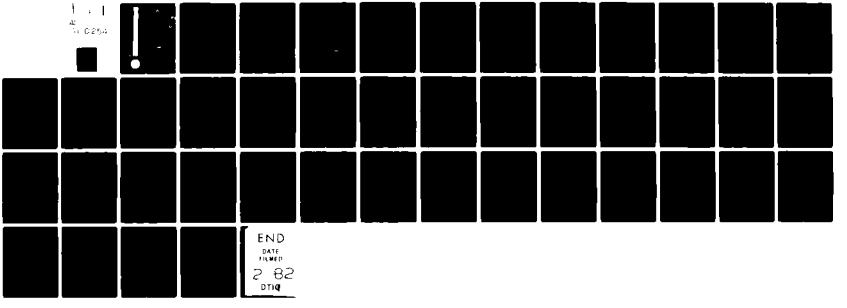
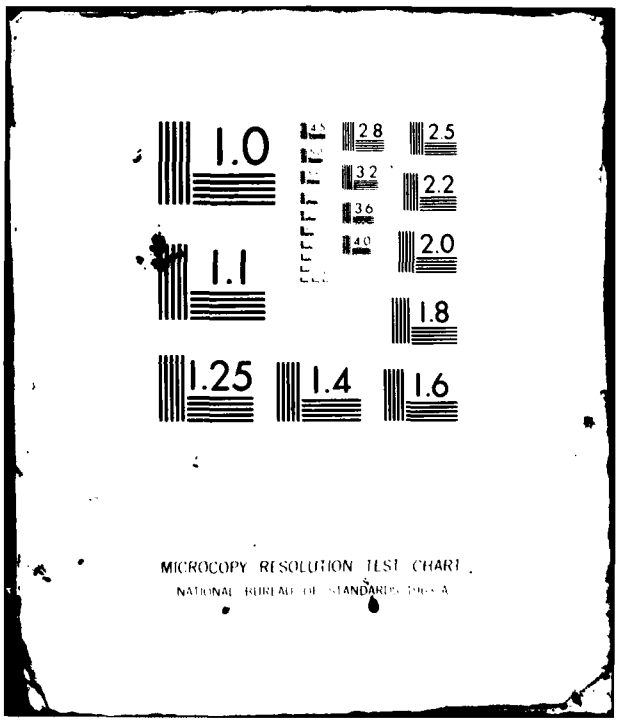
AD-A110 254 STATE UNIV OF NEW YORK AT BUFFALO AMHERST DEPT OF CO--ETC F/G 12/1
A NOTE ON COMPUTING THE ASYMPTOTIC FORM OF A LIMITED SEQUENCE O--ETC(U)
NOV 81 N V FINDLER, N MAZUR, B MCCALL AFOSR-81-0220

UNCLASSIFIED AFOSR-TR-81-0872 NL



END
DATE FILMED
2 82
DTIC


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963-A

AFOSR-TR- 81-0872

LEVEL II

3

# A NOTE ON COMPUTING THE ASYMPTOTIC FORM OF A LIMITED SEQUENCE OF DECISION TREES*



BY

NICHOLAS V. FINDLER,
NEAL MAZUR, AND BEDE MCCALL

NOVEMBER 1981

DEPARTMENTAL TECHNICAL REPORT NUMBER: #189

GROUP FOR COMPUTER STUDIES OF STRATEGIES
TECHNICAL REPORT NUMBER: # 3

*THE WORK DESCRIBED HAS BEEN SUPPORTED BY THE AIR FORCE OFFICE OF SCIENTIFIC RESEARCH GRANT [redacted]. AFOSR-81-0220

STATE UNIVERSITY OF NEW YORK AT BUFFALO



*Department of Computer Science*

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

## REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

| | |
|---|---|
| 1. REPORT NUMBER: AFOSR-TR- 81-0872 | 2. GOVT ACCESSION NO.: AD-A110254 |
| 3. RECIPIENT'S CATALOG NUMBER: | |
| 4. TITLE (and Subtitle): A NOTE ON COMPUTING THE ASYMPTOTIC FORM OF A LIMITED SEQUENCE OF DECISION TREES | 5. TYPE OF REPORT & PERIOD COVERED: INTERIM, 1 JUL 80 - 30 JUN 81 |
| | 6. PERFORMING ORG. REPORT NUMBER: |
| 7. AUTHOR(s): Nicholas V. Findler, Neal Mazur and Bede McCall | 8. CONTRACT OR GRANT NUMBER(s): AFOSR-81-0220 |
| 9. PERFORMING ORGANIZATION NAME AND ADDRESS: Department of Computer Science, State University of New York at Buffalo, 4226 Ridge Lea Road, Amherst NY 14226 | 10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: 61102F; 2304/A2 |
| 11. CONTROLLING OFFICE NAME AND ADDRESS: Directorate of Mathematical & Information Sciences, Air Force Office of Scientific Research, Bolling AFB DC 20332 | 12. REPORT DATE: NOV 81 |
| | 13. NUMBER OF PAGES: 38 |
| 14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) | 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)

Learning programs, decision-support tools, optimization techniques, decision trees, extrapolation techniques for multi-dimensional entities.

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

It is often desirable to compute the asymptotic form of a limited sequence of decision trees (DT's) after the set of decision variables has reached a stable, constant membership. Such is the case when the DT's characterize the evolving behavior of a learning strategy in the Quasi-Optimizer system under development.

An algorithm is described which builds the asymptotic decision (CONTINUED)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

ITEM #20, CONTINUED:

tree in breadth-first order, node-by-node, starting at the root. In the course of computing the asymptotic value of a node (that is, its out-degree and the limiting points of its subranges), three situations may arise:

(i) the computation is possible;

(ii) the computation seems to be possible but more DT's have to be acquired of the evolving strategy to reach the desired level of statistical significance;

(iii) the computation is not possible.

(3)

# A NOTE ON COMPUTING THE ASYMPTOTIC FORM OF A LIMITED SEQUENCE OF DECISION TREES*

BY

NICHOLAS V. FINDLER,
NEAL MAZUR, AND BEDE MCCALL

NOVEMBER 1981

DEPARTMENTAL TECHNICAL REPORT NUMBER: #189

GROUP FOR COMPUTER STUDIES OF STRATEGIES
TECHNICAL REPORT NUMBER: # 3

*THE WORK DESCRIBED HAS BEEN SUPPORTED BY THE AIR FORCE OFFICE OF SCIENTIFIC RESEARCH GRANT [redacted]
AFOSR-81-0220

# A Note on Computing the Asymptotic Form of a Limited Sequence of Decision Trees

Nicholas V. Findler, Neal Mazur, and Bede McCall
Group for Computer Studies of Strategies
Department of Computer Science
State University of New York at Buffalo

ABSTRACT

It is often desirable to compute the asymptotic form of a limited sequence of decision trees (DT's) after the set of decision variables has reached a stable, constant membership. Such is the case when the DT's characterize the evolving behavior of a learning strategy in the Quasi-Optimizer system under development.

