LUX ET VERITAS

SYSTEMS AND INFORMATION SCIENCES

DEPARTMENT OF ENGINEERING
AND APPLIED SCIENCE

YALE UNIVERSITY

NEW HAVEN, CONNECTICUT

APPLICATION OF A HIERARCHICAL STRUCTURE STOCHASTIC LEARNING AUTOMATON

R. G. Neville, M. S. Chrystall and P. Mars

S & IS Report No. 7906

September, 1979

Application of a Hierarchical Structure Stochastic Learning Automaton

R. G. Neville, M. S. Chrystall, and P. Mars

1. Introduction.

One of the principal application areas for stochastic computing research[1,2,3] is the implementation of adaptive control systems using stochastic learning automata (SLA) structures. A learning automaton is ideally suited to the problem of parameter optimisation of a noisy multimodal system (figure 1), since the inherent principle of random search avoids the effect of locking-on to local optima unavoidable with normal gradient methods. The automaton, in conjunction with a suitable interface, interacts with its environment in a manner analogous to a conventional feedback control system to evolve a 'suitable' final structure (figure 2).

By combining the results of earlier work in hardware stochastic computing systems[4,5] and extensive simulation studies of learning automaton behaviour,[6,7] it is possible to synthesise practical learning systems capable of on-line operation. Hardware designs for 2-state systems have subsequently been described[8,9] which verified that suitably fast learning behaviour was possible.

In order to implement large-scale systems, a hierarchical structure automaton was developed, using a 2-state SLA in a time-shared mode[10]. This system has already been tested using a simple simulated plant, and the work described here details the application to the more practical case of systems with multidimensional, multimodal performance criteria[11,12,13].

2. Review of Hierarchical System.

The hierarchical SLA evolved as a means of enabling a practical large-scale automaton to be constructed which would be capable of high-speed operation with the minimum of hardware.

The approach adopted was to time-share a single 2-state SLA in the tree-structure shown in figure 3. The random access memory, which stores the intermediate decision probabilities, fulfils the requirement for a memory in the automaton to establish the priority of state order during the learning period. Any one state or action probability is given by the product of the decision probabilities along the appropriate path through the tree. This configuration does of necessity involve more serial processing operations, but the savings in hardware are felt to far outweigh the speed penalty. Another advantage is that a modular construction greatly simplifies the design requirements of much larger systems. The algorithm circuit can retain the standard 4-term format used previously with 2-state systems[8,9], and is also time-shared at each level.

Using these principles, a hardware system with up to 128 states was constructed, based on the availability of a suitable commercial random access memory (RAM) with 128 bytes of storage. The memory requirements are determined by the number of levels in the system. The number of decision probabilities to be remembered at level p is 2^{p-1}; therefore, an r-level system (2^r states) requires a total RAM allocation of (2^r-1) bytes.

In the 2-state system (figure 4), the ADDIE[4] output is sampled by a flip-flop whose state then denotes the output action. In the hierarchical system application (figure 5), the state of this 'system flip-flop' is referred to as the 'decision bit', and the information is stored in a 'state latch' with one location per system level, as indicated in figure 6. The state latch thus acts as a small 'scratch-pad' memory, tracking the path taken through the tree during each cycle of operation by means of the stored decision bits, and also forms the basis of the memory address circuitry.

Overall control of the system is effected by means of a multiphase clock, which can be programmed to permit a choice of automaton size (via the number of levels), adjustable learning or ADDIE response time, and externally determined delay in output response time to cater for different plant time constants, thereby preserving fully the flexibility of design referred to above.

The design of a system with a non-binary multiple of states is more complicated. One possible solution, as yet untried, is to insert a decoder between the existing SLA and the plant. The SLA is organised with the next-highest binary number of states, and the decoder arranges for redundant states to be paired-off with 'active' states. This would cause initial bias in the learning behaviour, but the effect may not be too significant when averaged out over the learning period.

3. Preliminary Results.

The operating principle of the hierarchical system is illustrated by figure 7. This shows the learning characteristic of a 3-level, 8-state system converging to state 4. The learning time is typically three times that of a 2-state system, as would be expected from the time-shared nature of the learning process. The hierarchical SLA can, it is felt, be considered akin to a system of cooperative games[14] between 2-state automata, one at each level. Each automaton decides, in turn, which of the two locations below its own current position in the decision tree should be chosen, having been steered to that position by the automaton above it.

Initial optimisation experiments, reported previously,[10] were carried out using the most elementary of simulated plant circuits, as shown in figure 8. One selected action, in this case number 41, carries a low penalty probability, $c_\ell = 0.25$, and all others a higher one, $c_h = 0.875$. Figure 9 shows the resulting learning behaviour, presented in the form of an 'output map' derived by D/A conversion of the state latch contents. The approximate length of one iteration, using an 8-bit ADDIE and 2.5 MHZ master clock, is 50 μs, so that the indicated learning time of 50 ms corresponds to some 1,000 iterations.

Because of variance inherent in the ADDIE, there is always a small probability of incorrect decisions, at any level, beyond the initial learning period. This results in the sporadic occurrence of incorrect output actions which can be seen on the output map.

4. Application to Multimodal Systems.

In order to simulate a multimodal environment, a rather more sophisticated plant circuit is required. It was decided to use as an example a well-known P.I. function which has a clearly defined global optimum, local optimum and saddle-point.[15]

In order to match this P.I. surface to the SLA, a program (SLIG) was written which computed the function representing the P.I. surface and provided a choice of output options. In figure 10, the surface is plotted with a fine grid to illustrate its features. Figure 11 presents another view of the surface using a 16 x 8 grid to show how it is partitioned into 128 discrete elements. SLIG also generates a table of penalty probability values $\{c_i\}$ corresponding to each surface element, and writes these to a dump file. A third option is to produce a punched tape of binary c_i values for use with a custom-built PROM programmer for 2708-type PROMS (1K x 8 bits).

A useful feature of the program is the provision for a choice of compression factor applied to the range of c_i values derived from the P.I. function. This enables the values of the penalty probabilities to be constrained between chosen limits within the full scale range [0,1], to test the discriminatory powers of the SLA. For the learning runs reported here, the compression factor was set at 0.95, giving in this case a minimum c_i of 0.025 and a maximum of 0.85.

To examine the performance of the SLA under non-stationary conditions, it was arranged that a reflected version of the P.I. be stored in the next 128 bytes of the PROM (i.e. x & y - axes reversed). In this way, simply switching the most significant address line abruptly changes the plant seen by the SLA.

The configuration of the simulated plant is shown in figure 12. The PROM is at the centre, storing the c_i values as 8-bit numbers, each addressed by the appropriate action output from the SLA. The presence of noise on the surface is simply effected by interposing a full adder fed with noise derived from the central PRBS source. The resultant noise-corrupted value is then passed to a standard noise comparator arrangement which produces a stochastic pulse train whose 'value' represents the current penalty probability. This is then sampled by the penalty/reward flip-flop to produce the appropriate system response (0:reward, 1:penalty) to be fed to the algorithm circuit. The use of computer facilities in the preparations for these SLA experiments is summarised in figure 13.

In order to allow maximum flexibility in the presentation of data from SLA learning runs, it was decided to use a computerised data logging system. A program (SLOG) was written which recorded the output action after each iteration and wrote the values to a data file, together with relevant parameters for the particular experiment. A companion program (SLAG) was then written to process the data, together with the c_i file provided by SLIG, to form the output map (c.f. figure 9), penalty curve (i.e. a plot of actual received penalty against iteration number), and a set of cumulative distribution curves illustrating the evolution of automaton action at various stages during the learning process. Of this family of programs, SLAG and SLIG were written primarily for a main-frame system (DEC 20-40) while SLOG was run on an LSI 11/03 system. All three are in FORTRAN, though some MACRO routines are called as appropriate. The use of SLOG and SLAG in data presentation is summarised in figure 14.

The data logging process requires the SLA to interrupt the 11/03 each time an output action is present. The circuit used to accommodate this is shown in figure 15. The clock pulse which activates the system state output latch is differentiated to produce a narrow spike, shorter than the computer's interrupt response time. This sends an interrupt request via the flip-flop, which is subsequently reset by the reply signal from the computer.

5. Results and Comments.

A total of seven experiments were performed with the hierarchical SLA using the performance index described above with a superimposed noise component of $\pm\delta$ distributed uniformly over the surface. Four bits of noise were in fact added, so that δ represented 8 units or approximately 3% of the full-scale range (0-255) of the 8-bit c_i values. The results are detailed below with appropriate comments.

<u>Expt. 1</u>: 128-state, L_{R-P} scheme, $\alpha = 0.437$, $\beta = 0.992$.

The output map, penalty curve and distribution curves for this experiment are shown in figures 16,17,18 respectively. The form of the output map is essentially similar to figure 9, which was reproduced from an oscilloscope trace, and again convergence is obtained in around 1,000 iterations. This particular learning run gives convergence to the optimum action (100), while the effect of spurious switching to suboptimal actions, commented on earlier, shows up clearly on the penalty curve as transient 'spikes' to a higher level of received penalty.

<u>Expt. 2</u>: 128-state, L_{R-P} scheme, $\alpha = 0.25$, $\beta = 0.875$ (figures 19-21).

This experiment was chosen to illustrate the effect of a low ratio of reward to penalty (γ-factor). The output map presents a rather chaotic picture, as does the penalty curve to some extent. However, the distribution curves prove that the SLA is, in fact, selecting actions at or around the optimum, since a set of peaks is evident, spaced at intervals of 8, which accords with the partitioning of the surface into 16 x 8 elements for 128 actions.

This result, therefore, bears out the expected performance of an L_{R-P} scheme with a small measure of expediency, i.e. rapid, reliable convergence to a condition in which favourable actions are selected, though with probability somewhat short of unity.

<u>Expt. 3</u>: 128-state, L_{R-I} scheme, $\alpha = 0.25$ (figures 22-24).

This result is a classic example of an L_{R-I} automaton locking-on to the wrong action. The learning characteristics are very similar to those of expt. 1, but this time action 91 is selected. This illustrates the basic flaw in the L_{R-I} scheme, in that its high degree of expediency (ε-optimality) can result in an inability to escape from the situation where the wrong action is chosen, as may well occur with a non-stationary environment.

Expt. 4: 128-state, L_{R-P} scheme, $\alpha = 0.5$, $\beta = 0.992$ (figures 25-27).

While the above experiments lasted for 2,000 iterations, with static environment, this experiment was run for 5,000 iterations, with the plant switched after 2,048 iterations, as described earlier, under the control of a binary counter fed with state output latch clock pulses. In this particular case, the set-up was reversed so that the 'reflected' plant was chosen first.

The result shows convergence initially to actions 19 and 20, followed by an interim adjustment period, and culminates in convergence to the 'new' optimum of action 100. The first half of the experiment did not exhibit optimal behaviour, since action 28 would be preferred, but the overall result does nonetheless illustrate the ability of the SLA to track a nonstationary environment without excessive delay, provided an L_{R-P} reinforcement scheme is employed.

The following three experiments were chosen to illustrate the flexibility of the hierarchical SLA. By altering the programmable system clock referred to earlier, the size of the structure is immediately changed (in binary multiples).

Expt. 5: 64-state, L_{R-I} scheme, $\alpha = 0.125$ (figures 28-30).

In the case of the 64-state system, only every second grid point on the P.I. surface is addressed (odd numbers). The optimum action is then 101, which becomes 51 in the nomenclature of the 64-state system. This result demonstrates once again an L_{R-I} scheme locking-on to the wrong action, in this case 50.

Expt. 6: 32-state, L_{R-I} scheme, $\alpha = 0.125$ (figures 31-33).

This result for a 32-state system is essentially similar to Expt. 5, showing, in this case, convergence to action 24, while 26 is optimum.

Expt. 7: 16-state, L_{R-I} scheme, $\alpha = 0.125$ (figures 34-37).

This last experiment is significant, in that by addressing itself to only every eighth element of the P.I. surface, the 16-state SLA effectively sees only the rather shallow front edge (see figure 11). The corresponding penalty probability range here is only 0.71 to 0.85, which clearly presents a severe test of discrimination. The result obtained shows convergence to action 15, whereas 13 is the optimum. The penalty curve does, however, show a useful reduction in received penalty as a result of SLA action.

As the number of states is reduced, the learning time is clearly reduced also. Therefore, two sets of distribution curves were obtained for this experiment. The first set covers 2,000 iterations, as before, and shows that all significant activity is covered by the first 1,000 iterations. A second set (figure 37) was, therefore, constructed to cover this initial period, and these illustrate more clearly the evolution of the selected action.

In all of these experiments, a 12-bit ADDIE was used for greater accuracy and lower variance, at the expense of operating speed. Actual learning times for the above results can be estimated in the context of an approximate iteration time of 500 μs.

6. Conclusions.

The results of these experiments clearly indicate the power of the hierarchical SLA as a means of achieving rapid optimisation of a multimodal system, irrespective of contour, or of the presence of noise in the system. Even in cases where non-optimal convergence occurred, due to the use of a reinforcement algorithm where such behaviour is a known hazard, the automaton chose actions adjacent to the optimum and, therefore, performed its allotted task of reducing the average received penalty, thereby achieving a corresponding improvement in system performance approaching the optimum value.

It must be stressed that at no time did convergence to the local optimum occur, demonstrating that the SLA has purely altitude sensitivity over the P.I. surface, as opposed to the gradient sensitivity of conventional hill-climbing methods.

The result with a switched environment is particularly significant, since it is likely that many SLA applications will involve non-stationary plant. The presence of noise on the P.I. surface does not seem to impair significantly the performance of the SLA, and indeed it can be argued that its perturbating effect on the value of received penalty would help to dislodge a highly expedient learning scheme from an incorrect action to which it might otherwise lock-on. This would permit a slightly higher degree of expediency, which does have desirable features, to be catered for in the reinforcement scheme.

Further applications of the SLA will concentrate on two main areas: adaptive control and telephone traffic routing. Both of these topics have received attention in the past at simulation level, but the development of a viable hardware automaton should enable fully operational learning control systems to be demonstrated for these particular applications.

Acknowledgment

The research reported here was sponsored by the U. K. Science Research Council and supported in part by the National Science Foundation under Grant No. 03664.

References.

[1] Gaines, B. R., "Stochastic Computing," AFIPS SJCC, 1967, 30, pp. 149-156.

[2] Poppelbaum, W. J., Afuso, C., and Esch, J. W., "Stochastic Computing Elements and Systems," AFIPS FJCC, 1967, 31, pp. 635-644.

[3] Proc. 1st Int. Symp. on Stochastic Computing and its Applications, INPT, Toulouse, France, 1978.

[4] Miller, A. J. and Mars, P., "Optimal Estimation of Digital Stochastic Sequences," Int. J. Syst. Sci., 1977, 8, pp. 683-696.

[5] Miller, A. J. and Mars, P., "Theory and Design of a Digital Stochastic Computer Random Number Generator," Trans. IMACS, 1977, 19, pp. 198-216.

[6] Narendra, K. S. and Thathachar, M.A.L., "Learning Automata - A Survey," IEEE Trans., 1974, SMC-4, pp. 323-334.

[7] Viswanathan, R. and Narendra, K. S., "Expedient and Optimal Variable Structure Stochastic Automata," Becton Center, Yale University, 1970 Tech. Report CT-31.

