeneous Markov Decision Processes [*]

James C. Bean
Department of Industrial and Operations Engineering
The University of Michigan
Ann Arbor, MI 48109-2117

Wallace J. Hopp
Department of Industrial Engineering and Management Sciences
Northwestern University
Evanston, IL 60208

Izak Duenyas
Department of Industrial and Operations Engineering
The University of Michigan
Ann Arbor, MI 48109-2117

Abstract

We formulate a mixed integer program to determine whether a finite time horizon is a forecast horizon in a nonhomogeneous Markov decision process. We give a Bender's decomposition approach to solving this problem that evaluates the stopping rule, eliminates some suboptimal combinations of actions and yields bounds on the maximum error that could result from the selection of a candidate action in the initial stage. The integer program arising from the decomposition has special properties allowing efficient solution. We illustrate the approach with numerical examples.

[*]This work was supported in part by the National Science Foundation under Grants ECS-8700836 and DDM-9018515 to the University of Michigan and ECS-8619732 to Northwestern University.

1. Introduction

Many important problems can be modelled as nonhomogeneous Markov decision processes including production/inventory, equipment replacement, capacity expansion and R&D planning (see Hopp, Bean and Smith (1987)). It is well known that any finite state nonhomogeneous Markov decision process can be transformed into a countable state homogeneous Markov decision process. However, since the resulting formulation has countably infinite states, traditional solution techniques, such as value or policy iteration, may not be effective.

Over a finite time horizon, a first period decision rule is optimal if it gives the decision, contingent on state, that maximizes expected value over the horizon. Over the infinite horizon, in the undiscounted case, all solutions may have infinite expected cost. The literature discusses several alternate optimality criteria for this case including average optimality, weak optimality and algorithmic optimality (see Flynn (1980)). In this paper we use *algorithmic optimality* as our criterion. See Hopp, Bean and Smith (where this concept is called *periodic forecast horizon optimal*) for a discussion of this criterion. An infinite horizon strategy is algorithmically optimal if there exists a subsequence of times such that the finite horizon optima to those times converges to this strategy in the metric of Hopp, Bean and Smith. For example, a first period policy is algorithmically optimal if it is optimal to an infinite number of finite horizon problems. In the discounted case algorithmic optimality agrees with standard discounted optimality (maximum expected present value).

The basis for this line of research is early work on the relationship between ergodicity and infinite horizon value such as Hinderer and Hübner [1974] and Morton and Wecker [1977]. More recent research has examined the relationship between ergodicity and solutions to propose algorithms for the finite state nonhomogeneous Markov decision process (see Bès and Lasserre (1986), Bès and Sethi (1988), Hopp (1989) and Bean, Smith and Lasserre (1990)). All attempt to identify a finite time, called a *forecast horizon*, sufficiently long that the initial decision rule for the finite horizon problem is known to be optimal in the infinite horizon problem, regardless of the data beyond the horizon. Each of these approaches solves finite horizon problems of increasing horizon until a stopping rule determines that a forecast horizon has been discovered. None of these approaches has been shown to identify short forecast horizons and be computationally tractable. Under special conditions, Tseng (1990) finds horizons a priori by use of contraction mappings.

In this paper we propose a stopping rule based on integer programming that has promise for solving realistically sized problems. It evaluates the stopping rule more effectively than the nonlinear programming and piecewise linearity approaches in Hopp, and finds a shorter stopping time than the tail value approaches in Bès and Lasserre; Bès and Sethi; and Bean, Smith and Lasserre.

Our target problem is the nonhomogeneous Markov decision process (NMDP) with finite state space $S = \{1, \ldots, n\}$ and finite action spaces, $A_k(i)$, in each state, $i \in S$, of stage $k = 0, 1, \ldots$. We denote the one period reward in state i of stage k under action $a \in A_k(i)$ by $r_k(i,a)$ and the one step transition probabilities from state i in stage k to state j in stage $k+1$ under action a as $p_k^{ij}(a)$. Future values are discounted with factor $0 < \alpha \leq 1$. Assume that $|r_k(i,a)| \leq \bar{R} \in \Re$ for all (k,i,a). Note that, since adding a constant to all one period rewards does not alter the optimal solution, the assumption that $\bar{R} < \infty$ allows us also to assume WLOG that $r_k(i,a) \geq 0$ for all k, i, a. This assumption simplifies later exposition.

The remainder of the paper is organized as follows. Section 2 details the integer programming stopping rule. Section 3 describes a modified Bender's decomposition to separate the mixed integer programs into pure linear programs and pure integer programs. Section 4 establishes action elimination results that reduce computation. Section 5 presents a tree search algorithm for solving the pure integer programs. Performance of the approach is discussed in Section 6. Finally, Section 7 contains conclusions and further research.

2. IP Stopping Rule

If we look at an N period truncation of the NMDP with salvage vector L, and let $v_k^N(i,L)$ represent the optimal value function in state i of stage k, the dynamic programming recursion can be written as

$$\begin{aligned} v_k^N(i,L) &= \max_{a \in A_k(i)} r_k(i,a) + \alpha \sum_{j \in S} p_k^{ij}(a) v_{k+1}^N(j,L), \quad i \in S; k = 0, \ldots, N \\ v_{N+1}^N(i,L) &= L(i), \quad i \in S. \end{aligned}$$

Let $\pi_k^N(i,L), k = 0, \ldots, N; i \in S$, represent the actions that achieve the above maximizations. Note that the assumption that $r_k(i,a) \geq 0$ for all k, i, a ensures that $v_k^N(i,L) \geq 0$ provided that all $L(i) \geq 0$. Hopp shows that an optimal solution to the infinite horizon problem will be returned if we let L equal the period $N+1$ *relative value function* (also referred to as *coherent value function* in Bean, Smith and Lasserre). That is, let $L(i) = \bar{v}_{N+1}(i)$, where

$$\bar{v}_k(i) = \lim_{N \to \infty} \bar{v}_k^N(i)$$

and

$$\bar{v}_k^N(i) \equiv v_k^N(i, \mathbf{O}) - v_k^N(0, \mathbf{O})$$

for an arbitrary reference state 0 and salvage vector of all zeroes, $\mathbf{O}$. These limits are shown to exist in Hopp.

We cannot compute this salvage function because we cannot finitely solve the infinite horizon problem. As shown in Hopp, under certain ergodic conditions, we can bound the range of potential values of the relative value function. If the first decision is the same for all salvage vectors within this range, we are sure to have hit upon the correct vector. This is the idea behind the following stopping rule.