An algorithm is described which builds the asymptotic decision tree in breadth-first order, node-by-node, starting at the root. In the course of computing the asymptotic value of a node (that is, its out-degree and the limiting points of its subranges), three situations may arise:

(i) the computation is possible;

(ii) the computation seems to be possible but more DT's have to be acquired of the evolving strategy to reach the desired level of statistical significance;

(iii) the computation is not possible.

## INTRODUCTION

Decision-trees (DT's) are often used to describe both deterministic and stochastic, multistage decision processes. Figure 1 shows schematically such a DT. The $x_i$'s are the decision variables, each associated with one level of the tree. The subrange into which the current value of the decision variable falls determines which branch emanating from the node at that level is followed. Thus any given combination of values of the decision variables leads to a particular terminal node (or leaf) associated with a certain action, $a_j$.

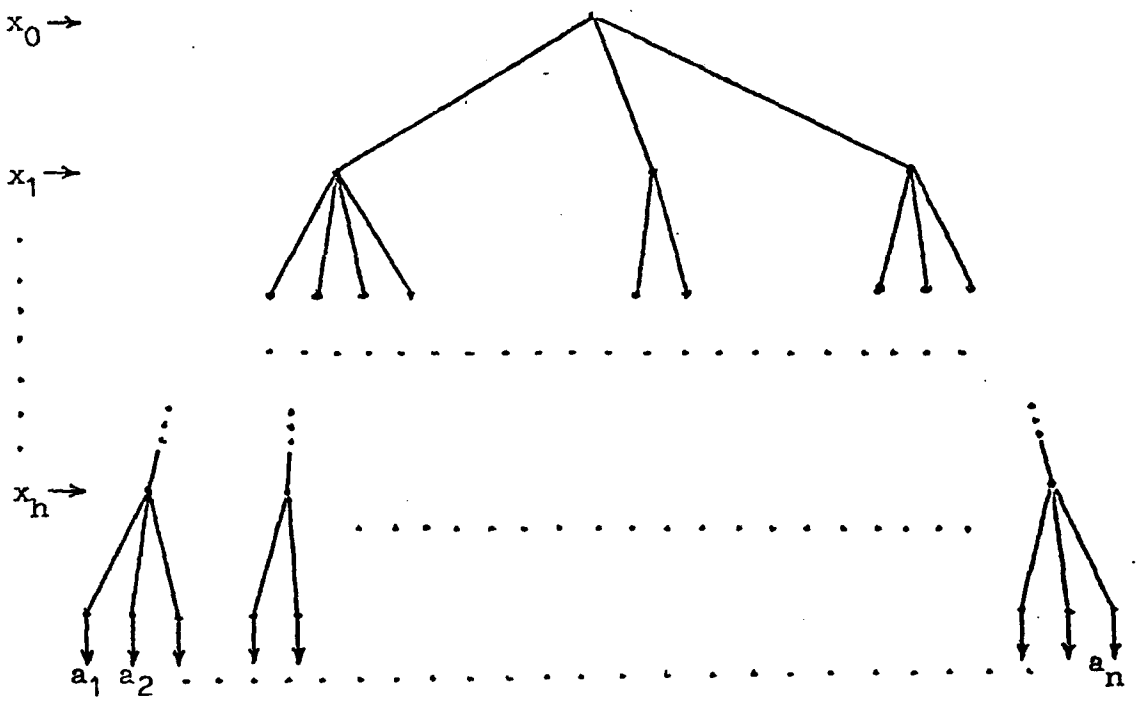
- - - - - - - - - - - - - - - - - - -

FIGURE 1 ABOUT HERE

- - - - - - - - - - - - - - - - - - -

In a previous article [1], we have defined the complexity of DT's and the power of heuristic rules employed in processing them. We have also outlined the Quasi-Optimizer (QO) system under development, which aims at generating automatically descriptive theories of competitive strategies and a normative theory that would be optimum in the statistical sense against a given set of strategies.

We have reported in a second article [2] on the completion of the first phase of the QO project. We have constructed a program called QO1, which builds a model, in the form of a DT, of a static strategy given as an impenetrable "black box" program. The strategy being modelled is asked what it would do in situations covering the space of competition. A 'situation' is specified in terms of the history of confrontations and the current values of the decision variables that the strategy



considers relevant. The action selected by the strategy is one from among all legally possible actions. We assume for the sake of generality that the situation is described by three types of information: numerical (including rank numbers), symbolic (attributes, categories), and structural data (hierarchies, relationships, context-dependent priorities). The model building program accepts as input an ordered list of all possible decision variables as well as the total range of and the minimum discernible difference between the values of each decision variable. The program can produce the result in two forms: the Minimum-Depth Decision Tree (MDDT) or the Normalized Decision Tree (NDT). The former contains levels associated with only those decision variables that are found to be statistically relevant for the strategy being modelled. The latter contains the union of the sets of variables represented in the MDDT's of several strategies and, therefore, all NDT's of these strategies are of uniform depth. This is accomplished by inserting levels of non-branching nodes (out-degree 1) in an MDDT corresponding to decision variables relevant to at least one other strategy. In the present paper, a DT will refer to a single strategy and will be of the MDDT-type.

One of the several problems arising in the QO project concerns strategies that vary over time. Such phenomena could be random fluctuations, cyclic variations, a monotonic tendency or some combinations of the above. The case of monotonic tendencies is particularly important since learning strategies exhibit such behavior. (There is, of course, also some additional noise due to different reasons, such as latent variables and stochastic components in the environment, measurement errors in the

modelling process, etc.)

In order to generate the normative theory against its competitors, it is necessary for the QO program to know the asymptotic form (AF) of learning strategies. One would like to be able to compute a sufficiently precise estimate of the AF rather than to wait indefinitely long until a learning strategy becomes "reasonably" constant. We describe in the following a technique for computing such an estimate.

## THE TECHNIQUE

Let the QO program be able to interrupt the learning process at certain time points, "freeze" the strategy in that form and take a "snapshot" of it, that is let QO1 construct a sequence of descriptive theories of the evolving strategy as a DT. Suppose we have $k$ such DT's constructed so far. We would like to make one of the following three statements on that basis:

(i) The AF can be expressed as a computable DT.

(ii) Take $m$ more snapshots as the statistical evidence so far is "promising" but not sufficient.

(iii) There is no statistical evidence so far that the (assumedly) learning strategy converges to a computable one.

There are three entities in DT's that can vary over time:

- the members of the set of decision variables that appear to be statistically relevant in any given snapshot;
- the out-degrees of every node in a DT;
- the location of the limiting points between the subranges each of which corresponds to a branch emanating from the parent node. (See Figure 2; the total range of

the values of decision variable x is normalized to 0-100.)