[8] Neville, R. G., Nicol, C. R. and Mars, P., "Synthesis of Stochastic Learning Automata," Electron, Lett., 1978, 14, pp. 206-208.

[9] Neville, R. G., Nicol, C. R. and Mars, P., "Design of Stochastic Learning Automata using Adaptive Digital Logic Elements," ibid, 1978, 14, pp. 324-326.

[10] Neville, R. G. and Mars, P., "Hardware Design for a Hierarchical Structure Stochastic Learning Automaton," Journal of Cybernetics and Information Science, (in press).

[11] Shapiro, I.J.,and Narendra, K.S., "Use of Stochastic Automata for Parameter Self-Optimisation with Multimodal Performance Criteria," IEEE Trans. Syst. Sci. Cybern., SSC-5, Oct. 1969, pp. 352-360.

[12] Viswanathan, R.and Narendra, K:S., "Application of Stochastic Automata Models to Learning Systems with Multimodal Performance Criteria," Becton Center, Yale University, Tech. Report CT-40, June 1971.

[13] Jarvis, R.A., "Adaptive Global Search by the Process of Competitive Evolution," IEEE Trans. Syst. Sci. Cybern., SMC-5, May 1975, pp. 297-311.

[14] Viswanathan, R. and Narendrá, K. S., "Games of Stochastic Automata," IEEE Trans. Syst. Man & Cybernetics, SMC-4, 1974, pp. 131-135.

[15] Asai, K. and Kitajima, S., "A Method for Optimizing Control of Multimodal Systems using Fuzzy Automata," Inf. Sc., 3, 1971, pp. 343-353.