Definition 1 (Stopping Rule (Hopp)) *Stop at the current horizon, N, if $\pi_0^N(i,L) = \bar{a}$ for all $L \in \Lambda$, where*

$$\begin{aligned}
\Lambda &= \{x \in \Re^n : x \geq 0, \|x\| \leq M, x_n = 0\} \\
M &= \frac{\bar{r}}{(1-\alpha a_0)} \\
a_0 &= \sup_k \max_{i,j \in S} \max_{a_1 \in A_k(i), a_2 \in A_k(j)} \sum_{s \in S} |p_k^{is}(a_1) - p_k^{js}(a_2)|/2 \leq 1 \\
\|x\| &= \max_{i,j} [x_i - x_j] \\
\bar{r} &= \sup_k \max_{i,j \in S} \max_{a_1 \in A_k(i), a_2 \in A_k(j)} [r_k(i,a_1) - r_k(j,a_2)]
\end{aligned}$$

Note that the stopping rule is well defined only if $\alpha a_0 < 1$.

Theorem 1 (Hopp) *If $\alpha a_0 < 1$ and there exists a unique decision, $\bar{a}$, such that*

$$r_0(i,\bar{a}) + \alpha \sum_{j \in S} p_0^{ij}(\bar{a}) \bar{v}_1(j) \geq r_0(i,a) + \alpha \sum_{j \in S} p_0^{ij}(a) \bar{v}_1(j)$$

for $i \in S$, then the stopping rule will eventually be satisfied at some finite N. Further, such an N is a forecast horizon. If the stopping rule is satisfied at some N, then N is a forecast horizon regardless of optimal action uniqueness.

Note that the hypothesis of Theorem 1 requires knowledge of data over the infinite horizon. However, the implementation will require only data up to some finite horizon, N.

To test whether a time horizon, N, satisfies this stopping rule, we must determine whether a single action is optimal for the N period dynamic program for each salvage vector in the convex set Λ. Since Λ includes the origin, we can find a candidate action by solving the finite horizon dynamic program with zero salvage values. The challenge is to see whether it is optimal for all other salvage vectors in Λ.

To develop an algorithmic test of the above stopping rule, first observe that the N period dynamic program, with salvage vector L, can be formulated as the linear program

$$\begin{array}{rcl} \min & & \sum_{i=1}^{n} v_0^N(i,L) \\ \text{subject to} & & \\ v_k^N(i,L) & \geq & r_k(i,a) + \alpha \sum_{j\in S} p_k^{ij}(a) v_{k+1}^N(j,L), \quad i \in S; k = 0,\ldots,N-1; a \in A_k(i) \\ v_N^N(i,L) & \geq & r_N(i,a) + \alpha \sum_{j\in S} p_N^{ij}(a) L(j), \quad i \in S; a \in A_N(i). \end{array}$$

Note that for each (k,i) pair, at least one of the constraints for $a \in A_k(i)$ must be tight in order for the $v_k^N(i,L)$ values to be a solution to the dynamic program. This is forced in the linear program by the objective.

For a given initial state, $\bar{i}$, suppose we have solved the N period problem to determine a candidate optimal action $\bar{a} = \pi_0^N(\bar{i}, \mathbf{O})$. Let $v_0^N(\bar{i}, L; \bar{a})$ represent the value function (assuming salvage vector L) when the initial action in state $\bar{i}$ is forced to be $\bar{a}$. To formulate a stopping rule, we would like to vary the $L(j)$ values and search for an $L \in \Lambda$ that makes $\pi_0^N(\bar{i}, L) = a \neq \bar{a}$. If no such a can be found then the stopping rule is satisfied. Altering the objective and freeing L gives the following linear program. Note that $v_k^N(i,\Lambda)$ is a value function with L varying over Λ.

$$\begin{array}{rcl} \min & & v_0^N(\bar{i},\Lambda;\bar{a}) - v_0^N(\bar{i},\Lambda) \\ \text{subject to} & & \\ v_0^N(\bar{i},\Lambda;\bar{a}) & = & r_k(\bar{i},\bar{a}) + \alpha \sum_{j\in S} p_0^{\bar{i}j}(\bar{a}) v_1^N(j,\Lambda) \\ v_0^N(\bar{i},\Lambda) & \geq & r_k(\bar{i},a) + \alpha \sum_{j\in S} p_0^{\bar{i}j}(a) v_1^N(j,\Lambda), \quad a \in A_0(i) \\ v_k^N(i,\Lambda) & \geq & r_k(i,a) + \alpha \sum_{j\in S} p_k^{ij}(a) v_{k+1}^N(j,\Lambda), \quad i \in S; k = 1,\ldots,N-1; a \in A_k(i) \\ v_N^N(i,\Lambda) & \geq & r_N(i,a) + \alpha \sum_{j\in S} p_N^{ij}(a) L(j), \quad i \in S; a \in A_N(i) \\ L & \in & \Lambda. \end{array}$$

By the definition of Λ, the condition $L \in \Lambda$ can be expressed as a set of linear inequalities.

This linear program may not return a valid solution to the dynamic program. It is possible for the optimal solution to this linear program to be slack for all constraints associated with a specific stage and state. In this case no decision is chosen for that dynamic programming state. To ensure that one constraint from each $A_k(i)$ set be tight, we introduce $0-1$ integer variables.

Let $\bar{a}$ be the candidate decision found by solving the dynamic program with zero salvage values at stage 0 and known state $\bar{i}$. To see if $\bar{a}$ is optimal for all $L \in \Lambda$, we can solve the following mixed integer program:

$$\begin{array}{rcl}
\min & & v_0^N(\bar{i},\Lambda;\bar{a}) - v_0^N(\bar{i},\Lambda) \qquad\qquad (\text{MIP}) \\
\text{subject to} & & \\
v_0^N(\bar{i},\Lambda;\bar{a}) & = & r_0(\bar{i},\bar{a}) + \alpha \sum_{j\in S} p_0^{\bar{i}j}(\bar{a}) v_1^N(j,\Lambda) \\
v_0^N(\bar{i},\Lambda) & \geq & r_k(\bar{i},a) + \alpha \sum_{j\in S} p_0^{\bar{i}j}(a) v_1^N(j,\Lambda), \quad a \in A_0(i) \\
v_k^N(i,\Lambda) & \geq & r_k(i,a) + \alpha \sum_{j\in S} p_k^{ij}(a) v_{k+1}^N(j,\Lambda), \\
 & & i \in S; a \in A_k(i); k = 1,\ldots,N-1 \\
v_N^N(i,\Lambda) & \geq & r_N(i,a) + \alpha \sum_{j\in S} p_N^{ij}(a) L(j), \quad i \in S; a \in A_N(i) \\
v_0^N(\bar{i},\Lambda) & \leq & r_k(\bar{i},a) + \alpha \sum_{j\in S} p_0^{\bar{i}j}(a) v_1^N(j,\Lambda) + (1 - y_k(\bar{i},a))H, \quad a \in A_0(\bar{i}) \\
v_k^N(i,\Lambda) & \leq & r_k(i,a) + \alpha \sum_{j\in S} p_k^{ij}(a) v_{k+1}^N(j,\Lambda) + (1 - y_k(i,a))H, \\
 & & i \in S; a \in A_k(i); k = 1,\ldots,N-1 \\
v_N^N(i,\Lambda) & \leq & r_N(i,a) + \alpha \sum_{j\in S} p_N^{ij}(a) L(j) + (1 - y_N(i,a))H, \\
 & & i \in S; a \in A_N(i) \\
\sum_{a\in A_k(i)} y_k(i,a) & = & 1, \quad i \in S; k = 1,\ldots,N \\
y_0(\bar{i},\bar{a}) & = & 0 \\
L & \in & \Lambda \\
y_k(i,a) & \in & \{0,1\}, \quad i \in S; a \in A_k(i); k = 0,\ldots,N
\end{array}$$

where H is a large number. Note that H forces the selection of one solution for each state much in the way we transform disjunctive constraints to conjunctive constraints. Any $H > \sum_{k=0}^{N} \alpha^k \bar{R}$ will lead to an optimal solution since this is an upper bound on any value function in this problem. For any $H > \sum_{k=0}^{N} \alpha^k \bar{R}$, if $y_k(i,a) = 0$, H is large enough to ensure that the constraint is satisfied for any solution. In effect, the constraint is imposed only when $y_k(i,a) = 1$.

