Operations Research

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

Technical Note—A Recursion Algorithm for Finding Pure Admissible Decision Functions in Statistical Decisions

Herbert Moskowitz,

To cite this article:
Herbert Moskowitz, (1975) Technical Note—A Recursion Algorithm for Finding Pure Admissible Decision Functions in Statistical Decisions. Operations Research 23(5):1037-1042. https://doi.org/10.1287/opre.23.5.1037

We conclude that $\lim_{\nu\to\infty} \partial\xi/\partial\nu>0$, for $t<9/K^2$ and $\lim_{\nu\to\infty} \partial\xi/\partial\nu<0$, for $t>9/K^2$. Thus, $\tau^*=\lim_{\nu\to\infty} \tau_\nu=9/K^2$, proving (2). The validity of (3) follows easily from the uniformity of the asymptotic (in ν) estimates of g and ξ for t on bounded intervals. This completes the proof of Theorem 1.

3. REMARK

We remark that Theorem 1 implies that for large ν the gamma plan expands more rapidly than the Koopman plan until the area of the ellipse of containment is approximately 6 times the area of the ellipse of uncertainty. After that, the gamma plan expands more slowly than the Koopman plan. If we define the effective area of coverage by time t to be $\bar{W}ut$ (the actual area swept if $W=\bar{W}$ and coverage were nonoverlapping), then τ^* can equivalently be expressed as the time at which the ratio of the effective area of coverage to the area of the ellipse of uncertainty is 9.

ACKNOWLEDGMENT

Research on this paper was supported in part by the Office of Naval Research under Contract Nos. N00014-69-C-0435 and N00014-74-C-0328.

REFERENCE

1. H. R. RICHARDSON AND B. BELKIN, "Optimal Search With Uncertain Sweep Width," *Opns. Res.* **20,** 764–784 (1972).

A Recursion Algorithm for Finding Pure Admissible Decision Functions in Statistical Decisions

HERBERT MOSKOWITZ

Purdue University, West Lafayette, Indiana

(Received original November 11, 1971; final, March 7, 1975)

This note develops a recursion algorithm for finding the set of ordinally admissible decision functions in statistical decision problems when such functions have a Cartesian product structure. An example is used to demonstrate the computations. The generality of the algorithm is discussed briefly, and some computational experience is presented.

WE FIRST define the conditional-utility vector $U(e, d)$ of d, given a fixed experiment e, by

$$U(e,\, d) = \sum_{j=1}^{j=m} u[e,\, z_j,\, d(z_j),\, \theta_1]p(z_j|\theta_1),\, \cdots,\, \sum_{j=1}^{j=m} u[e,\, z_j,\, d(z_j),\, \theta_n]p(z_j|\theta_n), \tag{1}$$

where $d\colon Z \to A$ is defined as the set of decision functions; $Z = \{z_1, \cdots, z_m\}$ are the messages; $A = \{a_1, \cdots, a_k\}$ are the actions; and $\Theta = \{\theta_1, \cdots, \theta_n\}$ are the states. The utility of the action associated with the jth message of e and gth state is defined as $u[e, z_j, d(z_j), \theta_g]$. The probability of the z_jth message, given θ_g, is defined as $p(z_j|\theta_g)$. From these definitions we may establish

LEMMA 1. *The conditional-utility vector*

$$U(e,\, d) = \sum_{j=1}^{j=m} V(i[e,\, d,\, j],\, j,\, e), \tag{2}$$

where $V(i, j, e)$ is the n-tuple whose gth component is $u(e, z_j, a_i, \theta_g)p(z_j|\theta_g)$ and $i[e, d, j]$ is the subscript of the act specified by d if z_j occurs when e was conducted; that is, $a_{i[e,d,j]} = d(z_j)$ [given e undertsood].

The proof of Lemma 1 is purely definitional and is left to the reader.

We now define and focus on the matrix γ whose (i, j)th element is the n-tuple $V(i, j, e)$. In essence Lemma 1 says that every decision function d is really a selection of a vector $V(i, j, e)$ from each column j of this matrix, with d's conditional-utility vector being the sum of the selected vectors. Lemma 2 shows how an undominated vector can be built up sequentially. Let $\sum_{l=1}^{l=t} X_l$ denote $\{\sum_{l=1}^{l=t} x_l\colon x_l \epsilon X_l,\ \forall l\}$.