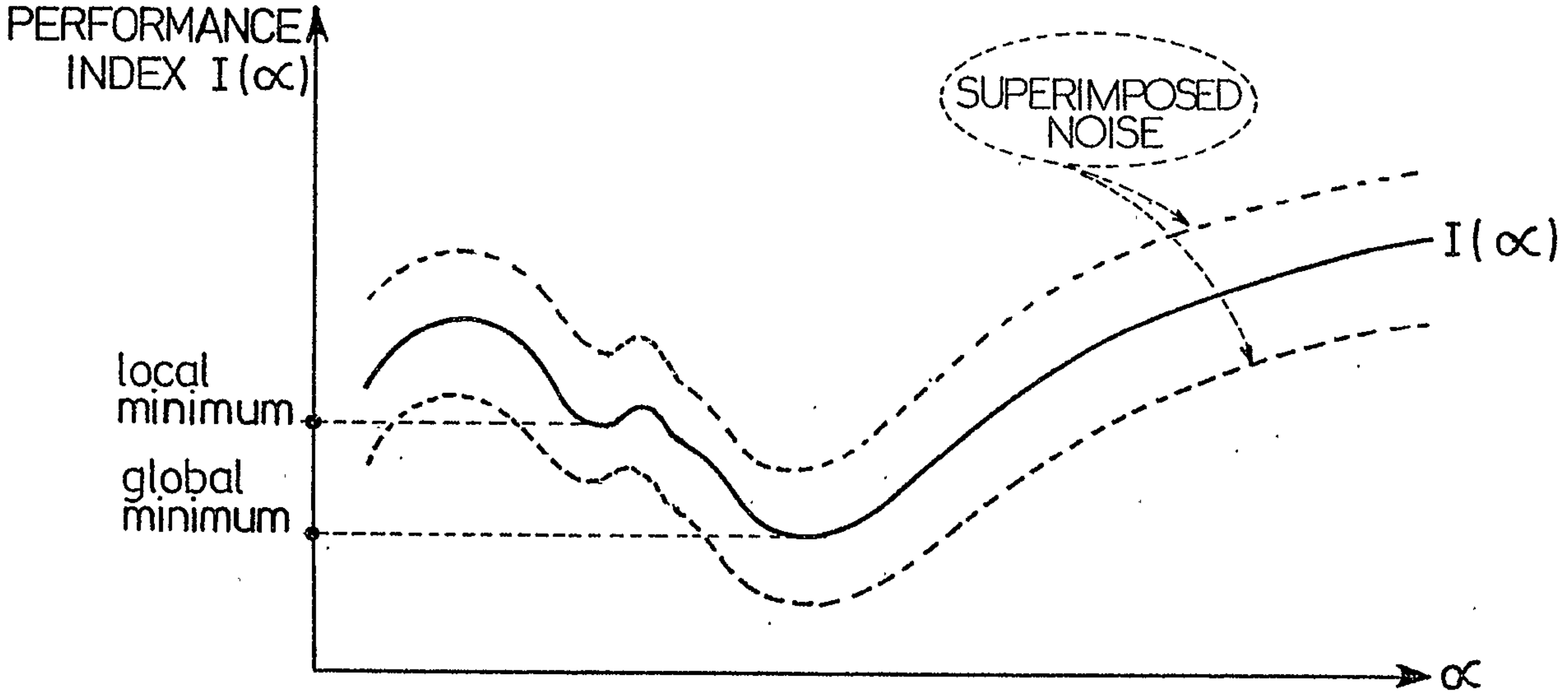


Figure 1 Multimodal Performance Index with Superimposed Noise

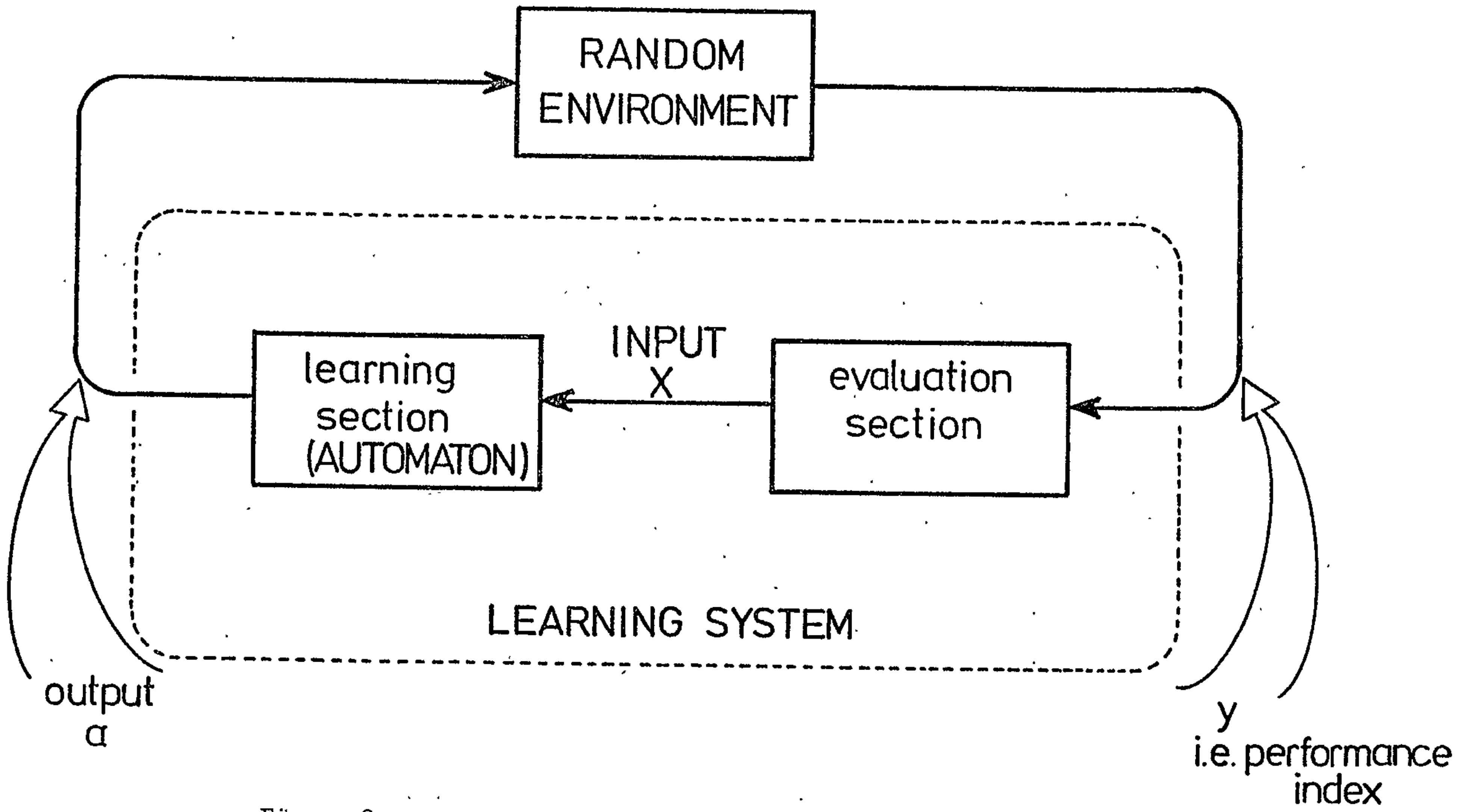


Figure 2 Interaction of SLA with Environment

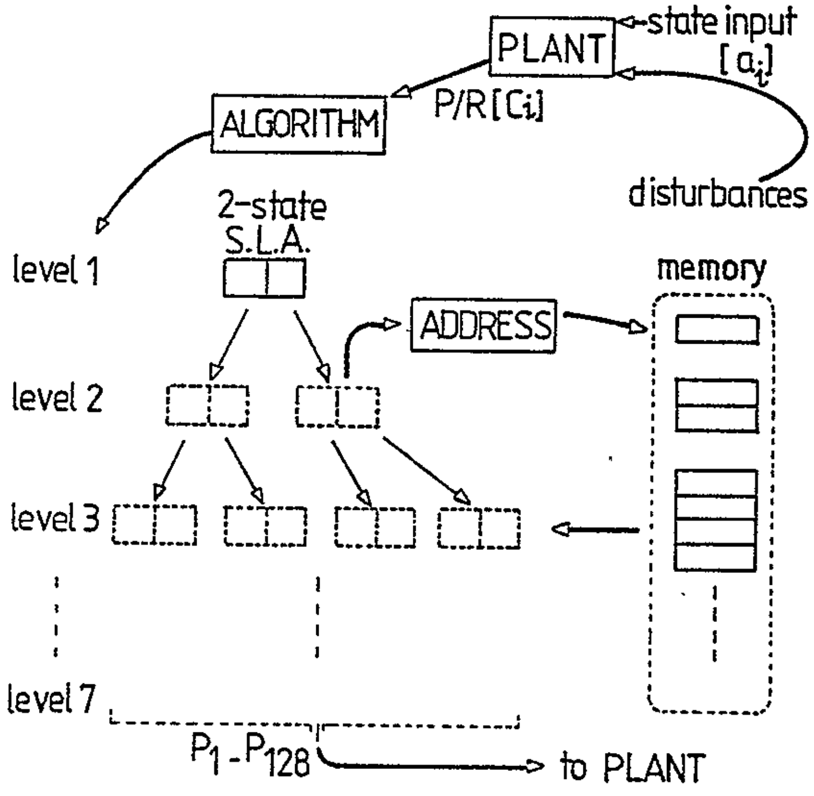


Figure 3 Hierarchical Structure SLA

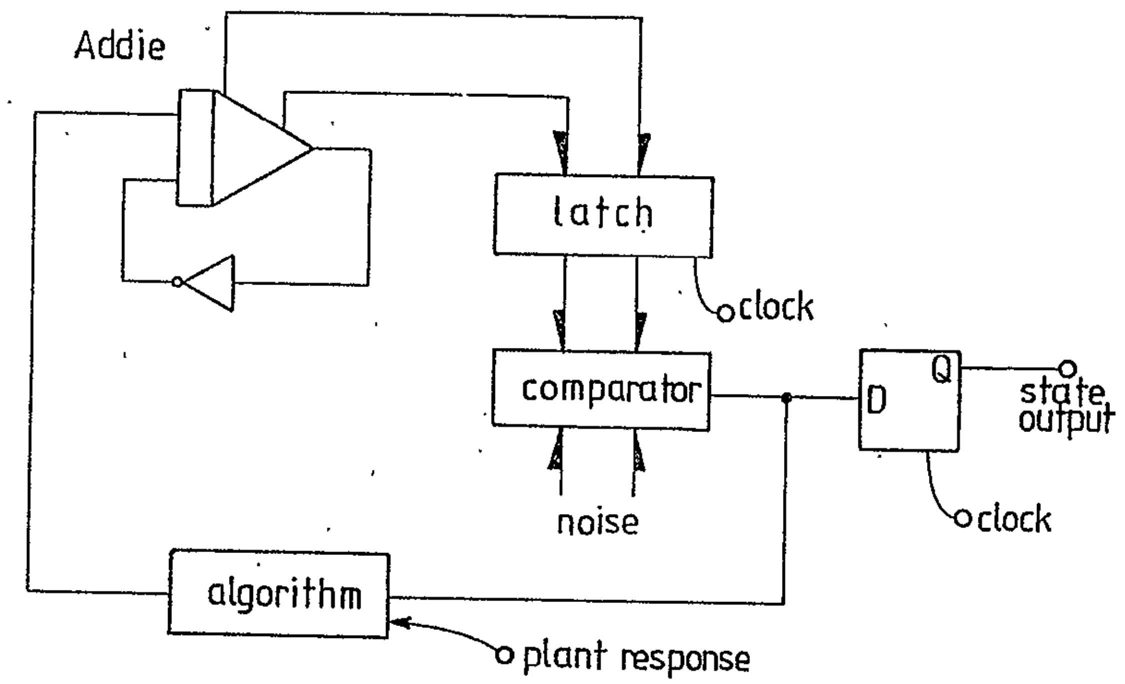


Figure 4 2 - State ADDIE SLA

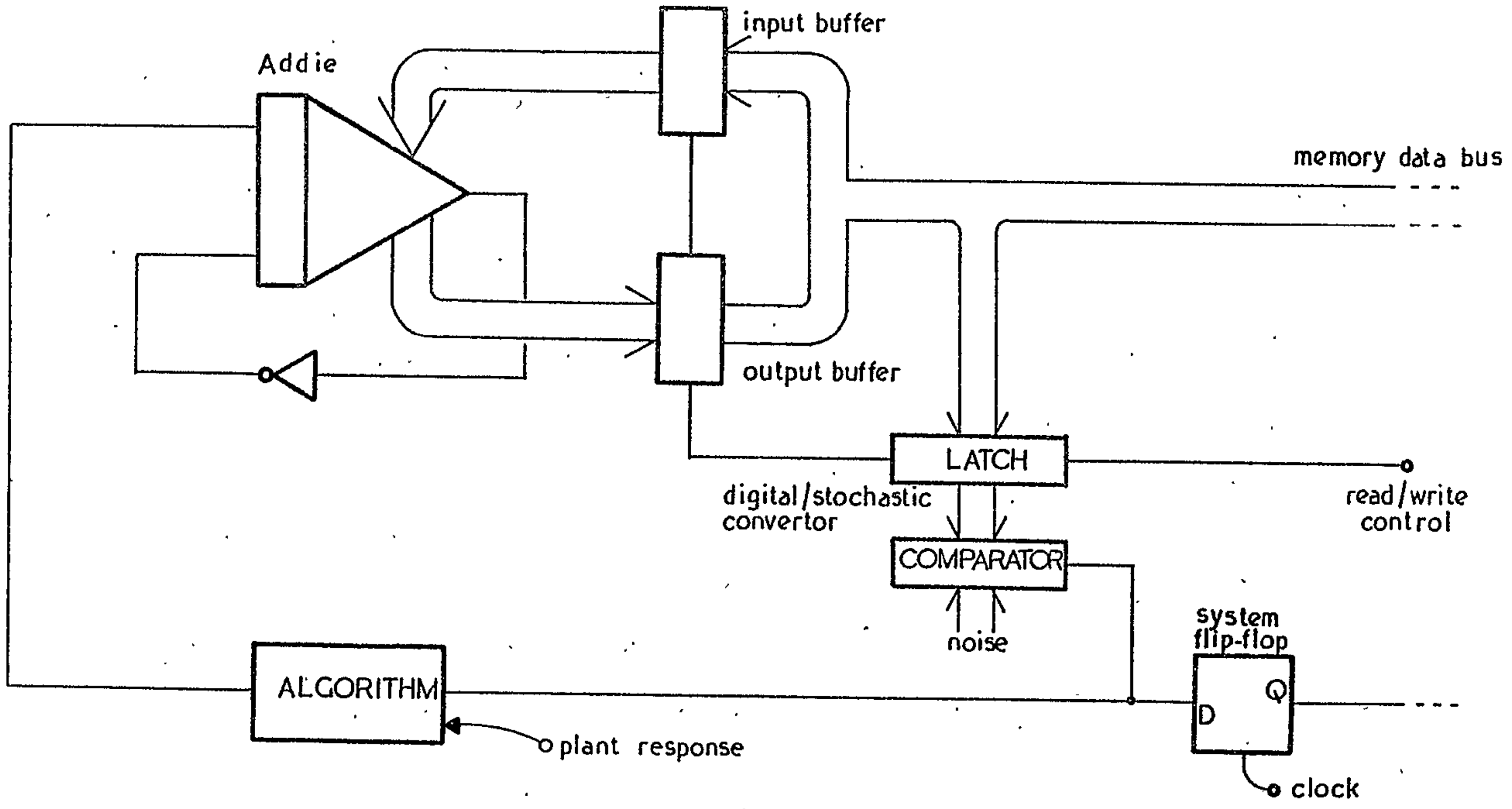


Figure 5 ADDIE SLA in Hierarchical System

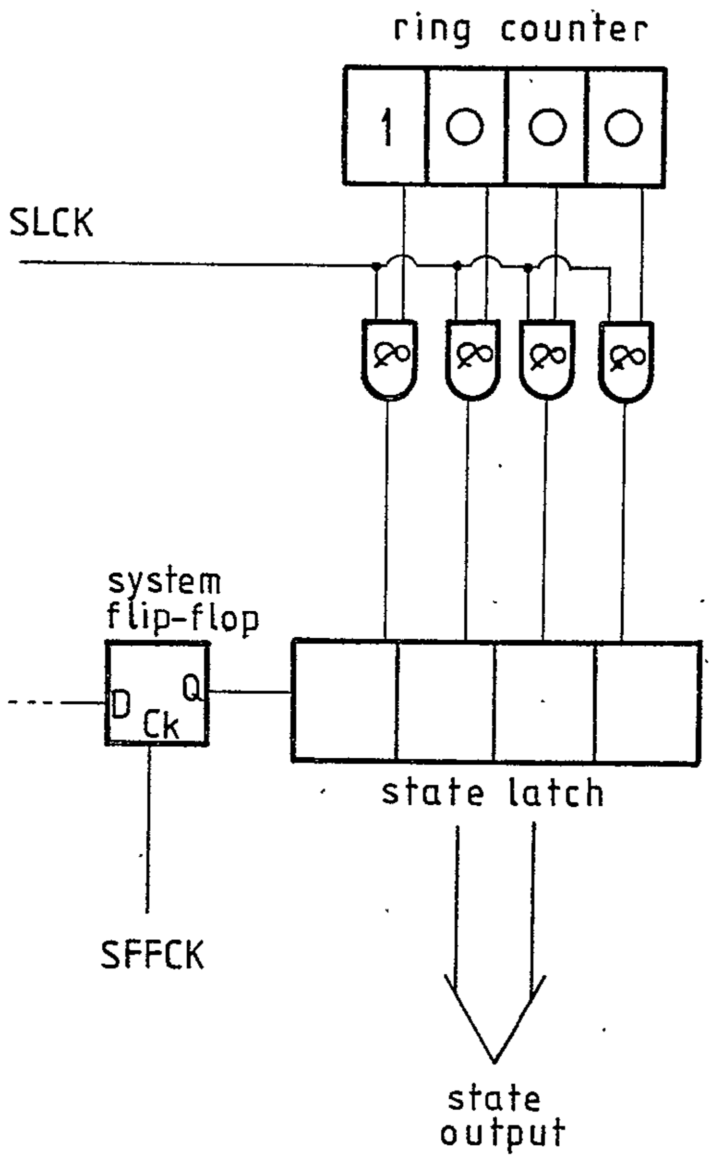


Figure 6 State Latch Loading Circuit

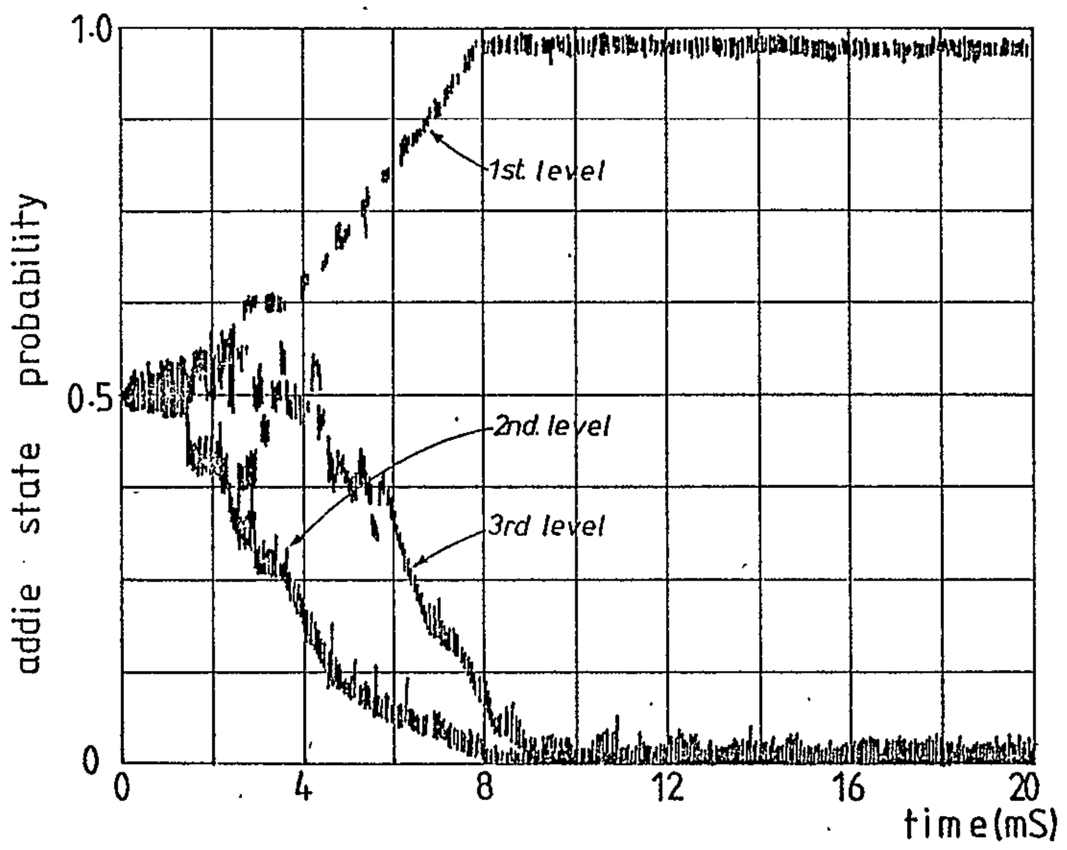


Figure 7 Learning Characteristic of an 8 - State System

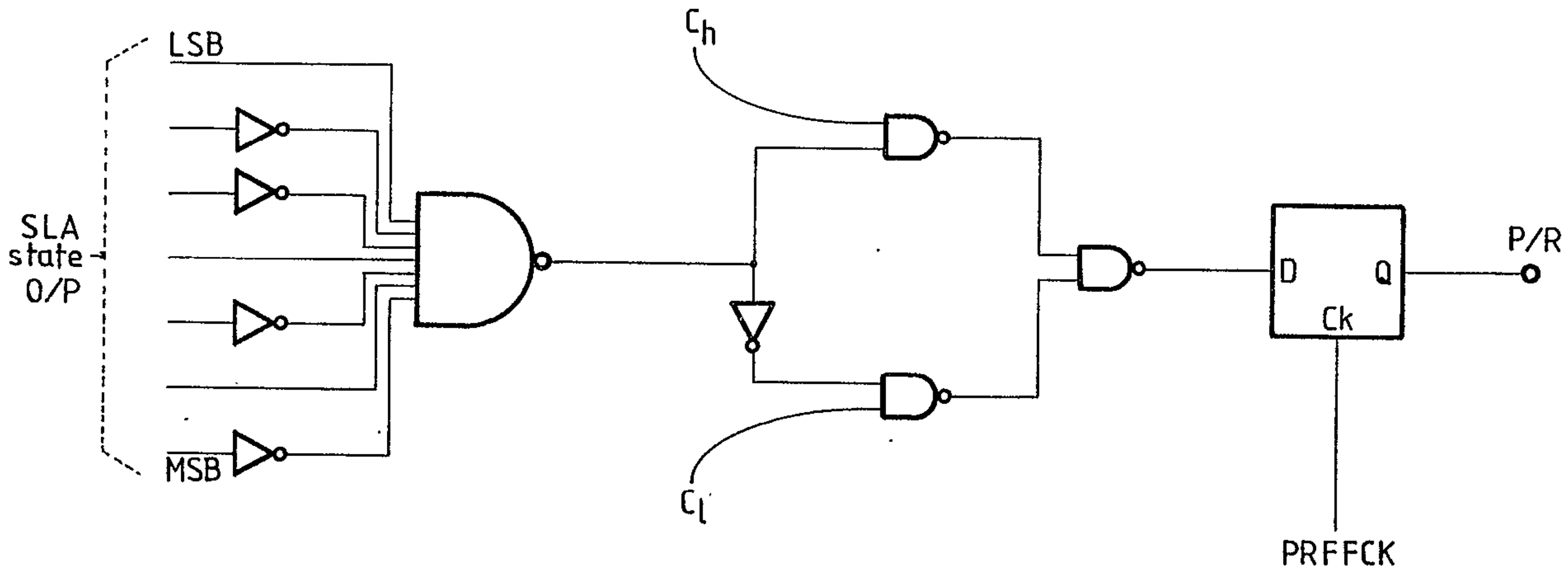


Figure 8 Simple Plant Simulator (State 41)

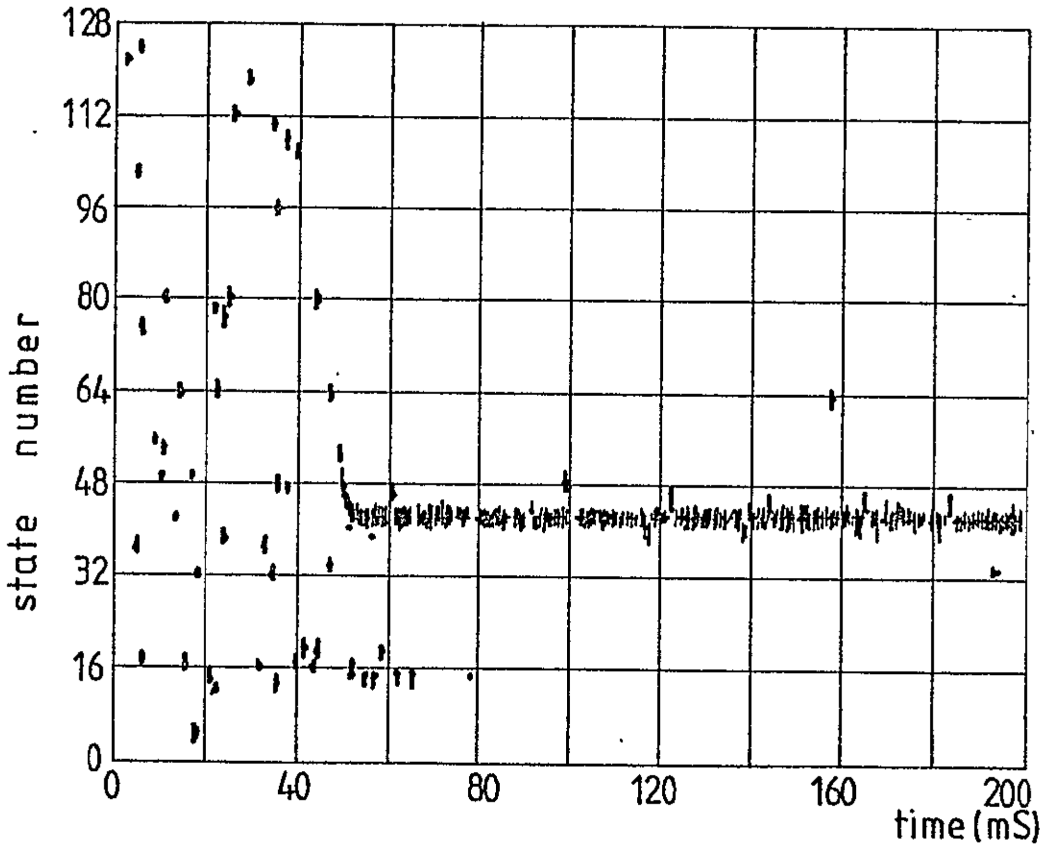


Figure 9 Output Map from "Simple Plant"