If $y_k(i,a) = 1$ then the corresponding constraint in (MIP) forces the value for that state to equal the value from choosing that action. The constraints corresponding to all other $a \in A_k(i)$ are reduced to the traditional inequalities. In any feasible solution, exactly one $y_k(i,a)$ will equal 1 for each (i,k) pair leading to a valid solution to the dynamic program. The constraint $y_0(\bar{i},\bar{a}) = 0$

forces (MIP) to look for some solution other than $\bar{a}$ and speeds computation. Recall that WLOG we can add the constraints $v_k^N(i, L) \geq 0$, $i \in S$, $k = 0, \ldots, N$.

This formulation yields the following results.

Theorem 2 *Under the hypotheses of Theorem 1, if the optimal objective in (MIP) is nonnegative, then (a) no $a \in A_0(\bar{i})$ is superior to $\bar{a}$ for any $L \in \Lambda$ and (b) the current horizon, N, is a forecast horizon so that $\bar{a}$ is optimal over the infinite horizon for any data beyond N.*

Proof: From Hopp, we know that the relative value function covering time $N + 1$ to infinity is some vector in Λ. From the formulation, a nonnegative objective means that $\bar{a}$ is optimal for all $L \in \Lambda$. ∎

Theorem 3 *Under the hypotheses of Theorem 1, if the objective value in (MIP) is strictly negative, then (a) the decision, a, for which $y_0(\bar{i}, a) = 1$ is better than $\bar{a}$ for some $L \in \Lambda$ and (b) the negative of the objective value represents the maximum loss, over the infinite horizon, from choosing $\bar{a}$.*

Proof: If $\bar{a}$ is not optimal, the loss from choosing it, over the infinite horizon, can be expressed

$$\lim_{N \to \infty} [v_0^N(\bar{i}, \mathbf{O}) - v_0^N(\bar{i}, \mathbf{O}; \bar{a})].$$

Hopp, Bean and Smith show that this limit exists. Hopp went on to show that this is the same as

$$v_0^N(\bar{i}, \bar{v}_{N+1}) - v_0^N(\bar{i}, \bar{v}_{N+1}; \bar{a})$$

which is bounded above by

$$\max_{L \in \Lambda} \left[v_0^N(\bar{i}, L) - v_0^N(\bar{i}, L; \bar{a}) \right]$$

since $\bar{v}_{N+1} \in \Lambda$. This last expression is the negative of that returned by (MIP), and bounds the absolute loss. ∎

If the optimal objective value is negative, but small in absolute value, the decision maker may choose to stop and select action $\bar{a}$ as approximately optimal in state $\bar{i}$. If not, we must increase N, recompute the optimal policy for the N period problem with zero salvage values, and reevaluate the stopping rule.

3. Evaluating the Stopping Rule

For ease of exposition, restate the problem (MIP) as:

$$\begin{aligned} \min \quad & dv & \text{(MIP)} \\ \text{subject to} \quad & & \\ Dv &\geq b - Ay & \\ 0 &= e - Ey & \\ v &\geq 0 & \\ y &\geq 0, \quad \text{integer,} & \end{aligned}$$

where e is a vector of ones, v includes the $v's$ and $L's$ from the previous representation of (MIP), $0 = e - Ey$ are the constraints $\sum_{a \in A_k(i)} y_k(i,a) = 1$, and $Dv \geq b - Ay$ are all other constraints (note that $Ey = e$ implies that $y \leq e$). We make use of the nonnegativity constraints on v that were included WLOG to make following derivations more standard.

A standard Bender's treatment fixes the integer variables, y, resulting in a linear program, (LP), and constructs its dual, (DL). The feasible region of (DL) is independent of the choice of y. Hence, for any choice of y, the optimal solution to (DL) is determined by one of the finite, constant set of extreme points and rays of this feasible region. If the y chosen dictates an extreme ray, that is, an unbounded solution, then (LP) is infeasible. Since we seek a y leading to a feasible solution, we will not choose such values of y. For a feasible y, the optimal solution is the best extreme point evaluated at this y. The problem of choosing the best extreme point while constraining y from choosing an extreme ray can be formulated as a pure integer program, (I). For further details on the process, see Salkin (1975).

In any reasonably sized problem, the number of extreme points and rays is large, making this direct approach impractical. The power of Bender's approach is that we need only generate a few extreme points and rays, those near the optimal solution.

Beginning with no extreme points or rays, a candidate y is substituted into (DL), which is solved. It determines an extreme point or ray which is appended to (I). Then (I) generates a new y. The process repeats until (DL) and (I) converge in value. The computational efficiency of the procedure depends on the ease with which (I) can be solved.

The special structure of the multiple choice constraints, $Ey = e$, allows a more efficient version of the decomposition. Since they are equality constraints and do not include v variables, we can determine their effect on the Bender's process analytically. If any y computed by (I) does not satisfy $Ey = e$, we can drive the objective of (DL) to infinity by driving the multiplier for the violated constraint to plus or minus infinity and setting all other multipliers to 0. Thus, (DL) would pass an extreme ray to (I) equivalent to appending the violated constraint in $Ey = e$ to (I). We can preempt this process by appending the constraints $Ey = e$ to (I) initially.

This technique accomplishes three desirable effects. First, it avoids the iterations of the algorithm that would ordinarily generate the multiple choice constraints in (I). Second, it provides structure to the integer program allowing simpler computation. The resulting problem becomes a multiple choice integer program (see Bean (1984)). Third, we are now certain that all y vectors passed from (I) satisfy these equality constraints.

After applying this simplification, we formulate (DL) as

$$\begin{array}{rl} \max & u(b - Ay) \\ \text{subject to} & \\ U = & \left\{ \begin{array}{ll} uD & \le d \\ u & \ge 0 \end{array} \right. \end{array} \qquad \text{(DL)}$$

Letting $z = \max_p u^p(b - Ay)$, we can reformulate (I) as

$$\begin{array}{rcl} \min & & z \\ \text{subject to} & & \\ u^p(b - Ay) & \le & z, \quad p = 1, 2, \ldots, P \\ w^q(b - Ay) & \le & 0, \quad q = 1, 2, \ldots, Q \\ Ey & = & e \\ y & \ge & 0, \quad \text{integer.} \end{array} \qquad \text{(I)}$$

where u^p and w^q are the extreme points and rays of U, respectively.