LEMMA 2. *If $X_1, \cdots, X_t$ are nonempty, finite sets of n-tuples, then*

$$V\text{-max}\Big(\sum_{l=1}^{l=t} X_l\Big) = V\text{-max}\Big[X_t + V\text{-max}\Big(\sum_{l=1}^{t-1} (X_l)\Big)\Big], \tag{3}$$

where V-max denotes vector maximization for finding the ordinally admissible set.

Proof. Let $X = \sum_{l=1}^{t-1} X_l$. The finiteness of every X_l and the definition of V-max imply that if $x \epsilon X$ but $x \notin V\text{-max}(X)$, then $\exists x' \epsilon V\text{-max}(X)$ such that $x' \geq$ but $\neq x$. Thus $x_t + x' \geq$ but $\neq x_t + x$ for every $x_t \epsilon X_t$. Hence $x_t + x \notin V\text{-max}(X_t + X)$, which implies that $V\text{-max}(X_t + X) \subset V\text{-max}[X_t + V\text{-max}(X)]$, from which the assertion is obvious.

The algorithm follows immediately from Lemma 2 and is stated as the following corollary.

COROLLARY. *If $X_j(e) = \{V(i,\ j,\ e)\colon i = 1, \cdots, k\}$ and $Y_j(e) = V\text{-max}\cdot [\sum_{h=1}^{h=j} X_h(e)]$, then*

$$Y_j(e) = V\text{-max}[X_j(e) + Y_{j-1}(e)], \quad for \quad j = 1, \cdots, m \tag{4}$$

where $Y_0(e) = \phi$, by definition.

Our objective is to determine $Y_m(e)$ by finding the vector-maximal $V(i, 1, e)$, forming the sums of them with all $V(i, 2, e)$, calculating the vector-maximal set of these sums, and so on. A simple example illustrates the algorithm.

Example. Consider the 3-act, 3-state, single-experiment, 3-message problem in Table I. From the action-state utility and conditional probability matrices, form the $k \times m$ (3×3) matrix of $V(i, j, e)$'s [given experiment e]. For example, from (2), we obtain $V(1, 1, e) = [(50)(0.1), (20)(0.3), (0)(0.6)] = (5, 6, 0)$. Recursive applications of (4) are shown in Table II. As seen, only 7 of the 27 pure decision functions are ordinally admissible.

Remarks. The algorithm is applicable to the case where the action sets are not the same for each message in an experiment. It is also applicable if attention must be restricted to some proper subset A'^z of the set of decision functions A^z. In this case, the algorithm would be applied to a generally larger subset of decision functions A''^z, where $A'^z \subset A''^z \subset A^z$ and where A''^z is defined by the individual action-message pairs comprising the decision functions of interest.

TABLE I
HYPOTHETICAL EXAMPLE

Action-state utility matrix

$\Theta \backslash A$	$u(e, z_j, d(z_j), \theta_g)$ a_1	a_2	a_3
β_1	50	20	0
β_2	20	20	20
β_3	0	10	60

Conditional probability matrix

$\Theta \backslash Z$	for Experiment (e): $p(z_j \mid \theta_g)$ z_1	z_2	z_3
θ_1	0.1	0.2	0.7
θ_2	0.3	0.3	0.4
θ_3	0.6	0.3	0.1

$V(i, j, e)$ Matrix

$A \backslash Z$	z_1	z_2	z_3
a_1	(5, 6, 0)	(10, 6, 0)	(35, 8, 0)
a_2	(2, 6, 6)	(4, 6, 3)	(14, 8, 1)
a_3	(0, 6, 36)	(0, 6, 18)	(0, 8, 6)

1. CHOICE OF EXPERIMENT

We next indicate how to proceed when the decision maker has a choice of experiment.

LEMMA 3. *If $X_1, \cdots, X_t$ are nonempty, finite sets of n-tuples, then*

$$V\text{-max}\left(\bigcup_{l=1}^{l=t} X_l\right) = V\text{-max}\left[V\text{-max}(X_t) \cup V\text{-max}\left(\bigcup_{l=1}^{t-1} V\text{-max}(X_l)\right)\right]. \tag{5}$$

Although involving set uinons rather than set additions, the proof is similar to that of Lemma 2 and is therefore omitted. A modification of the algorithm in (4) follows immediately from Lemma 3 and is stated as the following corollary.