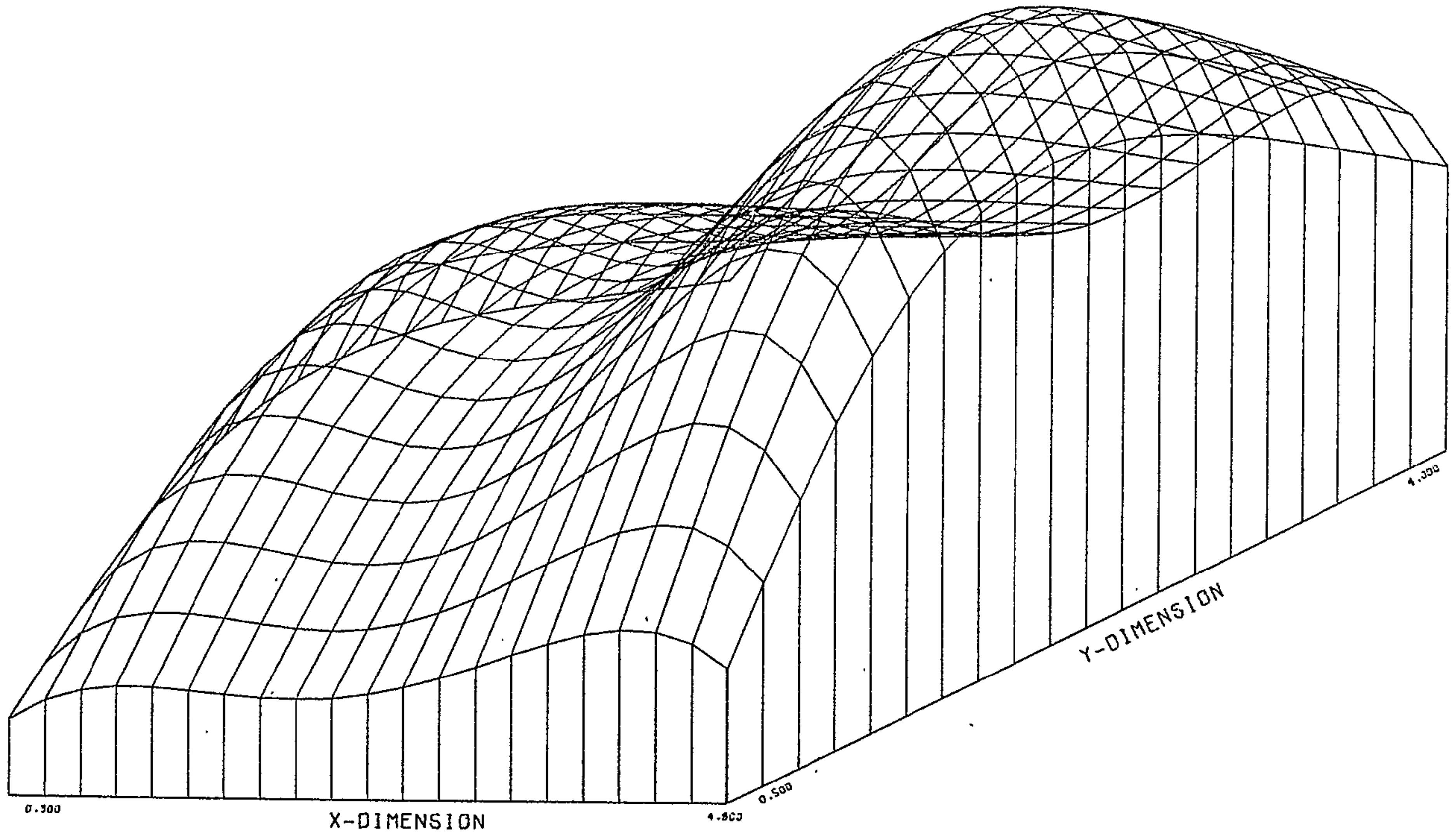


Figure 10 Three-Dimensional PI Surface

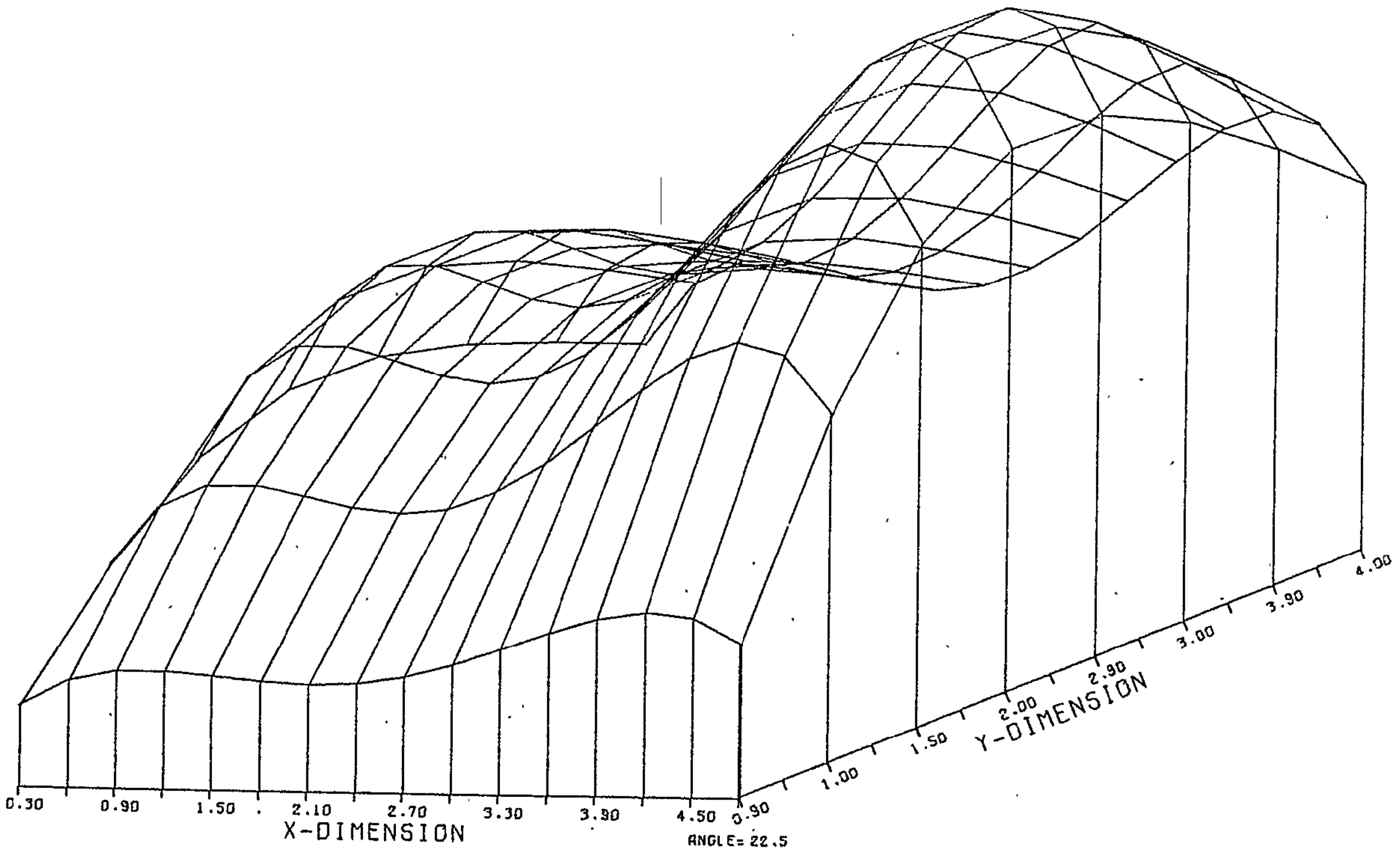


Figure. 11 PI Surface with 128 - Point Grid

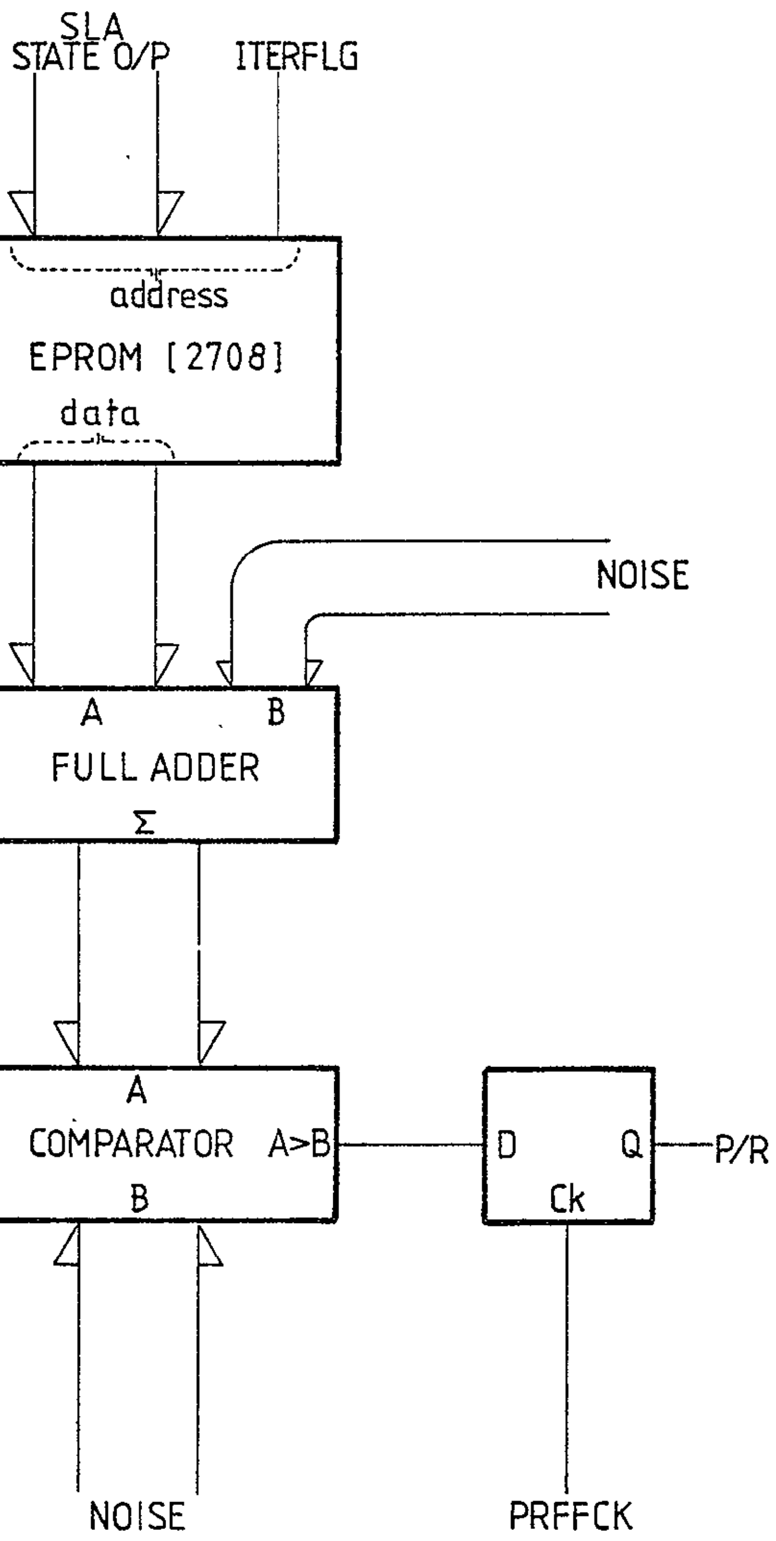


Figure 12 Plant Simulator using E P R O M

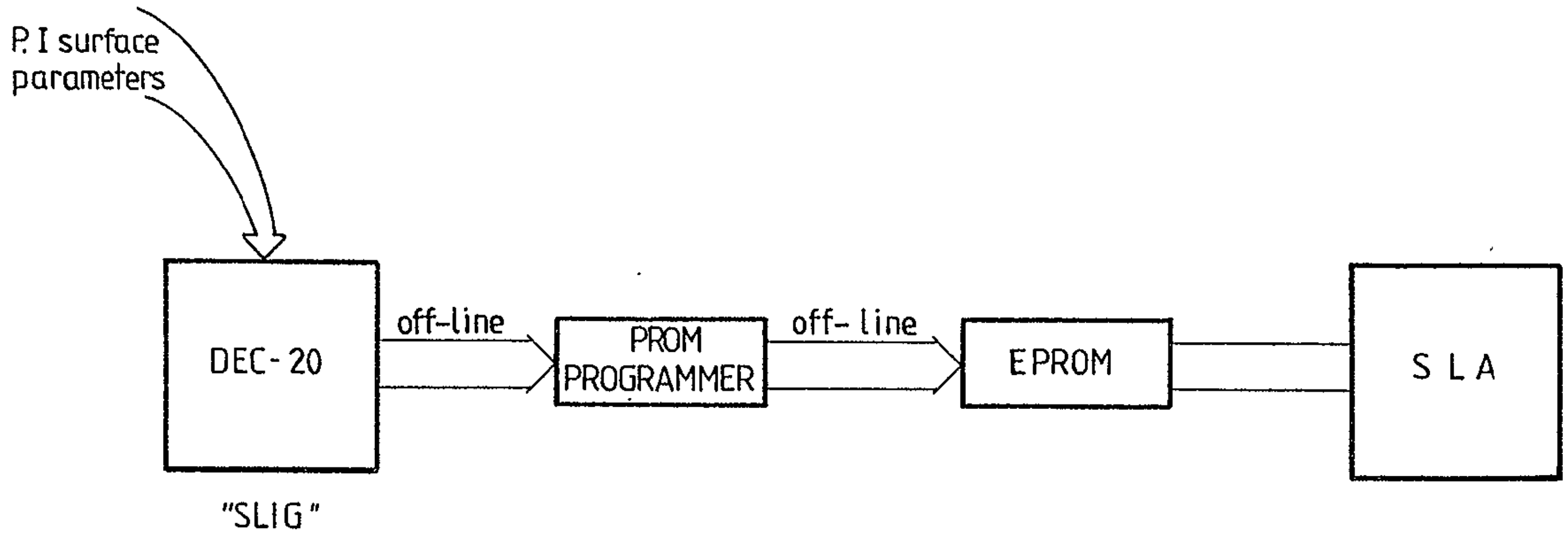


Figure 13 Data Preparation for SLA Experiments

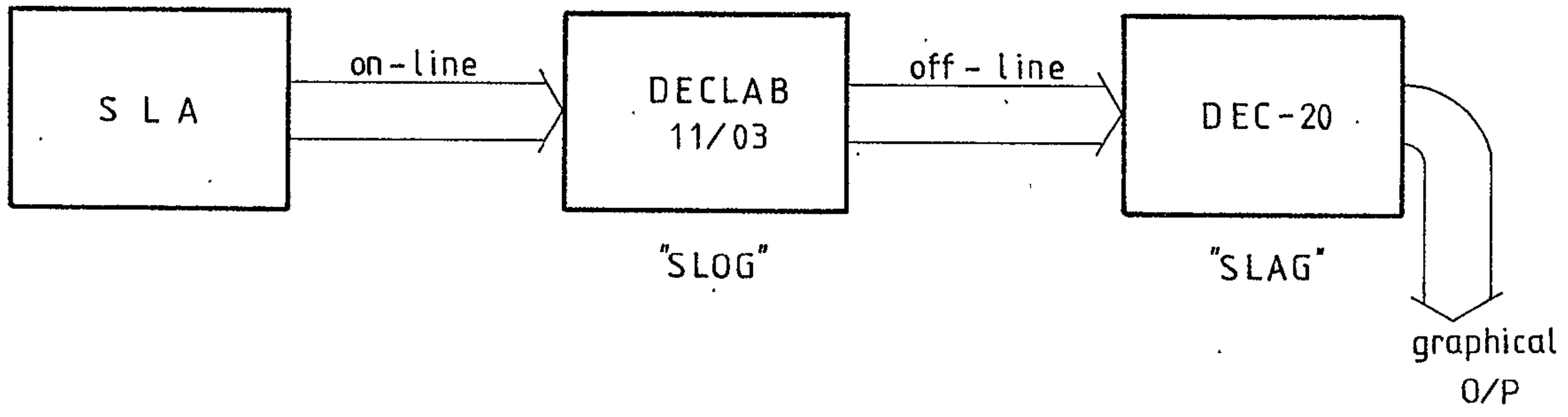