To solve a particular instance of (MIP) requires solution of a sequence of instances of (DL) and (I). The sequence of linear programs generated are closely related to each other, resulting in efficient use of optimal basic feasible solutions as starting points for subsequent runs. The key to efficient computation is the number, and ease of solution, of the sequence of integer programs. Below we show that the special structure of (I) allows efficient solution.

4. Solving the Integer Program

In general, integer programs are easier to solve if the formulation has a small duality gap, that is, the optimal values of the integer program and its linear relaxation are close. Preprocessing integer programs seeks such a formulation (see Martin and Schrage (1985)). We can substantially tighten the formulation of (I) by exploiting the interpretation of the constraints in the Markov decision process context. Each multiplier in w is associated with a decision, $y_k(i,a)$. For each decision, there are two associated multipliers, one for the constraint from the original dynamic program, and one from the linear programming relaxation of the mixed integer constraint. Denote these multipliers $w_k^1(i,a)$ and $w_k^2(i,a)$, respectively.

Given an integer solution, $\hat{y}$, for which (DL) is unbounded with ray w, define

$$\begin{aligned} Y_1 &= \{(k,i,a) : w_k^1(i,a) > 0\} \\ Y_2 &= \{(k,i,a) : w_k^2(i,a) > 0\}. \end{aligned}$$

That is, Y_1 is the set of (k,i,a) in the dynamic program that are chosen in (MIP) and have corresponding tight constraints of the traditional type. Analogously, Y_2 are those chosen (k,i,a) with tight constraints of the type added to force a valid solution. Each extreme ray of U will generate its own sets Y_1 and Y_2. For ease of exposition we suppress this dependence in notation.

Lemma 1 *Any extreme ray constraint generated during the Bender's process, $w(b - Ay) \leq 0$, allows precisely the same feasible y vectors as*

$$\sum_{Y_2} y_k(i,a) \leq |Y_2| - 1. \tag{1}$$

Proof: Let $\hat{y}$ be a solution generated by (I) that results in (DL) having unbounded value along extreme ray w which, in turn, induces index sets Y_1 and Y_2. Traditional Bender's decomposition would append the constraint $w(b-Ay) \leq 0$ to (I). Rephrasing and collecting terms, this is

$$\sum_{Y_1} w_k^1(i,a) r_k(i,a) + \sum_{Y_2} w_k^2(i,a) \left[-r_k(i,a) - H + H y_k(i,a)\right] \leq 0 \tag{2}$$

By hypothesis, $w(b - A\hat{y}) > 0$. If $\hat{y}_k(i,a) = 0$ then the constraint corresponding to multiplier $w_k^2(i,a)$ (in the dual of (DL)) is necessarily slack implying that $w_k^2(i,a) = 0$. Hence, $w_k^2(i,a) > 0$ implies that $\hat{y}_k(i,a) = 1$. Referring back to (2), this implies that $wb > 0$. Rearranging from (2) gives

$$\begin{aligned} &\sum_{Y_2} w_k^2(i,a) \left[-H + H y_k(i,a)\right] \leq \\ &-\sum_{Y_1} w_k^1(i,a) r_k(i,a) + \sum_{Y_2} w_k^2(i,a) r_k(i,a) < 0. \end{aligned} \tag{3}$$

The last inequality is a restatement of $wb > 0$. Divide each term in (3) by H. By choosing H sufficiently large, we get

$$\sum_{Y_2} w_k^2(i,a) y_k(i,a) < \sum_{Y_2} w_k^2(i,a). \tag{4}$$

Note that this is valid since U and its extreme points and rays are independent of H (see definition of (DL) where all instances of H are contained in the matrix A). Since there are a finite number of extreme rays we can take the maximum such H as our value of H in (MIP). Then (4) holds for each extreme ray constraint. If y is infeasible in (1) all variables indexed in Y_2 are one. Then, since $wb > 0$, y is also infeasible in (2). If y is feasible in (1) then since all $w_k^2(i,a) > 0$ for $(k,i,a) \in Y_2$, and $y_k(i,a) = 0$ for some $(k,i,a) \in Y_2$, y is feasible in (2). ∎

The parallel analysis for the extreme point constraints, $u^p(b - Ay) \leq z$, is more complicated. As with the extreme ray constraints, we can rearrange a particular extreme point constraint to

$$\sum_{Y_1} u_k^1(i,a) r_k(i,a) + \sum_{Y_2} u_k^2(i,a)[-r_k(i,a) - H + Hy_k(i,a)] \leq z. \qquad (5)$$

Lemma 2 *If any $y_k(i,a) = 0$ for $(k,i,a) \in Y_2$ then* (5) *is satisfied trivially.*

Proof: H can be made arbitrarily large, driving the left hand side to $-\infty$. ∎

From this Lemma, an extreme point constraint can only affect z, the objective of (I), if all variables in Y_2 are one, that is, $\sum_{Y_2} y_k(i,a) = |Y_2|$. In this case (5) reduces to

$$K \equiv \sum_{Y_1} u_k^1(i,a) r_k(i,a) - \sum_{Y_2} u_k^2(i,a) r_k(i,a) \leq z.$$

If there are a number of extreme point constraints, index their corresponding values as K^p. Given a possible solution, y, its objective value is determined as the largest K^p such that $\sum_{Y_2^p} y_k(i,a) = |Y_2^p|$.

We can construct a linear formulation of this observation by creating binary variables, x_p, such that $x_p = 1$ if and only if extreme point constraint p defines the objective value at the current y solution. That is, the constraint $K^p \leq z$ is tight. Assume, WLOG, that the extreme point constraints are indexed by increasing K^p. For a given y, $x_{\bar{p}}$ should equal one if $\sum_{Y_2^{\bar{p}}} y_k(i,a) = |Y_2^{\bar{p}}|$ and $\sum_{Y_2^p} y_k(i,a) < |Y_2^p|$ for all $p > \bar{p}$. In this case the objective of (I) is $K^{\bar{p}}$.

If $\sum_{Y_2^p} y_k(i,a) < |Y_2^p|$ for all p, then none of the extreme point constraints are binding and the objective is unbounded below. In this case let $x_0 = 1$.

Using these observations we reformulate (I) as

$$\min \sum_{p=1}^{P} K^p x_p - x_0 H$$

$$\text{subject to}: Ey = e$$

$$\sum_{Y_2^q} y_k(i,a) \le |Y_2^q| - 1, \quad q = 1,2,\ldots,Q \tag{I2}$$

$$\sum_{p=0}^{P} x_p = 1$$

$$\sum_{Y_2^p} y_k(i,a) \le |Y_2^p| - 1 + \sum_{j \ge p} x_j, \quad p = 1,2,\ldots,P$$

$$x_p, y_k \in \{0,1\}, \text{for all } p,k,$$

where H is a large positive number.