- - - - - - - - - - - - - - - - - - - - - - -

FIGURE 2 ABOUT HERE

- - - - - - - - - - - - - - - - - - - - - - -

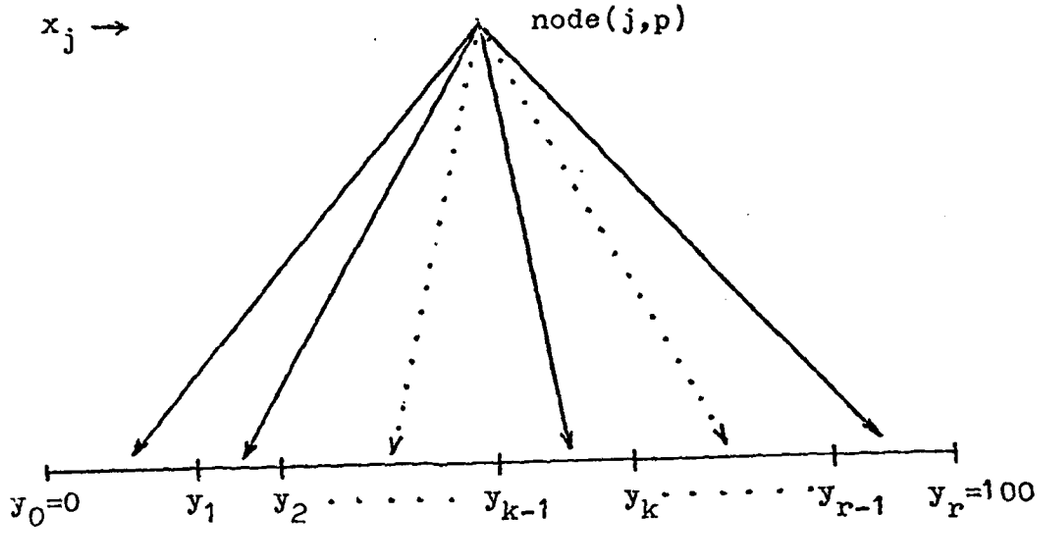
It is obvious that the above three entities are strongly related -- for example, one cannot refer to a stable condition concerning the subranges of the decision variable values unless the out-degrees of the parent nodes are constant over time.

Let us for the sake of explanation assume that the total range of each decision variable is normalized to (0, 100). (This also means that symbolic and structural data can be mapped onto such a numerical scale.)

The algorithm used is now described informally. The Appendix contains a more formal version written in a PASCAL-like communication language, augmented with some comments.

Suppose a learning strategy improves its level of performance through a series of confrontations. Let T confrontations comprise a cycle. After each cycle, learning is "turned off" and Q01 takes a snapshot of the strategy rendered temporarily static. The user estimates that K such snapshots (i.e. learning through K*T confrontations) will be sufficient to compute the asymptotic form of the DT's.

If the set of decision variables has reached a stable, constant membership only in L snapshots, out of the total K(>L) snapshots, some more cycles of confrontation are needed. Assuming a binomial distribution with p=L/K, the estimated number of additional snapshots to be taken is


$x_j \rightarrow$
node(j,p)
$y_0=0$
$y_1$
$y_2$
$y_{k-1}$
$y_k$
$y_{r-1}$
$y_r=100$

E= K/L*(K-L)

in order to get a total of about K snapshots with constant membership. (The most frequently occurring set of decision variables is considered the asymptotic membership.) If the membership is still unstable after having tried taking the additional sequence of snapshots, the computation is abandoned.

After a constant set of decision variables is found in a sufficient number of snapshots, the gradual build-up of the asymptotic decision tree (ADT) takes place, node-by-node, breadth-first, starting at the node. For every node being considered, the result of the computation can be

(i) satisfactory, in which case it is appended to ADT built so far;

(ii) satistically "promising" but not significant enough, in which case additional snapshots are taken the number of which is estimated on the basis of the convergence rate obtained;

(iii) unsatisfactory, in which case failure is reported.

A node, characterized by two subscripts, its level number and its ordinal number from the left, is "computed" when the asymptotic values of the out-degree (the number of branches emanating from it) and of the location of the boundary points between adjacent subranges of the decision variable in question are known. These numerical values are found by using for each a function EXPOL that extrapolates from known values. It fits two "envelope" curves of the $y=a_1.\exp(-c_1.x)+b_1$ and $y=a_2.\exp(-c_2.x)+b_2$ type. (The envelope curves contain in between the two of them all datapoints.) The asymptotic value sought is $y_\infty= (b_1+b_2)/2$ . In fact, the computational result referred to above,

"statistically promising", is related to a rate of convergence less than a threshold value. (We note that the actual method of fitting the envelope functions is somewhat more complicated than the algorithm in the Appendix suggests.)

Once the asymptotic value of the node's out-degree has been found, it is imposed to every corresponding node in the sequence of snapshots. This is done by using the principle of maximum sensitivity over the whole range of the decision variable. Namely, when the snapshot node's out-degree is less than the asymptotic value, the longest subrange of the node variable is halved, with both halves getting the same subtree, until the out-degree so obtained equals the asymptotic value. In the opposite case, in which the snapshot node's out-degree is to be reduced, those adjacent subranges are merged whose combined length is the shortest, until the out-degree so obtained is equal to the asymptotic value. The newly created subrange gets the subtree of that original subrange of the two which was the longer one. (The other disappears.)

The boundary points between these "synthetic" subranges are then extrapolated by EXPOL to complete the calculation of the asymptotic node, which is finally appended onto the ADT being built.

## SOME EXAMPLES

The algorithm, outlined in a PASCAL-like language in the APPENDIX, was programmed in LISP. However, one part of the program, which does the extrapolation of a sequence of integers, is in FORTRAN. (In fact, the routine that fits exponential curves has been adopted from the BMDP package [3].)

The interface between LISP and FORTRAN is implemented in a way appropriate for tasks in which an infrequent exchange of a small number of numerical values is needed between two programs in the respective languages. (The interface is briefly described in the paper on Q01, [2].) LISP establishes a file for data exchange and provides on it two lists of integers, the X and Y values of the datapoints to be extrapolated (as well as a control statements file to be executed by the system). The FORTRAN program then takes over, fits the envelope functions as defined in the Appendix, and computes the asymptotic value of the sequence of integers. (The algorithm used for it is somewhat complicated and is beyond the scope of this paper.) The value returned is either an integer, in case of convergence; the message 'PROMISING', in case the asymptotic value cannot (as yet) be computed but the rate of convergence is more than a certain threshold value; the message 'UNDEFINED' in case there is no well-defined, unique value to which the sequence tends; or the message 'DIVERGENT' if that is the case.

The sequence of trees, the "snapshots", were not generated by Q01 for the examples below. In order to be able to show the performance of Q02 in all possible cases, we have programmed tree-generating functions which are flexible and easily adjustable so that we could compare the results of Q02 with the outcomes expected a priori.

The output by the program Q02 consists of three types of information:

.the sequence of DT's input;

.the calls to and results of the EXTRAPOLATE function;

.the final result, that is the asymptotic tree -- possibly preceded by the message 'PROMISING' and additional DT snapshots yielding a convergent sequence of trees -- or one of the messages 'UNDEFINED' and 'DIVERGENT'.