Figure 14 Logging and Presentation of SLA Data

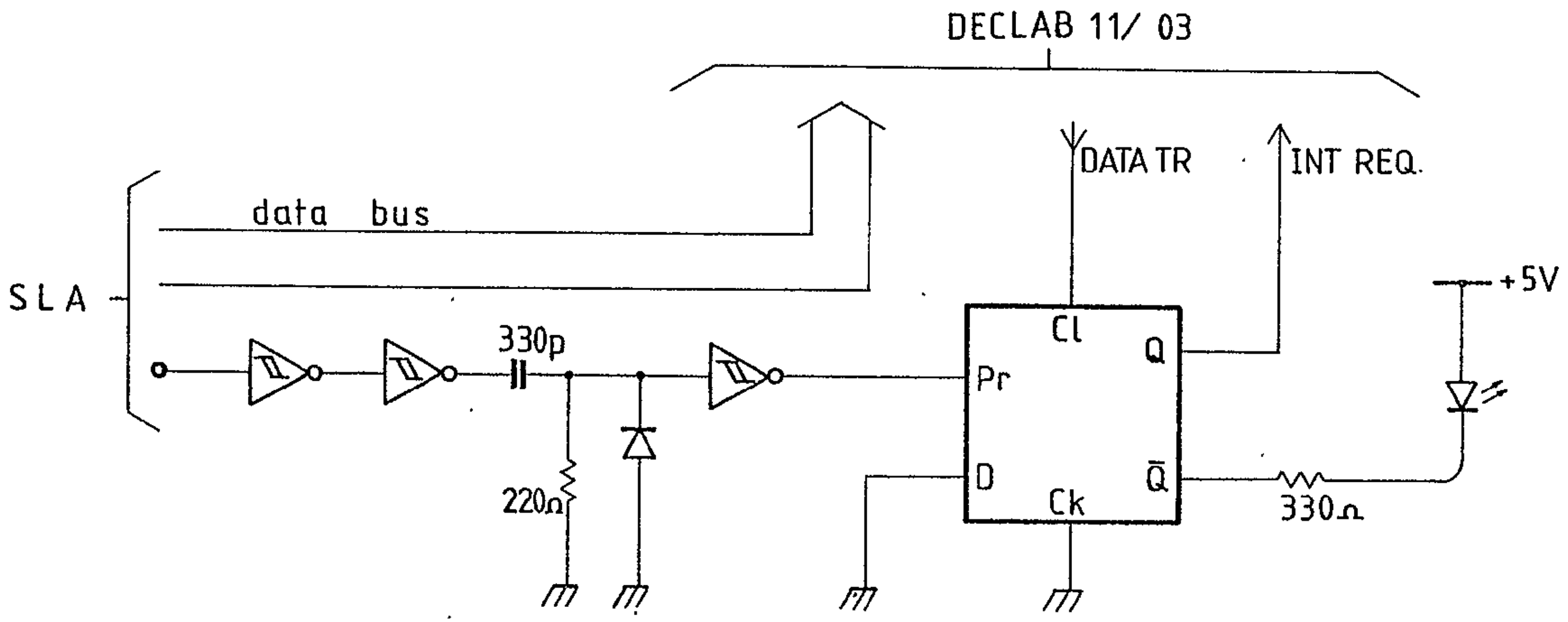


Figure 15 Declab 11/03 Interrupt Circuit

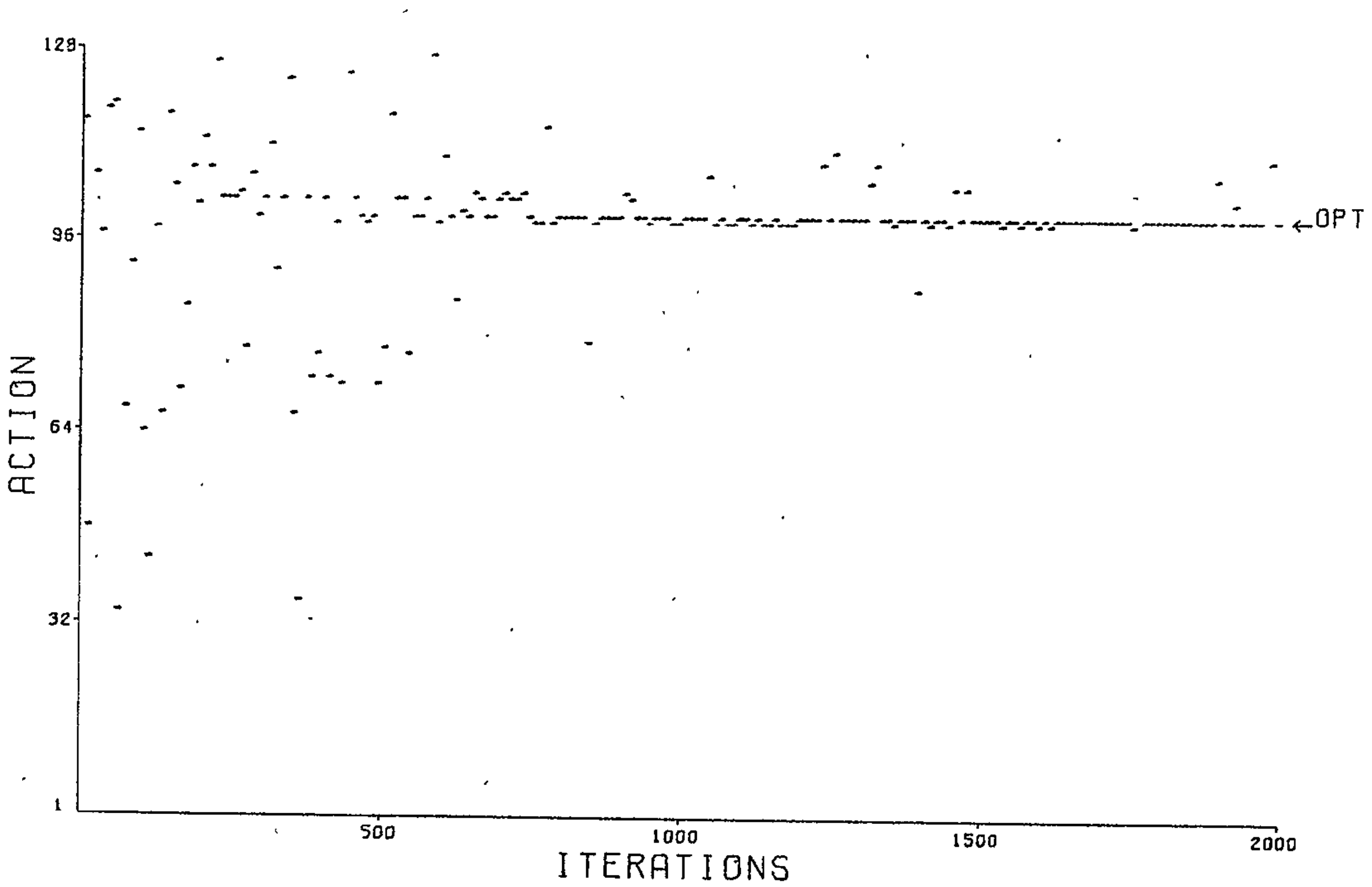


Figure 16 Output Map (1)

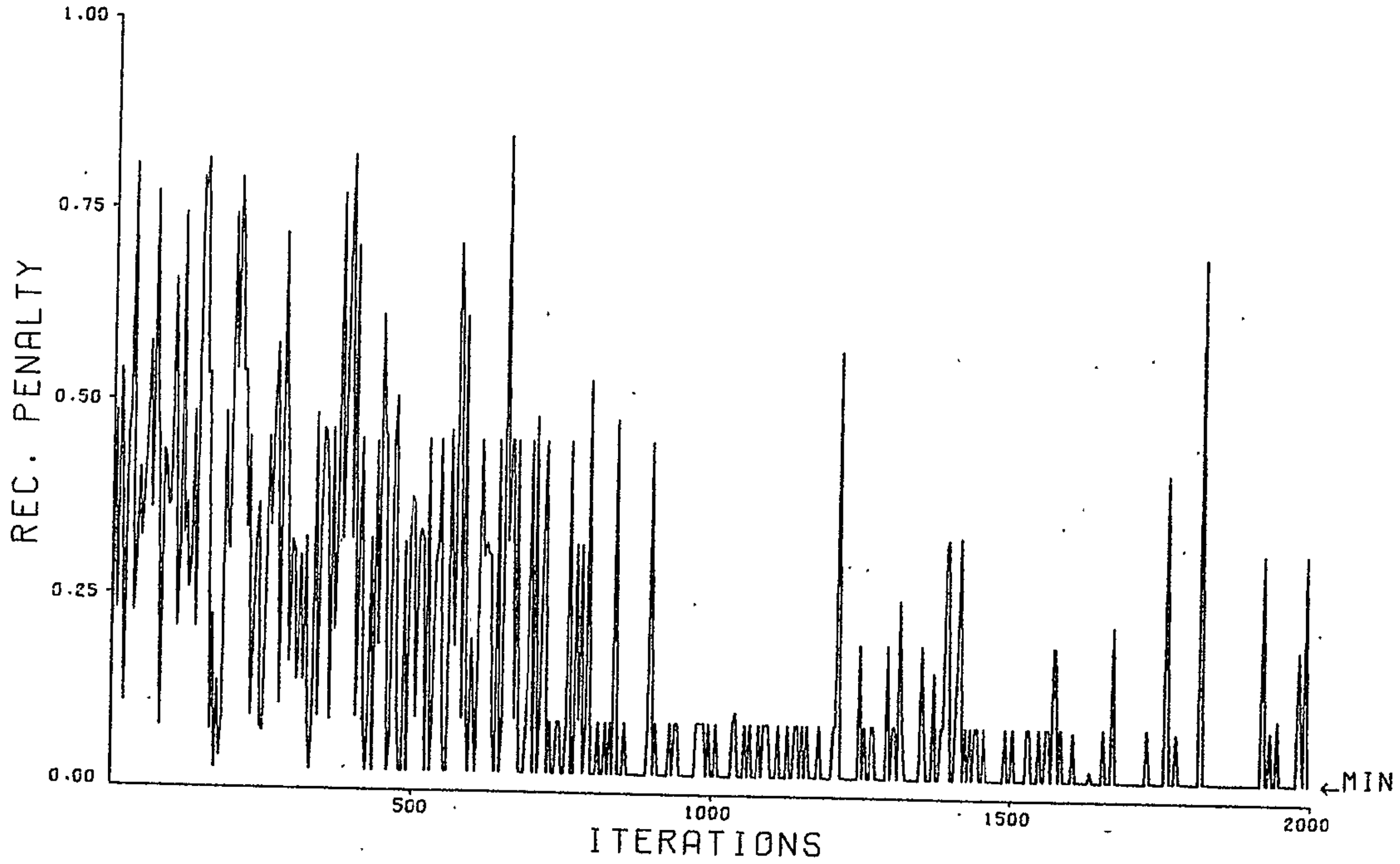


Figure 17 **Penalty Curve** (1)

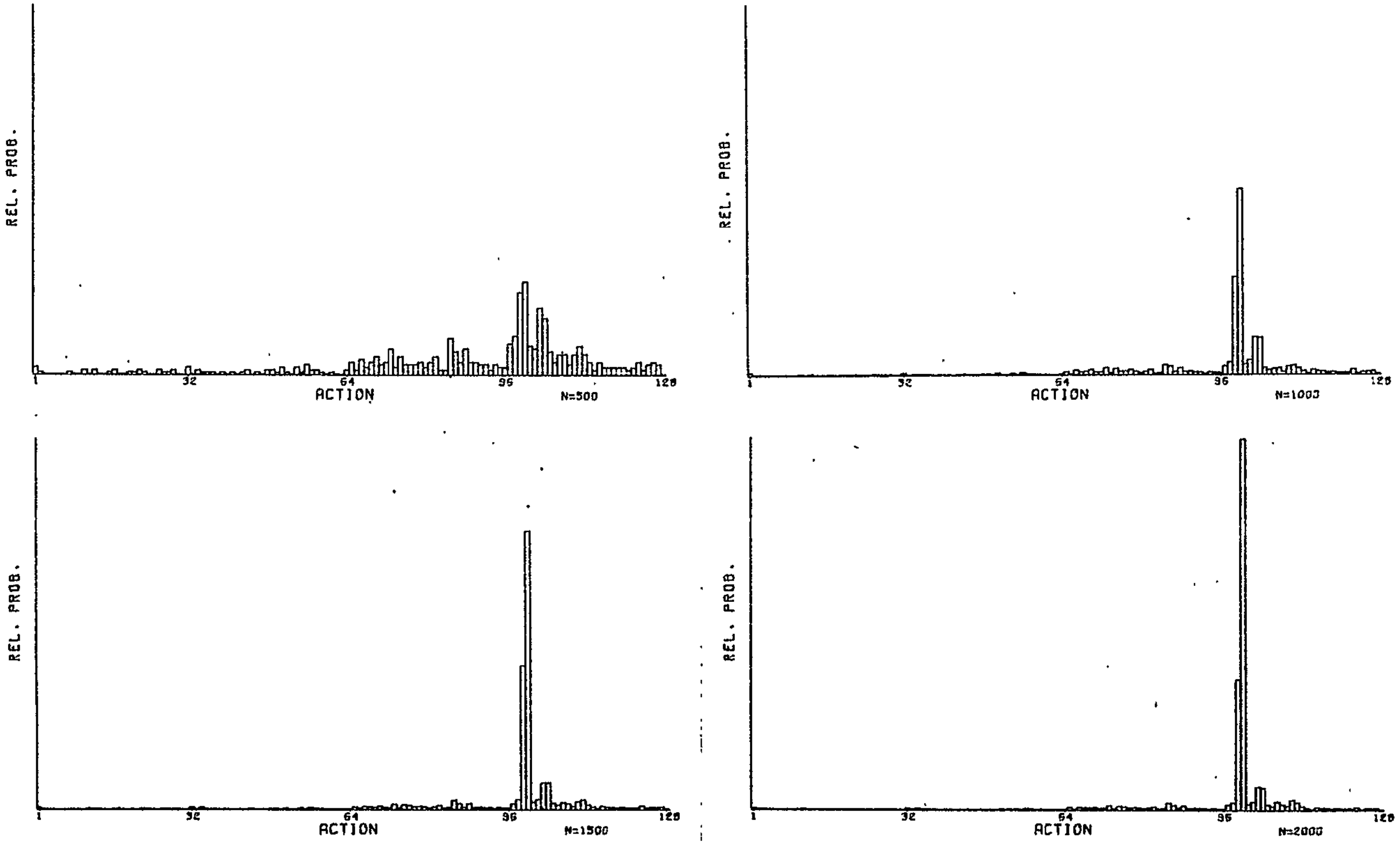


Figure 18 Distribution Curves (1)