We have proved

Theorem 4 *The integer program $(I2)$ returns the same solution as (I).*

Corollary 1 *A Bender's Decomposition using $(I2)$ in place of (I) will make exactly the same Bender's iterations as the original formulation.*

By the simple substitution $x'_p = 1 - x_p$, $(I2)$ can be formulated as a set packing problem. Such problems have small duality gaps and can be solved very efficiently (Chan (1987)). Below we present a tree search algorithm that exploits additional structure in $(I2)$ to solve the problem even more efficiently.

5. A Tree Search Algorithm

Consider the constraint matrix of $(I2)$. It has three major sections: the columns involving x variables, the y columns with multiple choice rows, and y columns with general set packing rows. Our approach will reduce the problem to searching for a feasible solution to the last of these segments.

If we fix $x_0 = 1$ and find a feasible y, we know that the optimal objective value is $-\infty$ and the corresponding y an optimal solution. If no feasible y exists, free x_0 and fix $x_1 = 1$. This imposes one constraint in $(I2)$ by relaxing the corresponding extreme point constraint. If a search turns up a feasible y, then the optimal value is K^1 and the corresponding y an optimal solution. Continue in this manner until a feasible y is found. It is not possible that $(I2)$ is infeasible for all x_p as this would imply that the value of the original dynamic program were $+\infty$. This is known not to be the case since we are using coherent salvage functions.

The effectiveness of this approach depends on the ease of finding feasible solutions. This is accomplished using the multiple choice tree structure of Bean (1984). The multiple choice constraints are embedded in the search tree so that all solutions considered satisfy them. This reduces the number of solutions to be considered and eliminates the need to consider the multiple choice constraints explicitly.

The algorithm for solving a single case of $(I2)$ is as follows:

Tree Search Algorithm

Step 0: Set $p = 0$. Go to Step 1.

Step 1: Search for a feasible y for $(I2)$ with $x_p = 1$. If none exists, go to Step 2. Else, go to Step 3.

Step 2: Increment p and go to Step 1.

Step 3: The feasible y is the optimal solution. The optimal value is K^p. Stop.

Noting that these integer programs arise from a Bender's decomposition, we will be solving a sequence of related problems. Between two successive integer problems in the Bender's process, the only change will be the addition of a single constraint. Hence, all values of p found to be infeasible are known to remain infeasible. In each iteration of the Bender's decomposition, begin the algorithm above with the p which ended the previous run. The new constraint will, by construction, eliminate the y returned in the last run. The new problem will have either a new optimal solution at the same p, or will force an increase in p. If the latter occurs then the lower bound will increase.

One final observation allows permanent elimination of some actions. Since the Bender's decomposition for the N period problem identifies y values that are infeasible for all choices of $L \in \Lambda$ in period $N+1$, and $\bar{v}^K_{N+1}(L) \in \Lambda$, these actions cannot become optimal in any K period problem with $K > N$. Further, increasing N simply adds constraints to (MIP). Adding constraints to a minimization problem can only increase the objective. The extreme point constraints in $(I2)$ give a lower bound on the optimal value of $(I2)$, which itself is a lower bound on (MIP). As N increases these lower bounds remain valid and aid in computation. Hence, we have proved

Lemma 4 *Constraints in $(I2)$ identified in the N period problem are valid in the K period problem for $K \geq N$.*

This lemma implies that if we solve the (MIP) for a given horizon N, get a negative objective, and decide to increase N and repeat the procedure, then all constraints can (and should) remain in the integer program. Leaving these constraints in place reduces the complexity of the (MIP) for longer time horizons.

We can now specify an algorithm for solving the (MIP).

Step 1: (Initialization) Set upper bound $z^u = \infty$, set lower bound $z^l = -\infty$. Choose an initial $\bar{y}$ (one candidate is the solution to the finite horizon problem with zero salvage values). Go to Step 2.

Step 2: (LP Phase) Solve (DL) with the given $\bar{y}$. If unbounded, append the constraint $\sum_{Y_2^q} y_k(i,a) \leq |Y_2^q| - 1$ to $(I2)$. If (DL) yields a finite solution, append the constraint $\sum_{Y_2^p} y_k(i,a) \leq |Y_2^p| - 1 + \sum_{j \geq p} x_j$ to $(I2)$, add $K^p x_p$ to the objective function of $(I2)$ and let $z^u = K^p$ if better than the current z^u. Reindex x_p so that the K^p remain nondecreasing in p. Go to Step 3.

Step 3: (IP Phase) Solve $(I2)$ by the Tree Search Algorithm. Let z^l equal the optimal value and $\bar{y}$ equal the new optimal solution. Go to Step 4.

Step 4: (Termination Test) If $z^l < z^u$ then go to Step 2. Otherwise, $\bar{y}$ is optimal. Solve L with this $\bar{y}$ to get an optimal v. Stop.

Note that at any intermediate point if $z^l \geq 0$ we know that the current N is a forecast horizon. Similarly, if $z^u < 0$, we know that this stopping rule cannot identify it as a forecast horizon. In either case, the analysis for this N may be stopped. However, in the latter case, we may wish to continue even though we have answered the question in order to generate additional constraints to aid in computation with a larger N.

6. Performance of the Stopping Rule

The effectiveness of the above stopping rule depends on (a) the speed with which the Bender's decomposition solves (MIP), and (b) the length of the forecast horizon identified by the stopping rule. To give a comparison of the length of the forecast horizons, we will contrast the horizons identified by our integer programming approach with those identified by a generalization of a method proposed by Bès and Lasserre. We can state this alternative stopping rule as follows.

Theorem 4 (Bès and Lasserre Stopping Rule) *Assuming* $\alpha a_0 < 1$, $\pi_k^K(\bar{i}, L) = \pi_k^N(\bar{i}, L)$ *for all* $K \geq N \geq k$, $L \in \Lambda$ *if*

$$v_k^N(\bar{i}, \text{O}) - \omega_k^N(\bar{i}, \text{O}) \geq 2\alpha M(\alpha a_0)^{N-k+1}$$

where

$$\omega_k^N(\bar{i}, \text{O}) = \max_{a \in \{A_k(\bar{i}) - \pi_k^N(\bar{i}, \text{O})\}} r_k(\bar{i}, a) + \alpha \sum_j p_k^{\bar{i}j}(a) v_{k+1}^N(j, \text{O})$$