At each call to EXTRAPOLATE, the program also prints the sequence of integers to extrapolated, and parameters a , b and -c of the upper and lower envelope functions fitted. These are of the form

$$y = a \cdot \exp(-c \cdot x) + b$$

The trees are printed in a format corresponding to their internal representation. The list of nodes at each level is prefixed by the symbolic name of the decision variable. The values of these decision variables are discriminated at the parent node of this level. The branches emanating from the parent node point to the subranges of values, as shown on Fig. 2. The name of the decision variable is followed by a list of nodes with each node consisting of two lists. The first one consists of the parent node's position description followed by the rank-number-from-the-left of the node in question (among the parent node's children). The root node's position description is (0 1). Therefore, the position description of the level 1 nodes are : (0 1 1), (0 1 2), (0 1 3), ... . The second list of the node gives the lower and upper boundary points of the subrange of values of this decision variable that the node represents. The subrange specification is replaced at the last (leaf) level by the strategy response. (The reserved symbol 'R' is then in the place of the decision variable name.)

Case Numbers 1 and 2: Result Promising, and then Convergent

```
(SNAPSHOTS TESTPR)


    SNAPSHOT  1

((A (0 1))
 (C ((0 1 1) (1 31)) ((0 1 2) (32 100)))
 (D ((0 1 1 1) (1 69))
    ((0 1 1 2) (70 100))
    ((0 1 2 1) (1 13))
    ((0 1 2 2) (14 26))
    ((0 1 2 3) (27 100)))
 (R ((0 1 1 1 1) (15))
    ((0 1 1 2 1) (19))
    ((0 1 2 1 1) (54))
    ((0 1 2 2 1) (55))
    ((0 1 2 3 1) (85))))

    SNAPSHOT  2

((A (0 1))
 (B ((0 1 1) (1 25)) ((0 1 2) (26 49)) ((0 1 3) (50 100)))
 (C ((0 1 1 1) (1 56))
    ((0 1 1 2) (57 78))
    ((0 1 1 3) (79 100))
    ((0 1 2 1) (1 56))
    ((0 1 2 2) (57 78))
    ((0 1 2 3) (79 100))
    ((0 1 3 1) (1 12))
    ((0 1 3 2) (13 100)))
 (D
  ((0 1 1 1 1) (1 35))
  ((0 1 1 1 2) (36 100))
  ((0 1 1 2 1) (1 43))
  ((0 1 1 2 2) (44 100))
  ((0 1 1 3 1) (1 43))
  ((0 1 1 3 2) (44 100))
  ((0 1 2 1 1) (1 35))
  ((0 1 2 1 2) (36 100))
  ((0 1 2 2 1) (1 43))
  ((0 1 2 2 2) (44 100))
  ((0 1 2 3 1) (1 43))
  ((0 1 2 3 2) (44 100))
  ((0 1 3 1 1) (1 78))
  ((0 1 3 1 2) (79 100))
  ((0 1 3 2 1) (1 88))
  ((0 1 3 2 2) (89 100)))
 (R
  ((0 1 1 1 1 1) (7))
  ((0 1 1 1 2 1) (22))
  ((0 1 1 2 1 1) (23))
  ((0 1 1 2 2 1) (29))
  ((0 1 1 3 1 1) (24))
  ((0 1 1 3 2 1) (30))
```

```
  ((0 1 2 1 1 1) (8))
  ((0 1 2 1 2 1) (23))
  ((0 1 2 2 1 1) (24))
  ((0 1 2 2 2 1) (30))
  ((0 1 2 3 1 1) (25))
  ((0 1 2 3 2 1) (31))
  ((0 1 3 1 1 1) (43))
  ((0 1 3 1 2 1) (63))
  ((0 1 3 2 1 1) (67))
  ((0 1 3 2 2 1) (88))))

        . . .

     SNAPSHOT  18

((A (0 1))
 (B ((0 1 1) (1 43)) ((0 1 2) (44 100)))
 (C ((0 1 1 1) (1 57))
    ((0 1 1 2) (58 100))
    ((0 1 2 1) (1 31))
    ((0 1 2 2) (32 100)))
 (R ((0 1 1 1 1) (15))
    ((0 1 1 2 1) (31))
    ((0 1 2 1 1) (61))
    ((0 1 2 2 1) (83))))

     SNAPSHOT  19

((A (0 1))
 (B ((0 1 1) (1 38)) ((0 1 2) (39 100)))
 (C ((0 1 1 1) (1 59))
    ((0 1 1 2) (60 100))
    ((0 1 2 1) (1 31))
    ((0 1 2 2) (32 100)))
 (R ((0 1 1 1 1) (14))
    ((0 1 1 2 1) (29))
    ((0 1 2 1 1) (59))
    ((0 1 2 2 1) (83))))

     SNAPSHOT  20

((A (0 1))
 (B ((0 1 1) (1 42)) ((0 1 2) (43 100)))
 (C ((0 1 1 1) (1 61))
    ((0 1 1 2) (62 100))
    ((0 1 2 1) (1 30))
    ((0 1 2 2) (31 100)))
 (R ((0 1 1 1 1) (13))
    ((0 1 1 2 1) (29))
    ((0 1 2 1 1) (60))
    ((0 1 2 2 1) (84))))
```

```
?(RUNPR)

THE STABLE SET OF DECISION VARIABLES IS:  (A B C R) .

THEREFORE, THE SNAPSHOTS WITH ORDINAL NUMBERS IN (1 2 6 16) HAVE
BEEN TEMPORARILY DELETED.


          CALL  TO  EXTRAPOLATE

EXTRAPOLATING THE OUTDEGREE OF THE NODE WITH
RANK-FROM-LEFT 1 IN LEVEL 0 (DECISION VARIABLE A ).

SEQUENCE OF INTEGERS : (4 3 3 3 2 2 2 2 2 2 2 2 2 2 2)

 --------  <ASYMPTOTIC VALUE =          1.8255 >

EXTRAPOLATE RETURN VALUE : 2


          CALL  TO  EXTRAPOLATE

EXTRAPOLATING THE SUBRANGE OF THE NODE WITH
RANK-FROM-LEFT 1 IN LEVEL 1 (DECISION VARIABLE B ).

SEQUENCE OF INTEGERS : (57 26 26 53 50 32 35 45 46 36 44 38 37 43 38 42)


 ------ ENVELOPE PARAMETER VALUES ------

     HIGH:      29.301661       -.040230      40.500000

     LOW :     -26.918121       -.152155      38.578933

 ----- DERIVATIVES AT ENDPOINTS -----

   UPPER =        -.527241     LOWER =         .195312

 --------  <PROMISING>

EXTRAPOLATE RETURN VALUE : PROMISING
```

```
ADDING 2 MORE TREES


    SNAPSHOT  21

((A (0 1))
 (B ((0 1 1) (1 43)) ((0 1 2) (44 100)))
 (C ((0 1 1 1) (1 59))
    ((0 1 1 2) (60 100))
    ((0 1 2 1) (1 31))
    ((0 1 2 2) (32 100)))
 (R ((0 1 1 1 1) (16))
    ((0 1 1 2 1) (29))
    ((0 1 2 1 1) (61))
    ((0 1 2 2 1) (84))))

    SNAPSHOT  22

((A (0 1))
 (B ((0 1 1) (1 40)) ((0 1 2) (41 100)))
 (C ((0 1 1 1) (1 58))
    ((0 1 1 2) (59 100))
    ((0 1 2 1) (1 31))
    ((0 1 2 2) (32 100)))
 (R ((0 1 1 1 1) (13))
    ((0 1 1 2 1) (30))
    ((0 1 2 1 1) (60))
    ((0 1 2 2 1) (83))))



      . . .



          CALL  TO  EXTRAPOLATE

EXTRAPOLATING THE RESPONSE OF THE NODE WITH
RANK-FROM-LEFT 1 IN LEVEL 3 .

SEQUENCE OF INTEGERS : (4 23 22 9 10 18 12 15 16 14 14 16 13 15 14 13 16
 13)
```

```
------ ENVELOPE PARAMETER VALUES ------

    HIGH:      24.868774      -.278260      15.967479

    LOW :     -10.714954      -.131484      13.268462

----- DERIVATIVES AT ENDPOINTS -----

  UPPER =        -.015187     LOWER =         .078091

--------  <ASYMPTOTIC VALUE =         14.6180 >

EXTRAPOLATE RETURN VALUE : 15


      . . .

          CALL  TO  EXTRAPOLATE

EXTRAPOLATING THE RESPONSE OF THE NODE WITH
RANK-FROM-LEFT 4 IN LEVEL 3 .

SEQUENCE OF INTEGERS : (98 94 75 76 89 77 80 87 82 79 85 83 84 83 83 84
84 83)


------ ENVELOPE PARAMETER VALUES ------

    HIGH:      17.338924      -.095370      83.666667

    LOW :     -19.183673      -.118447      83.113417

----- DERIVATIVES AT ENDPOINTS -----

  UPPER =        -.202873     LOWER =         .167787

--------  <ASYMPTOTIC VALUE =         83.3900 >

EXTRAPOLATE RETURN VALUE : 83
```