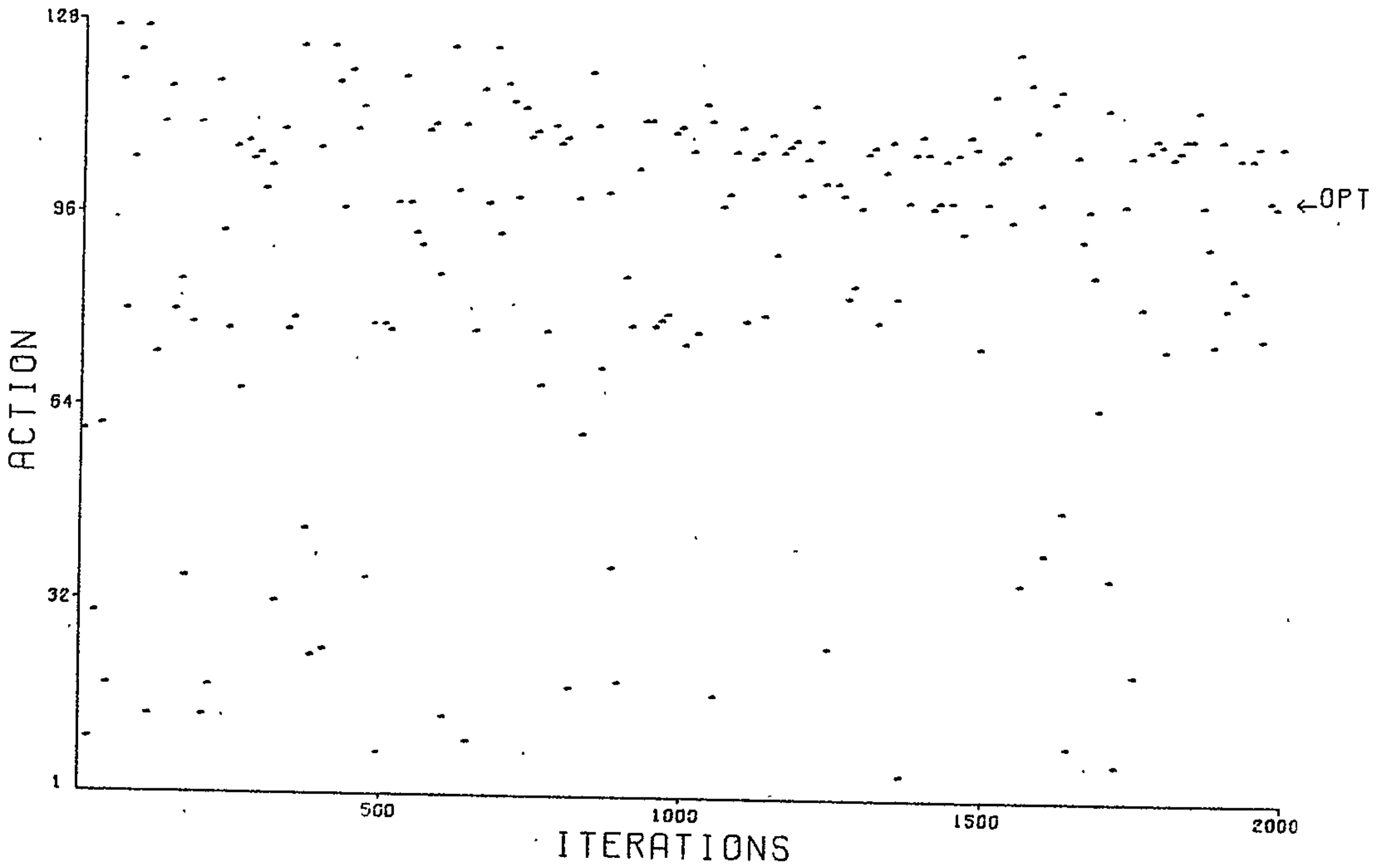


Figure 19 Output Map (2)

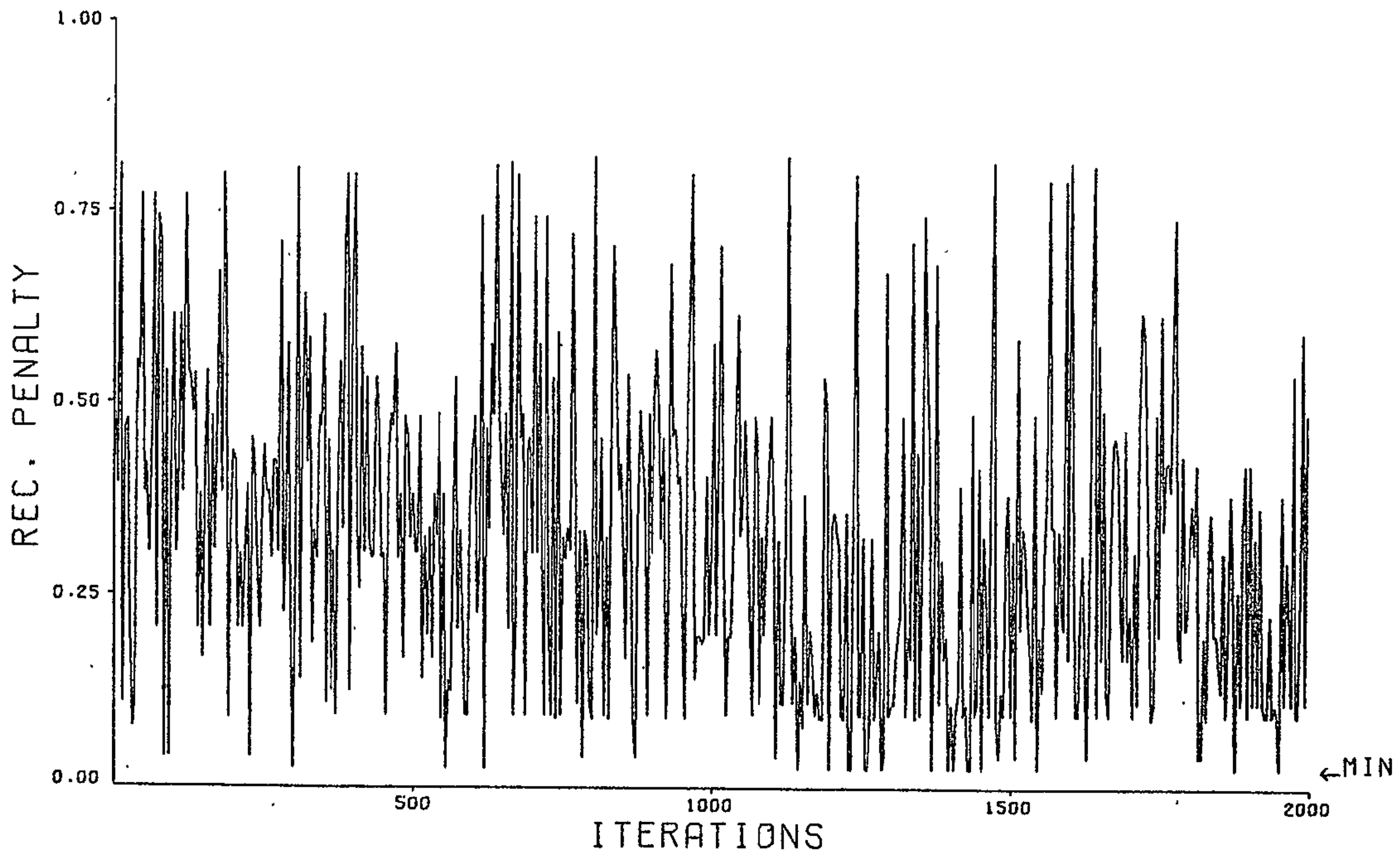


Figure 20 Penalty Curve (2)

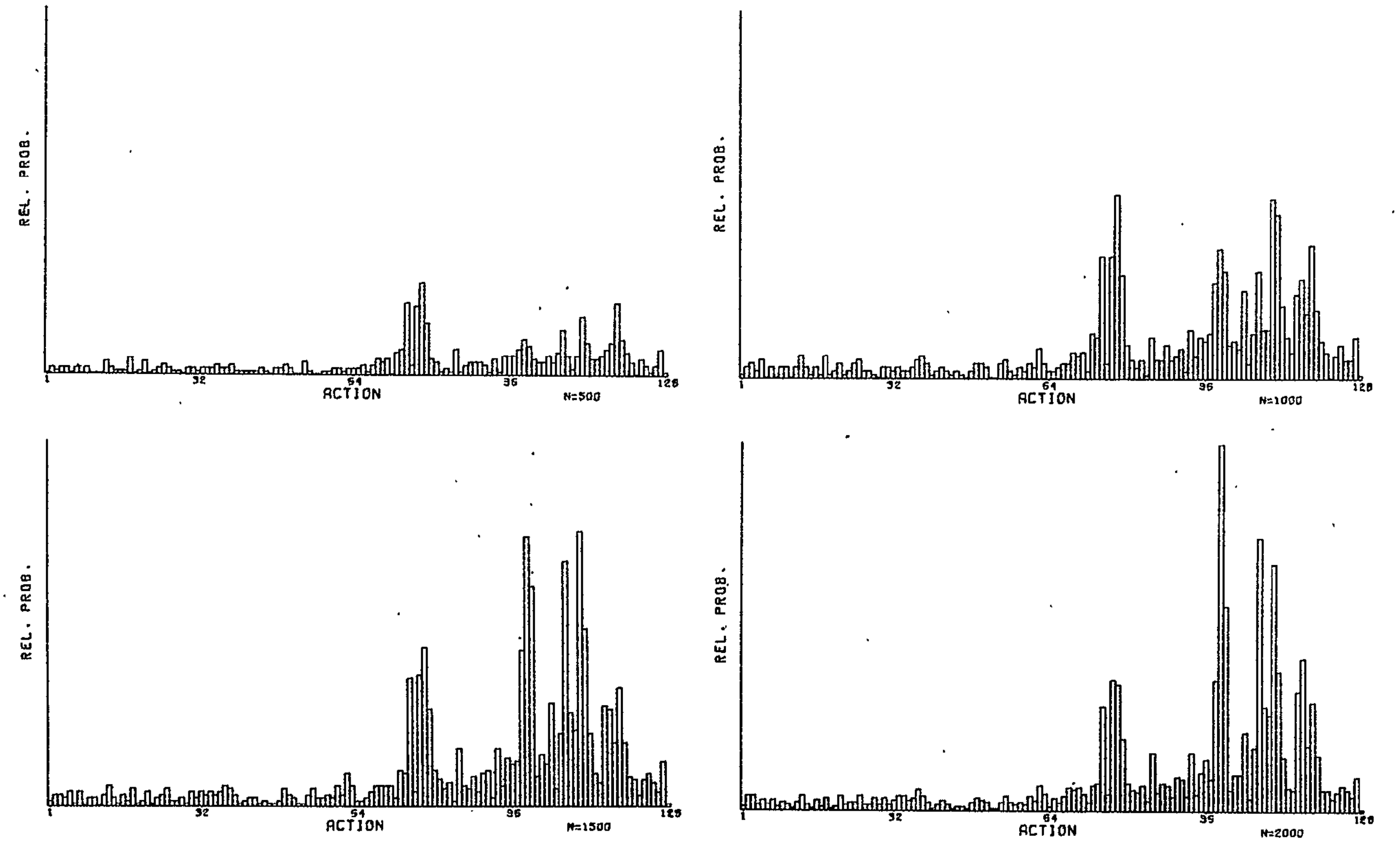


Figure 21 Distribution Curves (2)

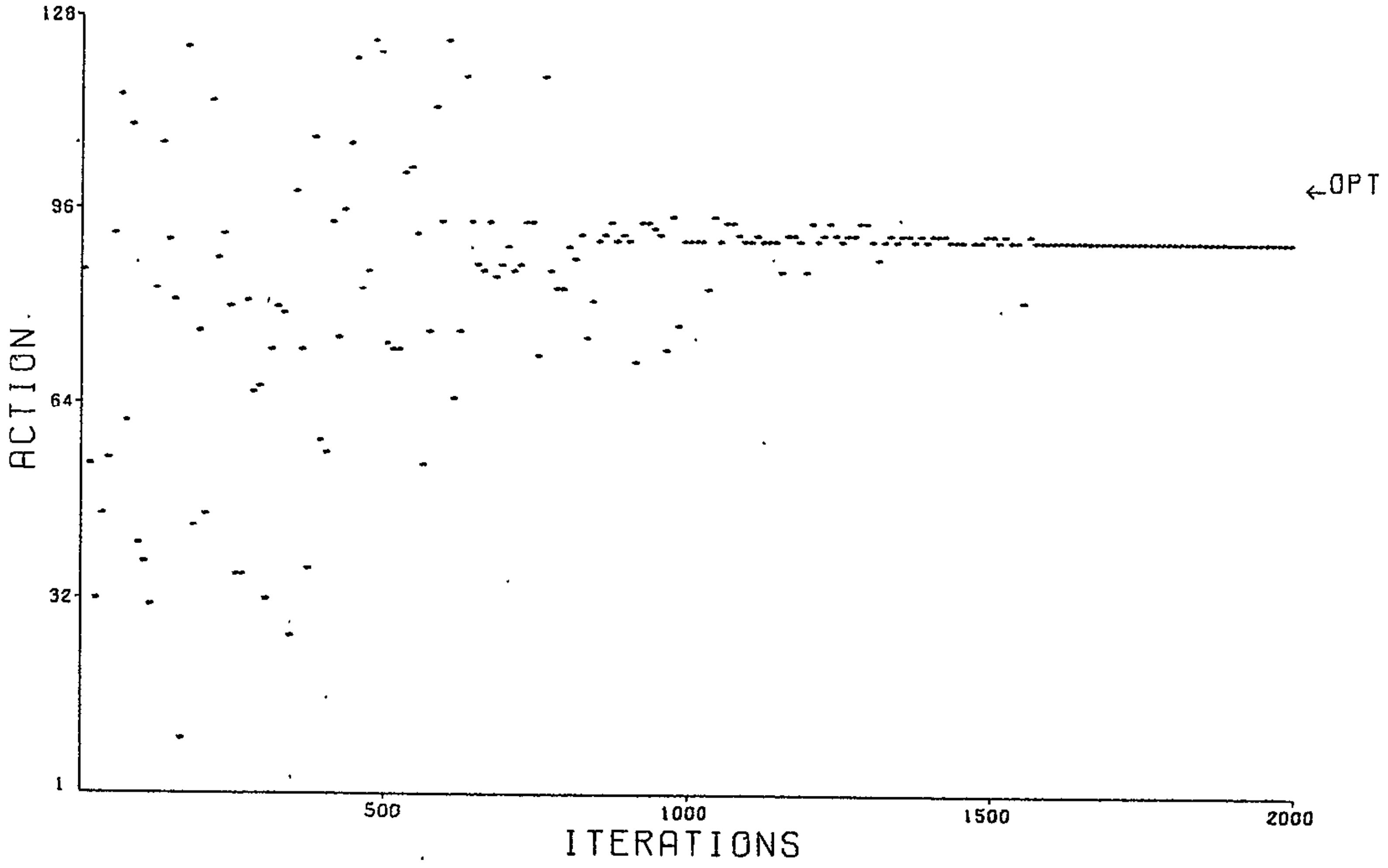


Figure 22 Output Map (3)