Proof: Letting

$$v_k^N(\bar{i}, L; a) \equiv r_k(\bar{i}, a) + \alpha \sum_j p_k^{\bar{i}j}(a) v_{k+1}^N(j, L)$$

we can write

$$\begin{aligned}
v_k^N(\bar{i}, \text{O}; \bar{a}) - v_k^N(\bar{i}, \text{O}; a) &= r_k(\bar{i}, \bar{a}) + \alpha \sum_j p_k^{\bar{i}j}(\bar{a}) v_1^N(j, \text{O}) \\
&\quad - r_k(\bar{i}, a) + \alpha \sum_j p_k^{\bar{i}j}(a) v_{k+1}^N(j, \text{O}) \\
&= v_k^N(\bar{i}, L; \bar{a}) - v_k^N(\bar{i}, L; a) \\
&\quad + \alpha \sum_j (p_k^{\bar{i}j}(\bar{a}) - p_k^{\bar{i}j}(a))(v_{k+1}^N(j, \text{O}) - v_{k+1}^N(j, L))
\end{aligned}$$

for all $a \in A_k(\bar{i})$. By standard contraction mapping arguments, it is straightforward to show that

$$|v_{k+1}^N(j, \text{O}) - v_{k+1}^N(j, L)| \leq M(\alpha a_0)^{N-k+1}$$

for all $L \in \Lambda$ and $j \in S$ and hence

$$v_k^N(\bar{i}, L; \bar{a}) - v_k^N(\bar{i}, L; a) \geq v_k^N(\bar{i}, k; \bar{a}) - v_k^N(\bar{i}, \text{O}; a) - 2\alpha M(\alpha a_0)^{N-k+1}$$

for all $a \in A_k(\bar{i})$. Thus, if $v_k^N(\bar{i}, \text{O}; \bar{a}) - v_k^N(\bar{i}, \text{O}; a) \geq 2\alpha M(\alpha a_0)^{N-k+1}$ for all $a \in A_k(\bar{i}) - \{\bar{a}\}$, then it follows by Theorem 1 in Hopp that $\pi_k^K(\bar{i}, L) = \bar{a}$ for all $K \geq N$. ∎

This stopping rule is precisely that of Bès and Lasserre for our discrete time Markov decision process, except that it has been modified to include the a_0 coefficient and thereby exploit the ergodic properties of the process. Simply put, the Bès and Lasserre rule identifies an action $\bar{a}$ to be algorithmically optimal in state $\bar{i}$ when it "beats" all other feasible initial actions by some threshold amount in the N-period problem. Since this threshold goes to zero as N goes to ∞, their rule, like ours, will eventually be satisfied provided that the infinite horizon optimal initial action is unique.

Proving the Bès and Lasserre stopping rule in the above manner shows it to be a sufficient condition for the Hopp stopping rule and therefore immediately implies the following:

Corollary 2 *The Hopp stopping rule will always identify forecast horizons shorter than or equal to those identified by the Bès and Lasserre stopping rule.*

However, while the Hopp stopping rule is guaranteed to find forecast horizons at least as short as the Bès and Lasserre rule, the integer programming test of this rule requires substantially more computation. The Bès and Lasserre rule merely requires solving an N-period dynamic program, while the integer programming approach requires solving an N-period dynamic program (to get a candidate action) and (MIP). In some cases, the fact that the (MIP) approach stops sooner may result in less total computation than the Bès and Lasserre rule. Often, however, this will not be the case. Despite this added computational burden, the (MIP) stopping rule may be more attractive in certain situations because it requires less reliance on forecasted data. The primary advantage of short forecast horizons is reduction of the need to forecast distant parameters, a procedure that is costly and inaccurate. Any additional computation required may be a small price for elimination of part of the forecasting burden.

To illustrate the performance of our stopping rule and compare it with that of Bès and Lasserre, we analyze three examples. In Example 1, we consider a problem with three states per stage and two actions per state and discount factor 0.9, and rewards and transition probabilities that are defined for stage 0 and stages m and $2m, m = 1, 2, \ldots$ in Tables 1 and 2.

To apply the algorithm for solving (MIP) for Example 1 with $N = 1$, we need an initial y vector. We get this by solving the dynamic program for $N = 1$ with zero salvage vectors and get $\pi_0^1(\cdot; \mathbf{O}) = (1\ 2\ 2)$ and $\pi_1^1(\cdot; \mathbf{O}) = (2\ 1\ 1)$. We use these values to initiate the algorithm for $\bar{i} = 1$ using action $\hat{a} = 1$ as the candidate action. The progress of the algorithm is summarized in Table 3.

In our first run of (DL) we fix $y_0(1,1) = 1, y_1(1,2) = 1, y_1(2,1) = 1, y_1(3,1) = 1$ and all other y values equal to zero. Then (DL) yields a bounded solution of 0.638, which becomes the upper bound, and generates a u-vector, which we append to the constraint set in $(I2)$. Solving $(I2)$ yields an optimal objective value of -44.362, which becomes the lower bound, and yields a new y vector. When this vector is used in (DL), it yields an unbounded solution, so the upper bound is unchanged and a y-constraint is added to the feasible region of $(I2)$. Specifically, the procedure adds the constraint that $y_1(2,2) \leq 0$ (i.e., $y_1(2,2) = 0$).

Solving $(I2)$ with this constraint added does not change the optimal objective value, but does yield a new y vector. When this y vector is used in (DL) it yields a bounded solution of 0.556, which becomes the new upper bound, and also produces a new u-constraint for $(I2)$. Solving $(I2)$ yields an objective of 0.556, which becomes the lower bound. Since the upper and lower bounds are the same, we can stop and the solution to (MIP) is 0.556. This means that action 1 in state 1 of stage 0 satisfies the stopping rule and is therefore optimal in all problems with time horizons longer than $N = 1$.

Now, as a case where the stopping rule is not satisfied immediately, consider Example 2, which is identical to Example 1 with the exception that, $r_0(1,2) = 10$ instead of 3 and $r_1(\cdot,2) = (11\ 9\ 13)$ instead of $(5\ 2\ 5)$. With these data, the dynamic program for $N = 1$ shows that $\pi_0^1(1) = 2$, so we make this our candidate for the test. As shown in Table 4, the algorithm solves (MIP) after adding two y-constraints.

In the very first solution of (DL), the optimal value, and hence the upper bound, was negative. From this we know immediately that the optimal objective value of (MIP) is negative and hence the stopping rule cannot be satisfied. We could immediately terminate the test, increase N to 2 and start again. However, the effort in solving (MIP) is not completely wasted, since it generates y-constraints which were appended to the feasible region of (I) and used in the procedure for $N = 2$. Whether or not the savings from finding such constraints justifies the work involved in solving (MIP) after its objective is known to be negative is an open research question.

To complete the test of the stopping rule in Example 2, we then increase N to 2, update (MIP) and repeat the algorithm. Note that since the feasible region of (MIP) decreases as we increase the time horizon, we can use the lower bound from the solution for $N = 1$ as the starting lower bound. After generating several constraints to prevent unboundedness, the algorithm terminates with an objective of 0.161. Since this is positive, the stopping rule is satisfied, and action 2 is verified to be optimal for all time horizons longer than 2. Note that in the iteration labelled IP2A in Table 4 for $N = 2$ both the upper and lower bounds are positive. Once again, we did not need to solve (MIP) to optimality. The fact that the lower bound was positive ensured that the final solution would be positive and the stopping rule would be satisfied.