THE ASYMPTOTIC FORM OF THE DECISION TREES IS:

```
((A (0 1))
 (B ((0 1 1) (1 40)) ((0 1 2) (41 100)))
 (C ((0 1 1 1) (1 59))
    ((0 1 1 2) (60 100))
    ((0 1 2 1) (1 31))
    ((0 1 2 2) (32 100)))
 (R ((0 1 1 1 1) (15))
    ((0 1 1 2 1) (29))
    ((0 1 2 1 1) (61))
    ((0 1 2 2 1) (83))))
```

## Case Number 3: Result Undefined

```
?(SNAPSHOTS TESTUN)


   SNAPSHOT  1

((A (0 1))
 (B ((0 1 1) (1 54)) ((0 1 2) (55 77)) ((0 1 3) (78 100)))
 (D ((0 1 1 1) (1 52))
    ((0 1 1 2) (53 100))
    ((0 1 2 1) (1 8))
    ((0 1 2 2) (9 100))
    ((0 1 3 1) (1 8))
    ((0 1 3 2) (9 100)))
 (R ((0 1 1 1 1) (16))
    ((0 1 1 2 1) (18))
    ((0 1 2 1 1) (52))
    ((0 1 2 2 1) (95))
    ((0 1 3 1 1) (53))
    ((0 1 3 2 1) (96))))

   SNAPSHOT  2

((A (0 1))
 (C ((0 1 1) (1 10)) ((0 1 2) (11 20)) ((0 1 3) (21 100)))
 (D ((0 1 1 1) (1 74))
    ((0 1 1 2) (75 100))
    ((0 1 2 1) (1 74))
    ((0 1 2 2) (75 100))
    ((0 1 3 1) (1 6))
    ((0 1 3 2) (7 100)))
 (R ((0 1 1 1 1) (18))
    ((0 1 1 2 1) (35))
    ((0 1 2 1 1) (19))
    ((0 1 2 2 1) (36))
    ((0 1 3 1 1) (68))
    ((0 1 3 2 1) (91))))

   SNAPSHOT  3

((A (0 1))
 (B ((0 1 1) (1 21)) ((0 1 2) (22 41)) ((0 1 3) (42 100)))
 (C ((0 1 1 1) (1 12))
    ((0 1 1 2) (13 24))
    ((0 1 1 3) (25 100))
    ((0 1 2 1) (1 12))
    ((0 1 2 2) (13 24))
    ((0 1 2 3) (25 100))
    ((0 1 3 1) (1 60))
    ((0 1 3 2) (61 70))
    ((0 1 3 3) (71 80))
    ((0 1 3 4) (81 100)))
 (R ((0 1 1 1 1) (31))
    ((0 1 1 2 1) (32))
    ((0 1 1 3 1) (35))
    ((0 1 2 1 1) (32))
```

. . .

SNAPSHOT 20

```
((A (0 1))
 (B ((0 1 1) (1 29)) ((0 1 2) (30 100)))
 (C ((0 1 1 1) (1 37))
    ((0 1 1 2) (38 100))
    ((0 1 2 1) (1 41))
    ((0 1 2 2) (42 100)))
 (R ((0 1 1 1 1) (20))
    ((0 1 1 2 1) (44))
    ((0 1 2 1 1) (56))
    ((0 1 2 2 1) (89))))
```

SNAPSHOT 21

```
((A (0 1))
 (B ((0 1 1) (1 30)) ((0 1 2) (31 100)))
 (C ((0 1 1 1) (1 36))
    ((0 1 1 2) (37 100))
    ((0 1 2 1) (1 25))
    ((0 1 2 2) (26 100)))
 (R ((0 1 1 1 1) (16))
    ((0 1 1 2 1) (45))
    ((0 1 2 1 1) (59))
    ((0 1 2 2 1) (92))))
```

SNAPSHOT 22

```
((A (0 1))
 (B ((0 1 1) (1 28)) ((0 1 2) (29 100)))
 (C ((0 1 1 1) (1 40))
    ((0 1 1 2) (41 100))
    ((0 1 2 1) (1 22))
    ((0 1 2 2) (23 100)))
 (R ((0 1 1 1 1) (19))
    ((0 1 1 2 1) (47))
    ((0 1 2 1 1) (58))
    ((0 1 2 2 1) (88))))
```

```
?(RUNUN)

THE STABLE SET OF DECISION VARIABLES IS:  (A B C R) .

THEREFORE, THE SNAPSHOTS WITH ORDINAL NUMBERS IN (1 2 8 19) HAVE
BEEN TEMPORARILY DELETED.


          CALL  TO  EXTRAPOLATE

EXTRAPOLATING THE OUTDEGREE OF THE NODE WITH
RANK-FROM-LEFT 1 IN LEVEL 0 (DECISION VARIABLE A ).

SEQUENCE OF INTEGERS : (3 3 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2)


--------  <ASYMPTOTIC VALUE =          1.8871 >

EXTRAPOLATE RETURN VALUE : 2


        . . .


          CALL  TO  EXTRAPOLATE

EXTRAPOLATING THE SUBRANGE OF THE NODE WITH
RANK-FROM-LEFT 1 IN LEVEL 2 (DECISION VARIABLE C ).

SEQUENCE OF INTEGERS : (24 29 46 46 32 45 41 37 42 35 36 42 35 41 41 37
36 40)


------ ENVELOPE PARAMETER VALUES ------


     HIGH:      13.458370      -.139997      41.182370

     LOW :     -13.539677      -.152142      36.186914

----- DERIVATIVES AT ENDPOINTS -----

   UPPER =        -.086599     LOWER =         .072481

--------  <ASYMPTOTIC VALUE =         38.6846 >

EXTRAPOLATE RETURN VALUE : 39
```

. . .