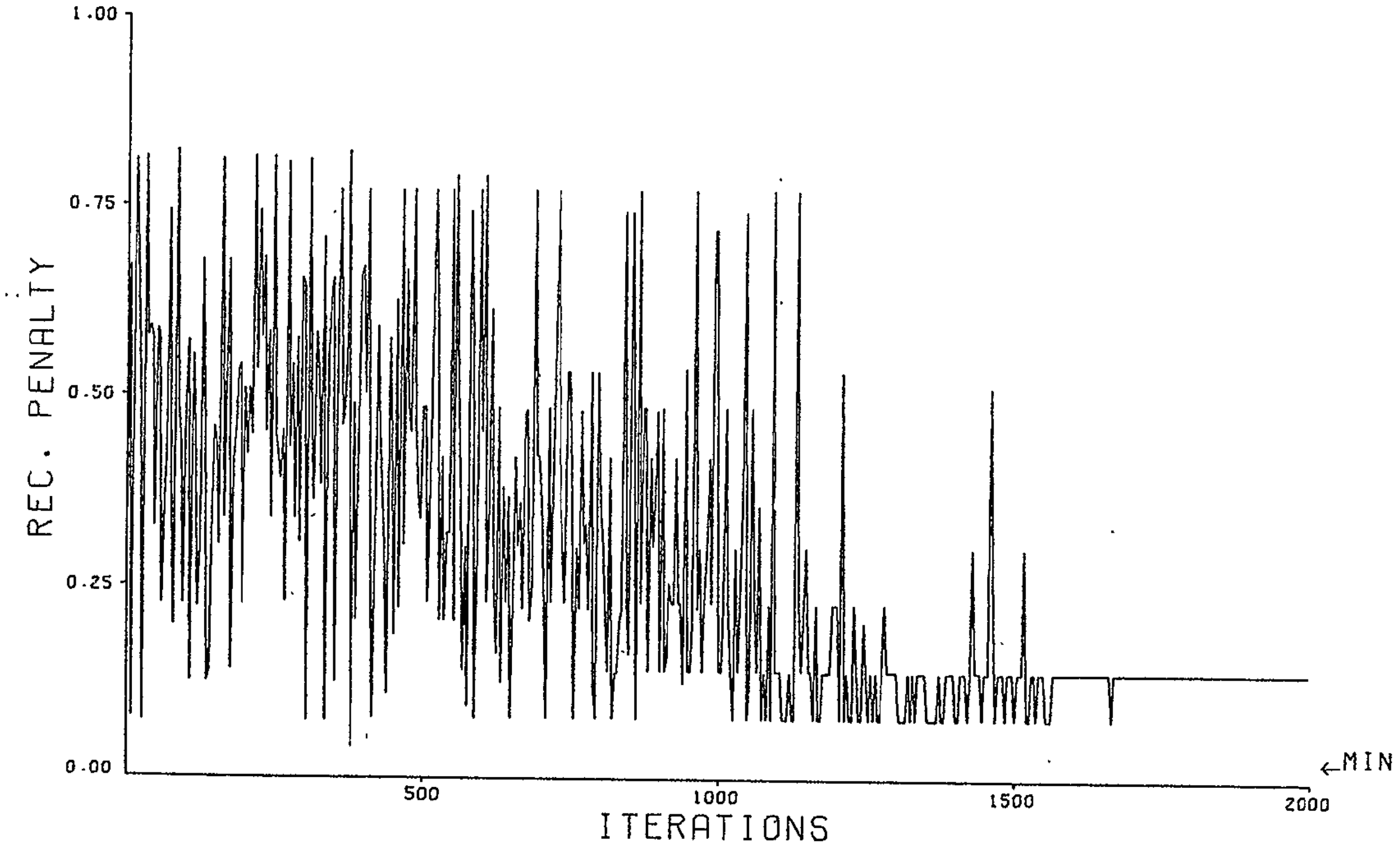


Figure 23 Penalty Curve (3)

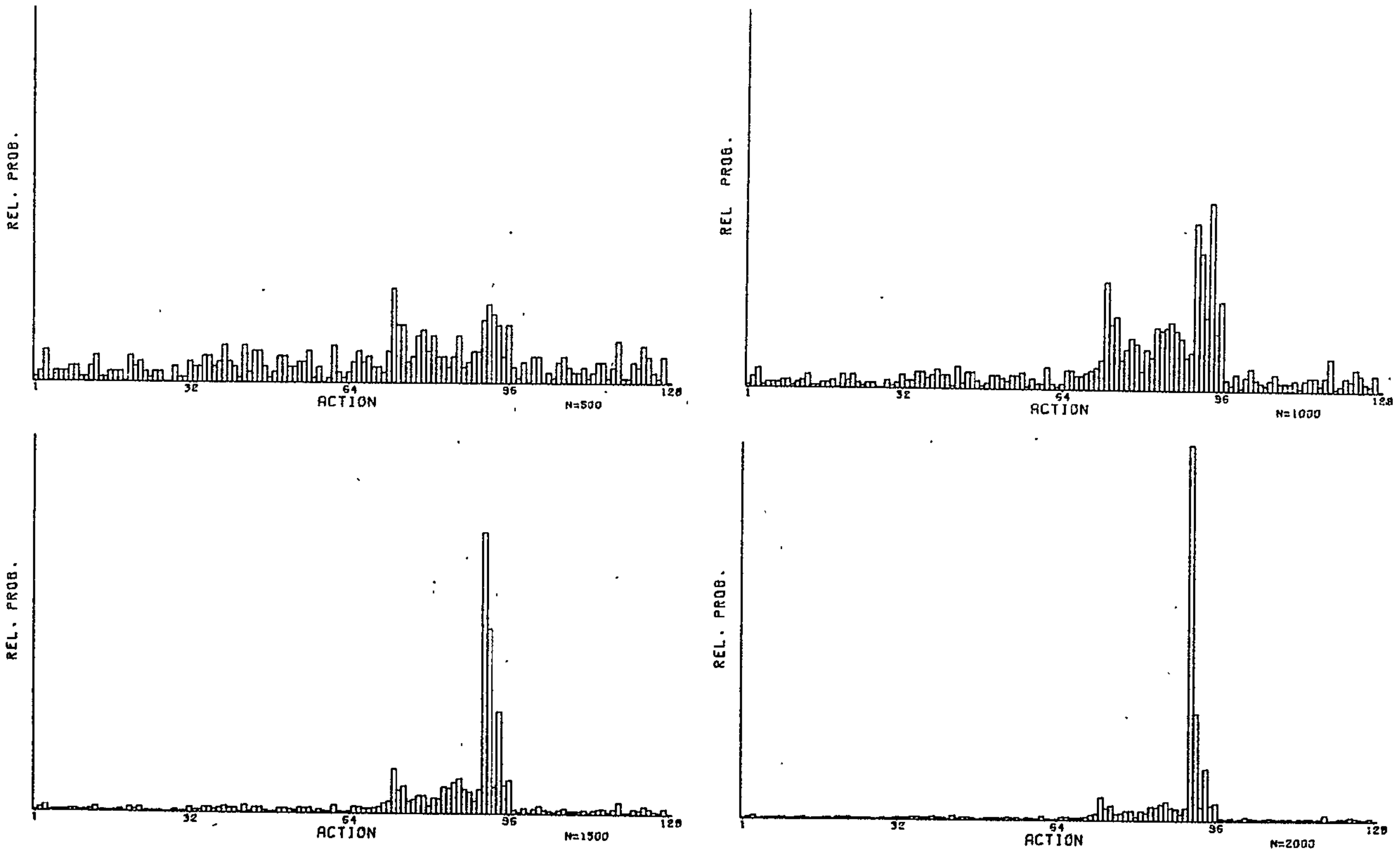


Figure 24 Distribution Curves (3)

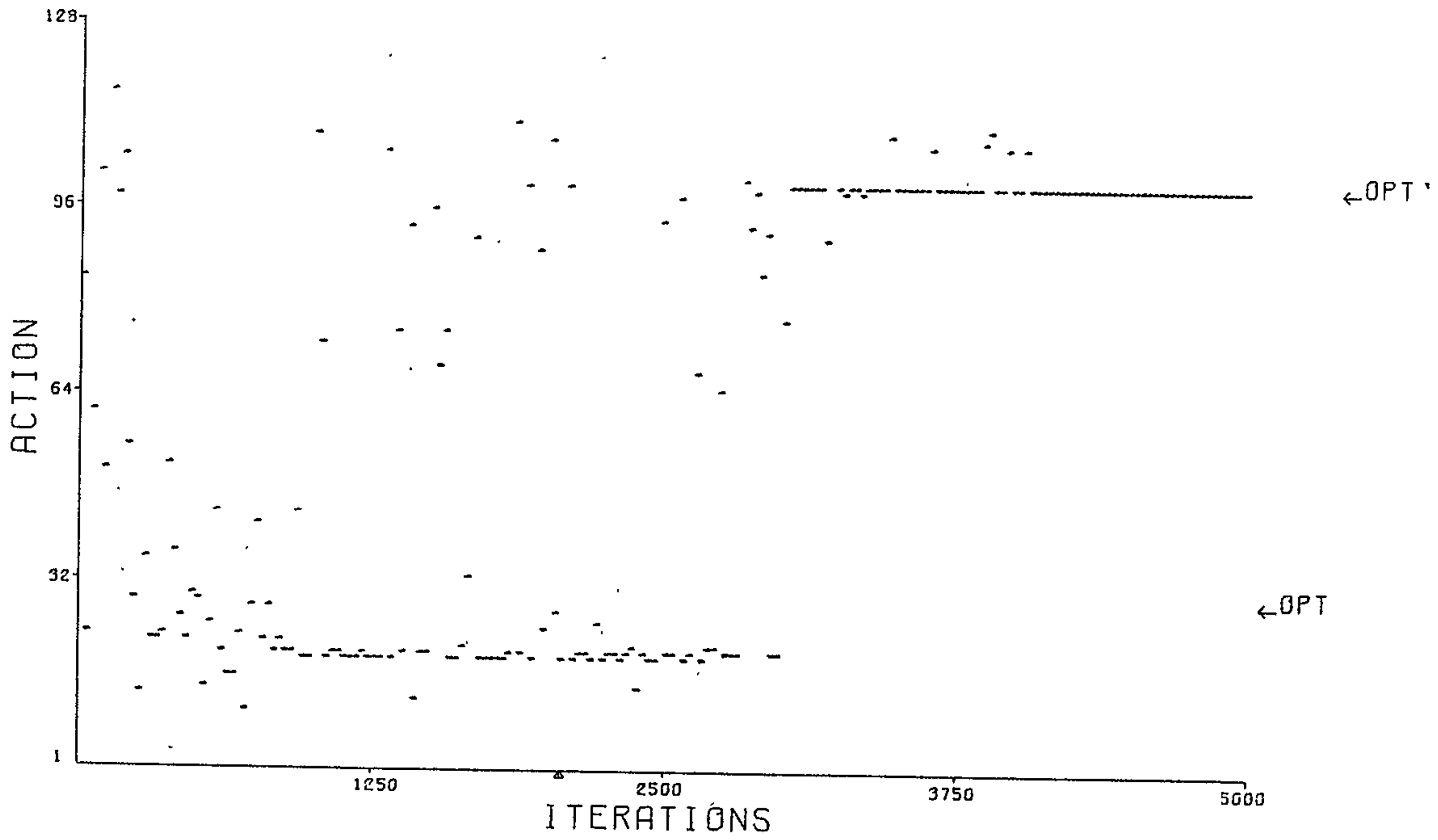


Figure 25 Output Map (4)

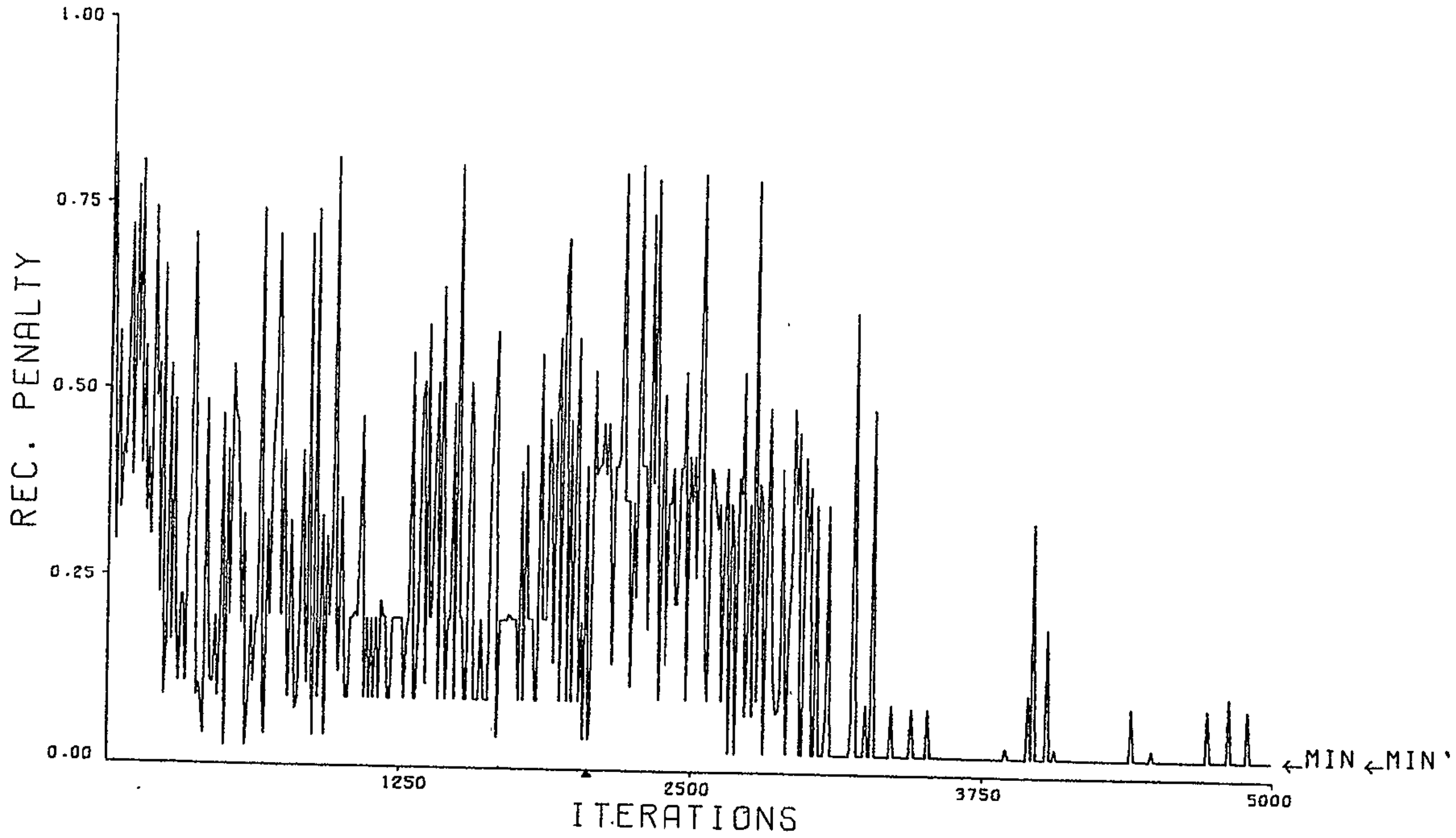


Figure 26 Penalty Curve (4)