Example 3 considers a replacement problem for a system subject to deterioration on state space $S = \{1, \ldots, 10\}$. In each state, the decision-maker can choose one of two feasible actions, "replace" (action 1) or "do nothing" (action 2). We assume that $\alpha = 0.8$ and the one period rewards and transition probabilities are as follows.

$$
\begin{aligned}
r_k(i,1) &= \begin{cases} 8 & , k=0, \ i=1 \\ 6+0.5k & , 1 \le k \le 4, \ i=1 \\ 7-i+0.5k & , 0 \le k \le 4, \ i>1 \\ 9-i & , k>4, \ \forall i \end{cases} \\
r_k(i,2) &= 10-i \quad , \forall i \\
p_k^{i1}(1) &= 1 \quad , \forall k, i \\
p_k^{ij}(2) &= \begin{cases} 0.7 & , j=1 \ \forall k, \ i \neq 10 \\ 0.3 & , j=i+1, \ \forall k, \ i \neq 10 \\ 1.0 & , i=j=10, \ \forall k \end{cases}
\end{aligned}
$$

To ensure nonnegativity of the one period rewards, we added 10 to each reward for our computational test.

The linear program, (DL), for Example 3 with $N = 1$ has 154 variables and 22 constraints. The largest integer program that must be solved to test $N = 1$ has 21 variables and 15 constraints. As shown in Table 5, the algorithm stops after six iterations and fails to identify $N = 1$ as a forecast horizon. Increasing N to 2 results in (DL) having 194 variables and 32 constraints and a largest integer program of size of 41 variables and 30 constraints. The algorithm identifies $N = 2$ as a forecast horizon in ten iterations. Because the linear programs are still not excessively large and the integer programs are of an extremely simple form, all iterations took less than a few seconds on a 386 PC.

To further test the solution procedure for solving $(I2)$, we generated an equipment replacement problem similar to Example 3 having 20 states per stage. The largest integer program encountered in this problem had 85 variables and 46 constraints. The process identified $N = 2$ as a forecast horizon in three iterations. While this example is not a large-scale problem, it does provide evidence that our stopping rule can be feasibly applied to realistically sized problems. Most importantly, all instances of $(I2)$ were solved in less than a second.

Finally, we turn to the question of how the forecast horizon generated by the integer programming approach compares with that identified by the Bès and Lasserre approach. Table 6 shows $v_0^N(\bar{\imath}, 0) - w_0^N(\bar{\imath}, 0)$ compared with $2\alpha\bar{r}(\alpha a_0)^N$ for various values of N for Example 1. We see that the termination test, showing that action 1 is infinite horizon optimal in state 1, is not satisfied until $N = 4$. Recall that the integer programming stopping rule identified this action as optimal with $N = 1$. Similarly, Table 7 shows that the Bès and Lasserre approach applied to Example 2 does not identify action 2 as optimal in state 1 until $N = 9$. The integer programming stopping rule identified this action as optimal for $N = 2$. For Example 3, $N = 25$ represents the shortest horizon satisfying the Bès and Lasserre criterion. This compares with a value of $N = 2$ generated by our (MIP) stopping rule. In this example, a forecaster would have had to develop 23 additional periods of data to run the faster algorithm.

As a theoretical comparison, consider the following example. At stage $k = 0$, we begin in state 1 and are faced with two possible actions. Action 1 takes us to state 1 with probability 1 and offers one period reward of ϵ. Action 2 takes us to state 2 with probability 1 and offers one period reward of zero. In stages 1 and beyond, there is only one action, which takes us to state 1 with probability 1 regardless of the starting state and offers no reward. Action 1 is optimal and has $v_0^N(1) - \hat{v}_0^N(1) = \epsilon$ for all $N \geq 1$. The integer programming stopping rule will terminate for $N = 1$ and verify that action 1 is optimal. However, since $a_0 = 1$, the Bès and Lasserre rule requires

$$v_0^N(1) - \hat{v}_0^N(1) \geq \frac{2\bar{r}\alpha^{N+1}}{1-\alpha}$$

or, equivalently

$$\epsilon \geq \frac{2\bar{r}\alpha^{N+1}}{1-\alpha}.$$

The closer α becomes to 1, the larger N must be to satisfy this requirement. Hence, while the integer programming stopping rule will always terminate for $N = 1$, the Bès and Lasserre rule can yield arbitrarily long horizons for this example. Although this is a contrived example, it serves to illustrate that the difference between the lengths of the forecast horizons identified by the two stopping rules can be substantial.

The primary reason for the improved performance of the integer programming stopping rule is that it tightly constrains the effect of varying salvage values in period $N + 1$ by adjusting decision rules in periods 1 through N as the salvage values change. The Bès and Lasserre approach does not limit the effect of the salvage values; it merely requires that the candidate action "beat" all other actions over the interval $[0, N]$ by an amount that makes it impossible for another action to "catch up" in subsequent periods. Adjusting the intermediate decision rules to evaluate the effect of changing the salvage values accounts for the increased computational complexity of the integer programming stopping rule, but appears worth the increased effort in light of the significantly shorter forecast horizons it produces.

7. Conclusions and Further Work

The integer programming approach to identifying forecast horizons in non-homogeneous Markov decision processes appears more attractive from a computational standpoint that the nonlinear programming and piecewise linearity approaches suggested in Hopp.

Evaluating the integer programming stopping rule may take longer than the tail value approaches, but finds shorter forecast horizons. Overly long time horizons entail more data collection (i.e., forecasting), and more reliance on uncertain future data.

In some specific applications it is possible to develop stopping rules that identify forecast horizons that are as short as possible given certain aggregate descriptors of the future (see e.g., Chand and Morton (1982), Hopp and Nair (1987)). In the general nonhomogeneous MDP, such perfect efficiency is probably not possible. Even so, it may be possible to further limit the size of the set Λ through more judicious preprocessing of the problem. Further work is needed to determine what improvements are possible in this arena.

The current solution system involves separate codes to solve the dynamic program (to find a candidate $\bar{a}$), the linear programs and the integer programs. Future research is needed to develop an integrated system for efficiently computing forecast horizons in nonhomogeneous Markov decision processes.

Acknowledgement

We would like to thank an anonymous referee for many insightful comments that greatly improved the paper.

References

[1] Bean, J.C. 1984. A Lagrangian Algorithm for the Multiple Choice Integer Program. *Opns. Res.* **32**, 1185-1193.

[2] Bean, J.C., R.L. Smith and J.B. Lasserre. 1990. Denumerable State Nonhomogeneous Markov Decision Processes *J. Math. Analysis and Appl.* **153**, 64-77.

[3] Bès, C., and J. Lasserre. 1986. An On-Line Procedure in Discounted Infinite-Horizon Stochastic Optimal Control. *J. of Optimization Theory and Applications* **50**, 61-67.