CALL TO EXTRAPOLATE

EXTRAPOLATING THE SUBRANGE OF THE NODE WITH
RANK-FROM-LEFT 3 IN LEVEL 2 (DECISION VARIABLE C ).

SEQUENCE OF INTEGERS : (60 55 15 19 47 20 21 43 23 22 43 44 25 22 43 41 25 22)

------ ENVELOPE PARAMETER VALUES ------

| | | | |
|---|---|---|---|
| HIGH: | 20.923879 | -.127042 | 42.412263 |
| LOW : | -28.615767 | -.259073 | 22.095785 |

----- DERIVATIVES AT ENDPOINTS -----

| | | | |
|---|---|---|---|
| UPPER = | -.162469 | LOWER = | .024815 |

-------- <UNDEFINED>

EXTRAPOLATE RETURN VALUE : UNDEFINED

(TREES FAIL TO CONVERGE AS THE ASYMPTOTIC TREE IS UNDEFINED)

Case Number 4: Result Divergent

```
?(SNAPSHOTS TESTDI)


   SNAPSHOT  1

((A (0 1))
 (B ((0 1 1) (1 27)) ((0 1 2) (28 100)))
 (C ((0 1 1 1) (1 69))
    ((0 1 1 2) (70 85))
    ((0 1 1 3) (86 100))
    ((0 1 2 1) (1 11))
    ((0 1 2 2) (12 100)))
 (D
  ((0 1 1 1 1) (1 27))
  ((0 1 1 1 2) (28 100))
  ((0 1 1 2 1) (1 9))
  ((0 1 1 2 2) (10 18))
  ((0 1 1 2 3) (19 100))
  ((0 1 1 3 1) (1 9))
  ((0 1 1 3 2) (10 18))
  ((0 1 1 3 3) (19 100))
  ((0 1 2 1 1) (1 62))
  ((0 1 2 1 2) (63 100))
  ((0 1 2 2 1) (1 70))
  ((0 1 2 2 2) (71 100)))
 (R
  ((0 1 1 1 1 1) (13))
  ((0 1 1 1 2 1) (25))
  ((0 1 1 2 1 1) (42))
  ((0 1 1 2 2 1) (43))
  ((0 1 1 2 3 1) (30))
  ((0 1 1 3 1 1) (43))
  ((0 1 1 3 2 1) (44))
  ((0 1 1 3 3 1) (31))
  ((0 1 2 1 1 1) (47))
  ((0 1 2 1 2 1) (53))
  ((0 1 2 2 1 1) (78))
  ((0 1 2 2 2 1) (75))))

   SNAPSHOT  2

((A (0 1))
 (B ((0 1 1) (1 54)) ((0 1 2) (55 77)) ((0 1 3) (78 100)))
 (D ((0 1 1 1) (1 52))
    ((0 1 1 2) (53 100))
    ((0 1 2 1) (1 8))
    ((0 1 2 2) (9 100))
    ((0 1 3 1) (1 8))
    ((0 1 3 2) (9 100)))
 (R ((0 1 1 1 1) (16))
    ((0 1 1 2 1) (18))
    ((0 1 2 1 1) (52))
    ((0 1 2 2 1) (95))
    ((0 1 3 1 1) (53))
    ((0 1 3 2 1) (96))))
```

. . .

SNAPSHOT 22

```
((A (0 1))
 (B ((0 1 1) (1 25)) ((0 1 2) (26 71)) ((0 1 3) (72 100)))
 (C ((0 1 1 1) (1 42))
    ((0 1 1 2) (43 100))
    ((0 1 2 1) (1 27))
    ((0 1 2 2) (28 100))
    ((0 1 3 1) (1 71))
    ((0 1 3 2) (72 100)))
 (R ((0 1 1 1 1) (24))
    ((0 1 1 2 1) (35))
    ((0 1 2 1 1) (45))
    ((0 1 2 2 1) (61))
    ((0 1 3 1 1) (77))
    ((0 1 3 2 1) (96))))
```

SNAPSHOT 23

```
((A (0 1))
 (B ((0 1 1) (1 26)) ((0 1 2) (27 72)) ((0 1 3) (73 100)))
 (C ((0 1 1 1) (1 44))
    ((0 1 1 2) (45 100))
    ((0 1 2 1) (1 29))
    ((0 1 2 2) (30 100))
    ((0 1 3 1) (1 68))
    ((0 1 3 2) (69 100)))
 (R ((0 1 1 1 1) (23))
    ((0 1 1 2 1) (97))
    ((0 1 2 1 1) (47))
    ((0 1 2 2 1) (59))
    ((0 1 3 1 1) (81))
    ((0 1 3 2 1) (93))))
```

SNAPSHOT 24

```
((A (0 1))
 (B ((0 1 1) (1 24)) ((0 1 2) (25 69)) ((0 1 3) (70 100)))
 (C ((0 1 1 1) (1 45))
    ((0 1 1 2) (46 100))
    ((0 1 2 1) (1 27))
    ((0 1 2 2) (28 100))
    ((0 1 3 1) (1 68))
    ((0 1 3 2) (69 100)))
 (R ((0 1 1 1 1) (26))
    ((0 1 1 2 1) (37))
    ((0 1 2 1 1) (45))
    ((0 1 2 2 1) (61))
    ((0 1 3 1 1) (76))
    ((0 1 3 2 1) (95))))
```

```
?(RUNDI)

THE STABLE SET OF DECISION VARIABLES IS:  (A B C R) .

THEREFORE, THE SNAPSHOTS WITH ORDINAL NUMBERS IN (1 2 8 15) HAVE
BEEN TEMPORARILY DELETED.


          CALL  TO  EXTRAPOLATE

EXTRAPOLATING THE OUTDEGREE OF THE NODE WITH
RANK-FROM-LEFT 1 IN LEVEL 0 (DECISION VARIABLE A ).

SEQUENCE OF INTEGERS : (4 4 4 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3)


 --------  <ASYMPTOTIC VALUE =          2.9186 >

EXTRAPOLATE RETURN VALUE : 3



          CALL  TO  EXTRAPOLATE

EXTRAPOLATING THE SUBRANGE OF THE NODE WITH
RANK-FROM-LEFT 1 IN LEVEL 1 (DECISION VARIABLE B ).

SEQUENCE OF INTEGERS : (41 13 18 33 22 30 32 22 25 29 27 25 24 28 26 25
27 25 26 24)


 ------ ENVELOPE PARAMETER VALUES ------


     HIGH:      15.223042       -.122477      27.527096

     LOW :     -64.396551       -.431457      22.559358

 ----- DERIVATIVES AT ENDPOINTS -----

   UPPER =         -.098621     LOWER =          .000884

 --------  <ASYMPTOTIC VALUE =         25.0432 >

EXTRAPOLATE RETURN VALUE : 25
```

. . .