TABLE II

RECURSION ALGORITHM SOLUTION TO EXAMPLE

$Y_j(e)=V-\max[X_j(e)+Y_{j-1}(e)]$, for $m=3$

$j=1$

$d(z_1)$ \ $d(z_0)$	$Y_1(e)$
	Φ
a_1	(5, 6, 0)
a_2	(2, 6, 6)
a_3	(0, 6, 36)

$j=2$

$d(z_2)$ \ $d(z_1)$	$Y_2(e)$		
	a_1	a_2	a_3
a_1	(15, 12, 0)	(12, 12, 6)	(10, 12, 36)
a_2	(9, 12, 3)*	(6, 12, 9)*	(4, 12, 39)
a_3	(5, 12, 18)*	(2, 12, 24)*	(0, 12, 54)

$j=3$

$d(z_3)$ \ $d(z_2)$	$Y_3(e)$				
	a_1a_1	a_1a_2	a_1a_3	a_2a_3	a_3a_3
a_1	(50, 20, 0)	(47, 20, 6)	(45, 20, 36)	(39, 20, 39)	(35, 20, 54)
a_2	(29, 20, 1)*	(26, 20, 7)*	(24, 20, 37)*	(18, 20, 40)*	(14, 20, 55)
a_3	(15, 20, 6)*	(12, 20, 12)*	(10, 20, 42)*	(4, 20, 45)*	(0, 20, 60)

* The vectors marked with an asterisk are dominated and therefore eliminated.

COROLLARY. *If $E=\{e_1, \cdots, e_w\}$ and if G_v denotes the set of all ordinally admissible (e_h, d) for $h\leqq v$, then for $v=1, \cdots, w$ we have*

$$G_v=V\text{-}\max[Y_{m_e}(e_v)\cup G_{v-1}], \tag{6}$$

where $G_0=\phi$, by definition.

We subscript m by e because the number of messages may depend upon the particular choice of e. In other words, we iterate recursively on the experiment (rather than on the message), keeping only the ordinally admissible experiment-decision function pairs as we work forward.

2. COMPUTATIONAL EXPERIENCE

The algorithm was programmed in FORTRAN IV and executed on the CDC 6500 computer.[1] The efficiency of the algorithm was tested on fifteen single-experiment problems, systematically varying in the number of messages, actions, and states, whose parameters were randomly generated. Total CPU time over all problems tested ranged from 0.06 seconds

TABLE III
ACTUAL VERSUS PREDICTED (FROM REGRESSION MODEL) CPU TIME*

Problem parameters			CPU time (seconds)	
Z	A	θ	Actual	Predicted
3	3	3	0.07	0.09
3	3	6	0.11	0.22
3	3	9	0.15	0.54
3	6	3	0.62	0.57
3	6	6	1.92	2.33
3	6	9	4.50	5.20
3	9	3	3.40	1.72
3	9	6	18.00	8.48
3	9	9	47.00	15.80
6	3	3	3.00	1.70
6	6	3	29.80	60.11
6	9	3	175.20	481.54

* $R^2=0.86$; $F_{2,60}=189.91$, $\sigma_e=0.89$.

for a problem with 27 decision functions to about 35 minutes for our largest size problem, which had 390×10^6 decision functions.

We developed the regression model of (7) to predict CPU time for any given problem (Table III):

$$\hat{Y}=0.0005X_2^{0.87X_1}X_3^{2.06}, \tag{7}$$

where $\hat{Y}$=estimated CPU time in seconds, $X_1=z$, number of messages, $X_2=a$, number of actions, and $X_3=\theta$, number of states.

The model structure was based on the premise that computational efficiency of the recursion algorithm was principally a function of the total number of decision functions (k^m) considered and the number of states (n). The sequence in which messages are processed also affects efficiency somewhat, but was ignored in the model.

ACKNOWLEDGMENTS

The author gratefully acknowledges the contributions of the two referees, Irving H. LaValle and Richard H. Shachtman, whose comments and suggestions throughout several versions of this manuscript have been invaluable. Also, the proof of Lemma 2, which improves upon my original proof, is essentially due to LaValle. Thanks also go to Raymond Jones, who programmed the algorithm, and to Gordon P. Wright and Rony Adelsman for their helpful suggestions on an earlier draft.

REFERENCES

1. H. Moskowitz and R. D. Jones, "A Program of a Dynamic Programming Algorithm for Finding Pure Admissible Decision Functions in Statistical Decisions," *Behavior Res. Methods and Instrumentation* **6,** 350–351 (May, 1974).