[4] Bès, C., and S. Sethi. 1988. Concepts of Forecast and Decision Horizons: Applications to Dynamic Stochastic Optimization Problems. *Math. O.R.* **13**, 295-310.

[5] Chan, T. 1987. A Multiplier Adjustment Based Branch-and-Bound Algorithm for Solving the Set Partitioning Problem. Unpublished Ph.D. Dissertation. Department of Industrial and Operations Engineering, the University of Michigan, Ann Arbor, MI 48109-2117.

[6] Chand, S. and T. E. Morton. 1982. A Perfect Planning Horizon for a Deterministic Cash Balance Problem. *Man. Sci.* **28**, 652-669.

[7] Flynn, J. 1980. On optimality Criteria for Dynamic Programs with Long Finite Horizons. *J. Math. Analysis and Appl.* **76**, 202-208.

[8] Hinderer, K. and G. Hübner. 1974. An Improvement of J.F. Shapiro's Turnpike Theorem for the Horizon of Finite Stage Discrete Dynamic Programs. *Trans. 7th Prague Conference on Information Theory, Statistical Decision Functions and Random Processes* **Vol. A**, Academia, Prague, 245-255.

[9] Hopp, W.J. 1989. Identifying Forecast Horizons in Nonhomogeneous Markov Decision Processes. *Opns. Res.* **37**, 339-343.

[10] Hopp, W.J., J.C. Bean, and R.L. Smith. 1987. A New Optimality Criterion For Nonhomogeneous Markov Decision Processes. *Opns. Res.* **35**, 875-883.

[11] Hopp, W. J. and S. K. Nair. 1991. "Timing Replacement Decisions Under Discontinuous Technological Change," *Naval Res. Log.*, **38**, 203-220.

[12] Martin, W.K. and L. Schrage. 1985. Subset Coefficient Reduction Sets for 0/1 Mixed Integer Programming. *Opns. Res.* **33**, 505-526.

[13] Morton, T. and W. Wecker. 1977. Discounting, Ergodicity and Convergence for Markov Decision Processes. *Man. Sci.* **23**, 890-900.

[14] Salkin, H.M. 1975. *Integer Programming.* Addison-Wesley, Reading, MA.

[15] Tseng, P. 1990. Solving H-Horizon, Stationary Markov Decision Problems in Time Proportional to log(H). *Opns. Res. Let.* **9** 287-297.

Stage	Action	$r_k(i,a)$		
k	a	$i=1$	$i=2$	$i=3$
0	1	10	5	2
0	2	3	7	12
m	1	2	8	12
m	2	5	2	5
2m	1	6	2	8
2m	2	3	1	10

Table 1: Rewards for Example 1.

Stage	State	Action	$p_k^{ij}(a)$		
k	i	a	$j=1$	$j=2$	$j=3$
0	1	1	0.3	0.3	0.4
0	2	1	0.4	0.4	0.2
0	3	1	0.4	0.2	0.4
0	1	2	0.2	0.2	0.6
0	2	2	0.3	0.3	0.4
0	3	2	0.5	0.0	0.5
m	1	1	0.4	0.6	0.0
m	2	1	0.0	0.4	0.6
m	3	1	0.3	0.3	0.4
m	1	2	0.4	0.2	0.4
m	2	2	0.2	0.2	0.6
m	3	2	0.5	0.3	0.2
2m	1	1	0.3	0.3	0.4
2m	2	1	0.5	0.0	0.5
2m	3	1	0.4	0.5	0.1
2m	1	2	0.2	0.6	0.2
2m	2	2	0.3	0.2	0.5
2m	3	2	0.3	0.4	0.3

Table 2: Transition Probabilities for Example 1.

Iteration	Upper Bound	Lower Bound	Comment
DL1	0.638	$-\infty$	u-vector generated
IP1A	0.638	-44.362	
DL2A	0.638	-44.362	y-constraint generated
IP1B	0.638	-44.362	
DL2B	0.556	-44.362	u-vector generated
IP2	0.556	0.556	

Table 3: Results for Example 1 with N=1.

Iteration	Upper Bound	Lower Bound	Comment
		$N = 1$	
DL1	-0.027	$-\infty$	u-vector generated
IP1A	-0.027	-90.030	
DL2A	-0.027	-90.030	y-constraint generated
IP1B	-0.027	-90.030	
DL2B	-0.027	-90.030	y-constraint generated
IP1C	-0.027	-90.030	
DL2C	-0.027	-45.030	u-vector generated
IP2	-0.027	-0.027	
		$N = 2$	
DL1	0.202	-0.027	u-vector generated
IP1A	0.202	-0.027	
DL2A	0.202	-0.027	y-constraint generated
IP1B	0.202	-0.027	
DL2B	0.202	-0.027	y-constraint generated
IP1C	0.202	-0.027	
DL2C	0.161	-0.027	u-vector generated
IP2A	0.161	0.063	
DL3	0.161	0.063	y-constraint generated
IP2B	0.161	0.161	

Table 4: Results for Example 2 with N=1,2.

Iteration	Upper Bound	Lower Bound	Comment
$N = 1$			
DL1	0.149	$-\infty$	u-vector generated
IP1	0.149	-148.500	
DL2	0.149	-148.500	u-vector generated
IP2	0.149	-28.500	
DL3	-0.697	-28.500	u-vector generated
IP3	-0.697	-0.697	
$N = 2$			
DL1	0.300	-0.697	u-vector generated
IP1A	0.300	-0.697	
DL2A	0.300	-0.697	y-constraint generated
IP1B	0.300	-0.697	
DL2B	0.300	-0.697	y-constraint generated
IP1C	0.300	-0.697	
DL2C	0.300	-0.697	y-constraint generated
IP1D	0.300	-0.697	
DL2D	0.300	-0.697	y-constraint generated
IP1E	0.300	0.300	

Table 5: Results for Example 3 with N=1,2.

N	$v_0^N(\bar{i})$	$w_0^N(\bar{i},0)$	$v_0^N(\bar{i},0) - w_0^N(\bar{i},0)$	$2\alpha M(\alpha a_0)^N$
1	17.830	11.820	6.010	21.13
2	23.208	17.134	6.074	11.41
3	29.373	23.304	6.069	6.16
4	33.734	27.664	6.069	3.33

Table 6: Bès and Sethi Approach for Example 1.

N	$v_0^N(\bar{\imath})$	$w_0^N(\bar{\imath},0)$	$v_0^N(\bar{\imath},0) - w_0^N(\bar{\imath},0)$	$2\alpha M(\alpha a_0)^N$
1	20.620	20.080	0.540	23.24
2	25.674	25.394	0.280	12.55
3	31.885	31.590	0.295	6.78
4	36.244	35.950	0.294	3.66
5	41.240	40.946	0.294	1.98
6	44.772	44.478	0.294	1.07
7	48.819	48.525	0.294	0.57
8	51.680	51.386	0.294	0.31
9	54.958	54.664	0.294	0.17

Table 7: Bès and Sethi Approach for Example 2.