CALL TO EXTRAPOLATE

EXTRAPOLATING THE RESPONSE OF THE NODE WITH
RANK-FROM-LEFT 1 IN LEVEL 3 .

SEQUENCE OF INTEGERS : (37 17 31 31 19 26 29 20 24 26 21 25 25 24 25 22 24 24 23 26)

------ ENVELOPE PARAMETER VALUES ------

| | | | |
|---|---|---|---|
| HIGH: | 12.922351 | -.140409 | 25.826385 |
| LOW : | -12.397016 | -.133827 | 22.844423 |

----- DERIVATIVES AT ENDPOINTS -----

UPPER = -.062409 LOWER = .066830

-------- <ASYMPTOTIC VALUE = 24.3354 >

EXTRAPOLATE RETURN VALUE : 24

CALL TO EXTRAPOLATE

EXTRAPOLATING THE RESPONSE OF THE NODE WITH
RANK-FROM-LEFT 2 IN LEVEL 3 .

SEQUENCE OF INTEGERS : (25 31 44 47 35 47 48 36 35 54 57 37 35 68 72 36 84 35 97 37)

-------- <DIVERGE>

EXTRAPOLATE RETURN VALUE : DIVERGE

(TREES DIVERGE)

## FINAL COMMENTS

The ability of the Quasi-Optimizer system has been extended so that it can create a model, a descriptive theory, of not only static but also learning strategies. It does so by extrapolating a sequence of snapshots taken of a learning strategy, rendered temporarily static, at different times.

## ACKNOWLEDGEMENTS

Paul Duerig and Edward Taussig participated in the early stages of programming QO2. The model building of static strategies, QO1, was programmed by J.P. Martins. The interface between LISP and FORTRAN was written by Don McKay. George Sicherman was a constant discussion partner and critic.

I am indebted to the Air Force Office of Scientific Research for their support, AFOSR Grant 81-0220.

## REFERENCES

[1] Findler, N.V., and van Leeuwen, J. (1979), On the complexity of decision trees, the Quasi-Optimizer, and the power of heuristic rules, Information and Control, 40, 1-19.

[2] Findler, N.V., and Martins, J.P. (1981), On automating computer model construction -- The second step toward a Quasi-Optimizer system, J. of Information and Optimization Sciences, 2, pp. 119-136.

[3] Dixon, W.J. and Brown, M.B. (Eds.) (1979), Biomedical Computer Programs P-Series, University of California Press; Berkeley, CA, 1979.

APPENDIX

## The Algorithm To Compute The Asymptotic Form of a Limited Sequence of Decision Trees

```
function Q02 ( function STRAT, Q01;
                integer snapshots-needed, learning-cycle-length);
{Q02 receives a learning strategy, STRAT, and a function Q01. Q01
makes a computer model, a "snapshot", of STRAT in the form of a
decision tree at the end of each learning cycle when learning is
temporarily disabled. The user also provides his estimate of how
many snapshots are needed, i.e. the length of the sequence of
decision trees to extrapolated, and how many confrontations are
needed in a learning cycle to make a significant change in the
strategy.}

sequence-of-trees := empty;
take-snapshots (snapshots-needed, learning-cycle-length,
                    sequence-of-trees);
find-stable-set-of-decision-variables (sequence-of-trees,
                        snapshot-numbers-of-trees-with-stable-set);
if result ≠ undefined
   then
     compute-asymptotic-tree (sequence-of-trees);
return (result)
end; {Q02}
```

```
procedure take-snapshots (snapshots-needed,
                          learning-cycle-length, sequence-of-trees);
repeat
   let-STRAT-learn-over-a-learning-cycle;
   take-a-snapshot-by-Q01;
   add-to-the-sequence-of-trees
until no-more-snapshots-needed;
return (sequence-of-trees)
end; {take-snapshots}
```

```
procedure find-stable-set-of-decision-variables (sequence-
          of-trees, snapshot-numbers-of-trees-with-stable-set);
maxvalue := 0;
classnumber := 0;
for each tree in sequence-of-trees do
   begin
     add-snapshot-number-of-tree-to-the-class-with-that-
       set-of-decision-variables;
     classnumber := classnumber + 1
   end;
{Establish classes of trees, each tree in a class having the
same set of decision variables.}
for each class of trees of constant set do
   if value-of-class (classnumber) > maxvalue
     then
       begin {record best class so far}
         maxclass := class (classnumber);
         maxvalue := value-of-class (classnumber)
       end; {record best class so far}
{The class with maximum value-of-class is selected.}
if maxvalue > class-threshold-value
  then
    begin {assign results}
      sequence-of-trees := maxclass;
      snapshot-numbers-of-trees-with-stable-set := snapshot-
        numbers-of-trees-in-maxclass
    end {assign results}
{If the value of the "best" class of trees was greater than a
threshold value, output the list of trees and the list of their
snapshot numbers in that class.}
```

```
else if (trees-added-more-than-twice)
        then result := undefined
        else
          begin {repeat with a longer sequence of trees}
            add-more-snapshots (sequence-of-trees, maxclass);
            find-stable-set-of-decision-variables (sequence-of-
              trees, snapshot-numbers-of-trees-with-stable-set)
          end {repeat with a longer sequence of trees}
```

{If the value of the best class was less than satisfactory, additional trees (snapshots) can be added twice to the original sequence of trees and the computation is repeated. Otherwise, the result of the computation of the asymptotic form is undefined.}

```
end; {find-stable-set-of-decision-variables}
```

```
function value-of-class (classnumber);
n := number-of-trees-in-class;
value := 0;
for each ordinal number of tree in the class, i, do
   value := value + exp(i);
value-of-class := c(n) * value + t(n)
end; {value-of-class}
```

{Here c(n) is a factor normalizing to one the sum of the weighting (recency) factors, $c(n) = (e - 1)/(\exp(n + 1) - e)$. The set of decision variables of more recent trees are more credible, closer to that of the asymptotic tree. The other term, t(n), assigns bigger weight to a class with more trees. Note that the snapshot number of a tree is its position in the total sequence of trees and not its position in the class.}

```
procedure add-more-snapshots (sequence-of-trees, maxclass);
k := total-number-of-trees-in-sequence;
n := number-of-trees-in-maxclass;
more-snapshots-needed := k * (k/n - 1) * class-threshold-value/
                value-of-class (maxclass);
take-snapshots (more-snapshots-needed, learning-cycle-length,
 sequence-of-trees)
end; {add-more-snapshots}
```

{The number of additional snapshots needed is calculated using the assumption of binomial distribution (see main text), which is then multiplied by the ratio between the minimum required and the current value of the sequence of trees to be extrapolated.}

```
function compute-asymptotic-tree (sequence-of-trees);
for each level of decision variable do
  for each node at this level do
    begin {asymptotic node computation}
      compute-asymptotic-form-of-node (level, rank-from-left);
      if result = asymptotic-node
        then append-to-asymptotic-tree (asymptotic-tree,
             asymptotic-node)
        else if result = promising and trees-added-less-than-twice
             then
               begin
                 add-more-trees (snapshots-needed-before,
                        rate-of-convergence);
                 compute-asymptotic-tree (sequence-of-trees)
               end
             else
               begin {set non-convergent result}
                 if result ≠ divergent
                   then result := undefined;
                 return (result)
               end {set non-convergent result}
    end {asymptotic node computation};
return (asymptotic-tree)
end; {compute-asymptotic-tree}
```

```
function compute-asymptotic-form-of-node (level, rank-from-left);
extrapolate(outdegree (node (level, rank-from-left), list-of-
                                    snapshot-numbers-of-trees-used);
if successful
  then
    begin {compute asymptotic node}
      asymptotic-outdegree := result;
      begin {compute asymptotic values of boundary points
              between subranges}
        impose-asymptotic-outdegree-on-corresponding-node-in-
          all-trees-used;
        repeat {for all subranges except last}
          extrapolate (upper-boundary-of-subrange
            (subrange-number, node (level, rank-from-left)),
             list-of-snapshot-numbers-of-trees-used);
          asymptotic-boundary-point := result;
          add-asymptotic-subrange-boundary-to-list-of-
            boundary-points
        until not successful or last subrange left
      end; {compute asymptotic values of boundary points
             between subranges}
      result := asymptotic-node
    end; {compute asymptotic node}
return (result)
end; {compute-asymptotic-form-of-node}
```

```
procedure
impose-asymptotic-outdegree-on-corresponding-node-in-all-trees-
    used (asymptotic-outdegree, level, rank-from-left,
          sequence-of-trees);
for each corresponding node do
   if node's-outdegree < asymptotic-outdegree
     then
       begin
         while outdegrees differ do
           begin
             halve-largest-subrange;
             copy-subtree-for-both-halves
           end
       end
     else
       if node's-outdegree > asymptotic-outdegree
         then
           begin
             while outdegrees differ do
               begin
                 combine-adjacent-subranges-with-shortest-
                   combined-length;
                 keep-subtree-of-longer-subrange-only
               end
           end
         else leave-subrange-of-node
end; {impose-asymptotic-outdegree-on-corresponding-node-in-all-
      trees-used}
```

```
function extrapolate (sequence-of-integer-values,
                      list-of-snapshot-numbers);
{This function returns either the asymptotic value to which the
given sequence of integers converges, if it does converge; or the
term 'PROMISING' if more datapoints are needed to decide; or the
term 'UNDEFINED' if no well-defined asymptotic value can be found;
or the term 'DIVERGENT' when that is the case.}
list-of-y-values := sequence-of-integer-values;
list-of-x-values := list-of-snapshot-numbers;
fit-upper-and-lower-envelope-functions (list-of-y-values,
                                        list-of-x-values);

{The upper envelope function is : Y1 = A1 * EXP (-C1 * X) + B1, and
the lower envelope function is : Y2 = A2 * EXP (-C2 * X) + B2,
where for all X : Y1(X) ≥ Y(X) ≥ Y2(X). The necessary and
sufficient conditions for the sequence of integer values to
converge are : C1 ≥ 0, C2 ≥ 0, and |B1 - B2| ≤ T1, a threshold
value.}
successful := false;
if C1 < 0 or C2 < 0
  then result := divergent
  else if abs (B1 - B2) > T1
       then if abs (Y1'(XMAX)) > T2 or abs Y(Y2'(XMAX)) > T2
            then
              begin
                rate-of-convergence := abs (Y1'(XMAX) -
                                            Y2'(XMAX));
                result := promising
```

```
                end;
            else result := undefined;
        else
          begin
            result := round ((B1 + B2/2));
            successful := true
          end
return (result)
end; {extrapolate}
```

{If the sequence of integer values is not divergent and the absolute value of the first derivative of at least one envelope function at the last x-value is greater than (another) threshold value, T2, then the result is 'PROMISING' and the rate of convergence is equal to the absolute value of the difference between the first derivatives of the two envelope functions at the last x-value. Otherwise, if the final slopes of both envelope functions are too small while the gap between the final values is too large, the result is 'UNDEFINED'. Finally, in the convergent case, the extrapolated value is the (rounded-off) arithmetic mean of the asymptotic values of the two envelope functions. We note that fitting envelope functions is quite tricky and its description is beyond the scope of this paper.}

```
procedure add-more-trees (snapshots-needed-before, rate-of-
                                        convergence);
{This procedure is used when the result of an extrapolation to a
sequence of integer values was 'promising'. In contrast, the
procedure 'add-more-snapshots' is invoked when there are not
enough trees in the "best" class of trees with identical decision
variables.}
more-snapshots-needed := beta * snapshots-needed before/
                                        rate-of-convergence;
{The number of additional trees needed is directly proportional
to the number of snapshots taken so far and inversely
proportional to the rate of convergence found in the function
'extrapolate' producing 'promising' results. It should also be
noted that the proportionality factor, beta, is domain-dependent.
Its optimum value is a function of the trade-off between the
costs of invoking Q01, i.e. taking one more snapshot, and of
recomputing the envelope functions. If the former is high,
relatively speaking, the smallest possible number of additional
snapshots should be taken. If the relative cost of recomputing
envelope functions for all the out-degree and boundary point
values is high, it is better to overestimate the additional
number of necessary snapshots.}
take-snapshots (more-snapshots-needed, learning-cycle-length,
                        sequence-of-trees);
end; {add-more-trees}
```

LEGEND FOR THE FIGURES

FIGURE 1 -- Schematic Representation of a Static Decision Tree.

Each level of the tree is identified with one of the decision variables $x_0, x_1, \ldots x_h$. The leaves attached to the branches at the last level, $a_1$, $a_2$, ... represent actions. A path down from to the root to an action in the decision tree is defined by a particular combination of values of the decision variables and characterizes the environment as perceived by the strategy which is represented by the decision tree.

FIGURE 2 -- A Node and Its Descendant Branches.

The range of the decision variable $x_j$, associated with the level of the node, $j$, is normalized to (0, 100). The node shown is the $p$-th from the left at the $j$-th level. The branch emanating from the node points to the $k$-th subrange from the left if $y_{k-1} < x_j \leq y_k$ with $1 \leq k \leq r$ and $y_0 = 0$, $y_r = 100.0$.


DAT
FILM