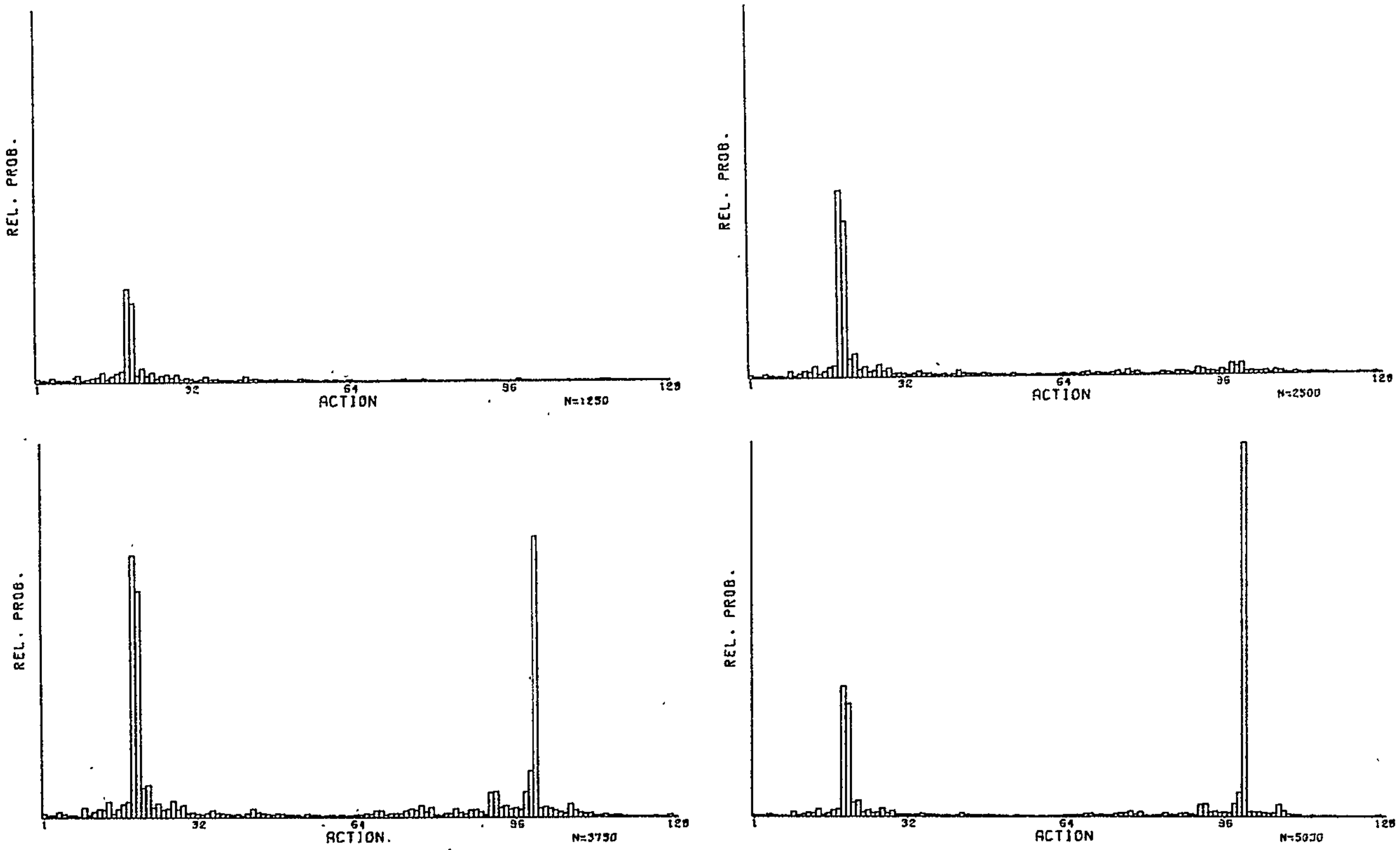
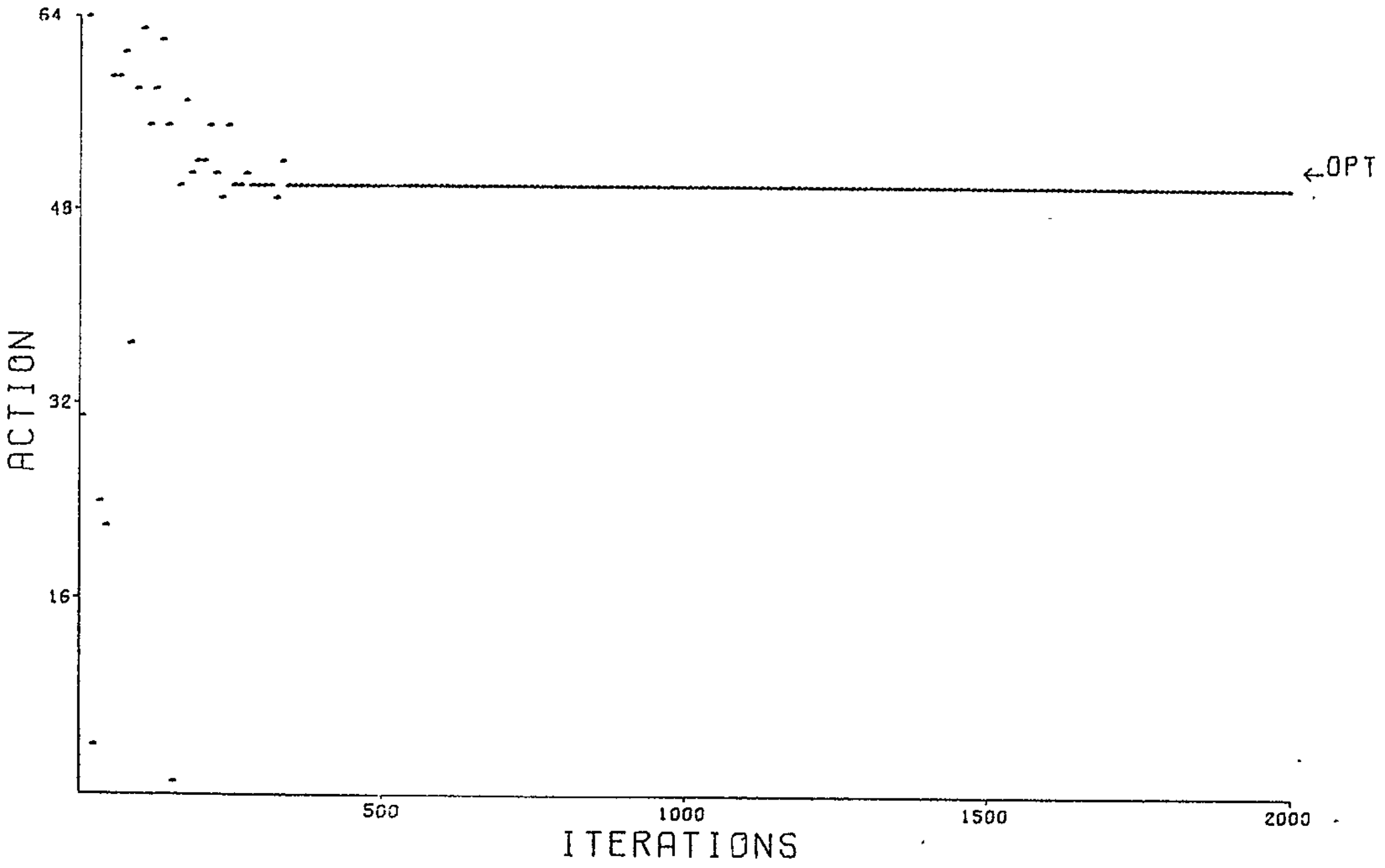


Figure 27 Distribution Curves (4)



<u>Figure 28</u> <u>Output Map</u> (5)

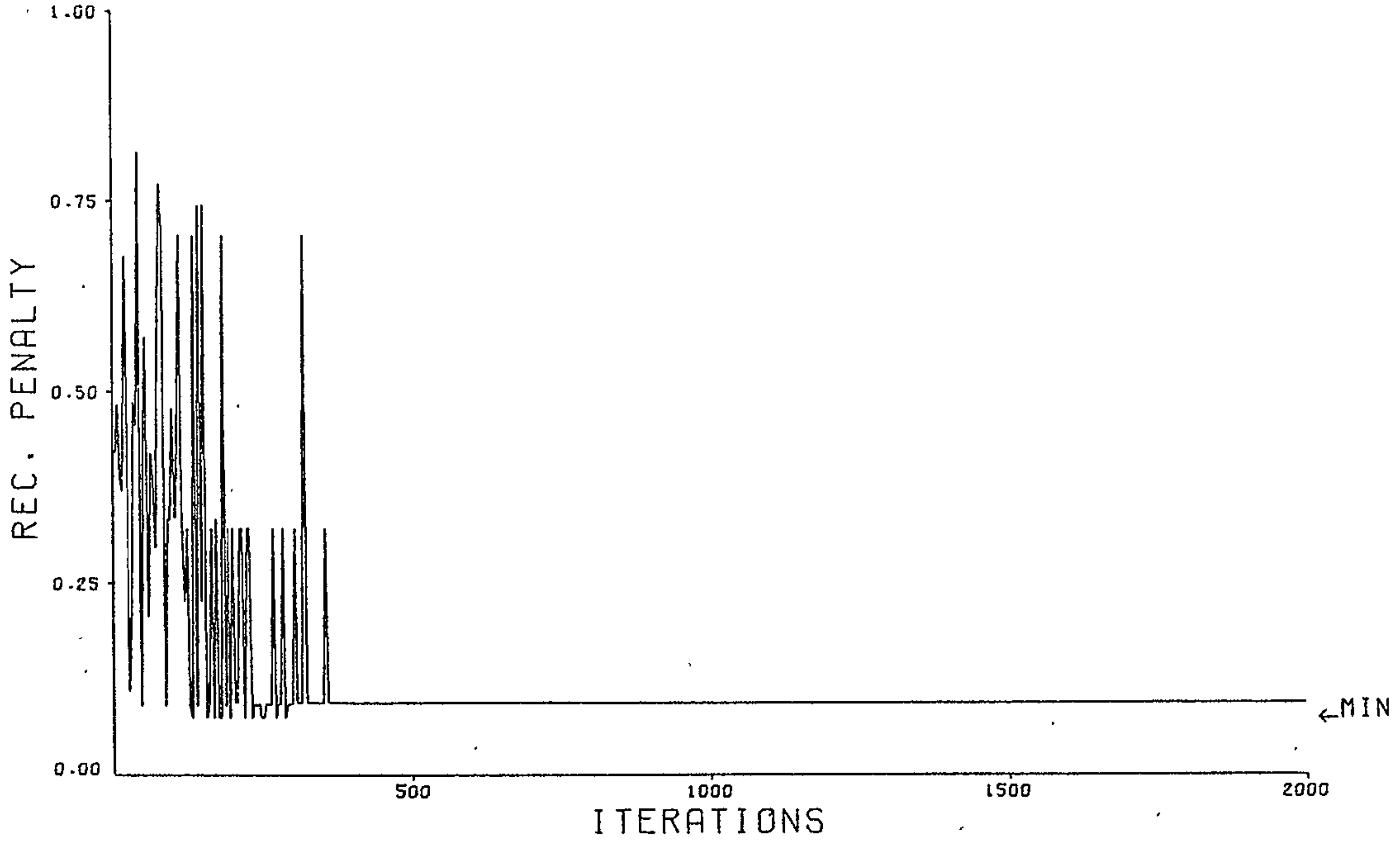


Figure 29 Penalty Curve (5)

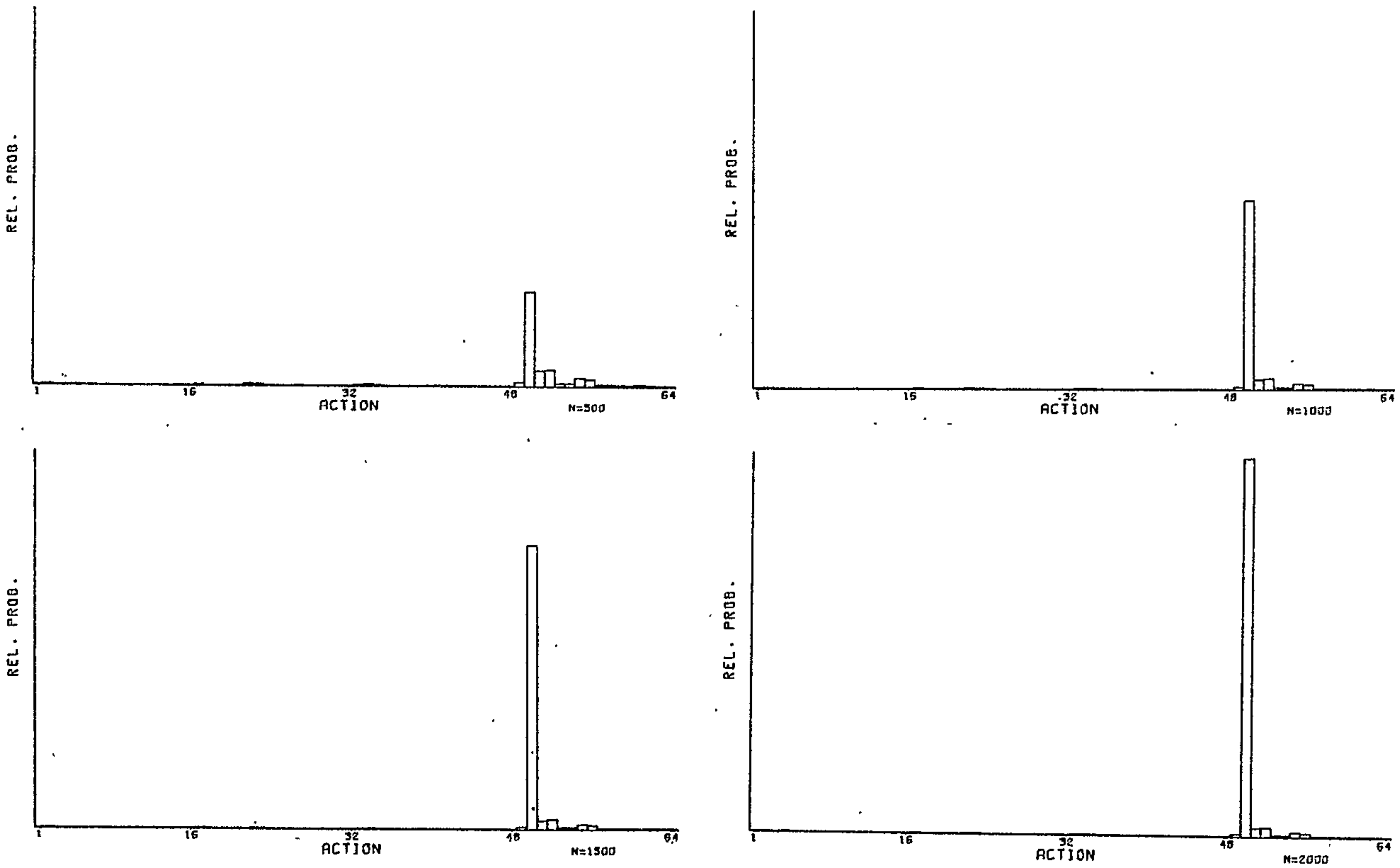


Figure 30 Distribution Curves (5)

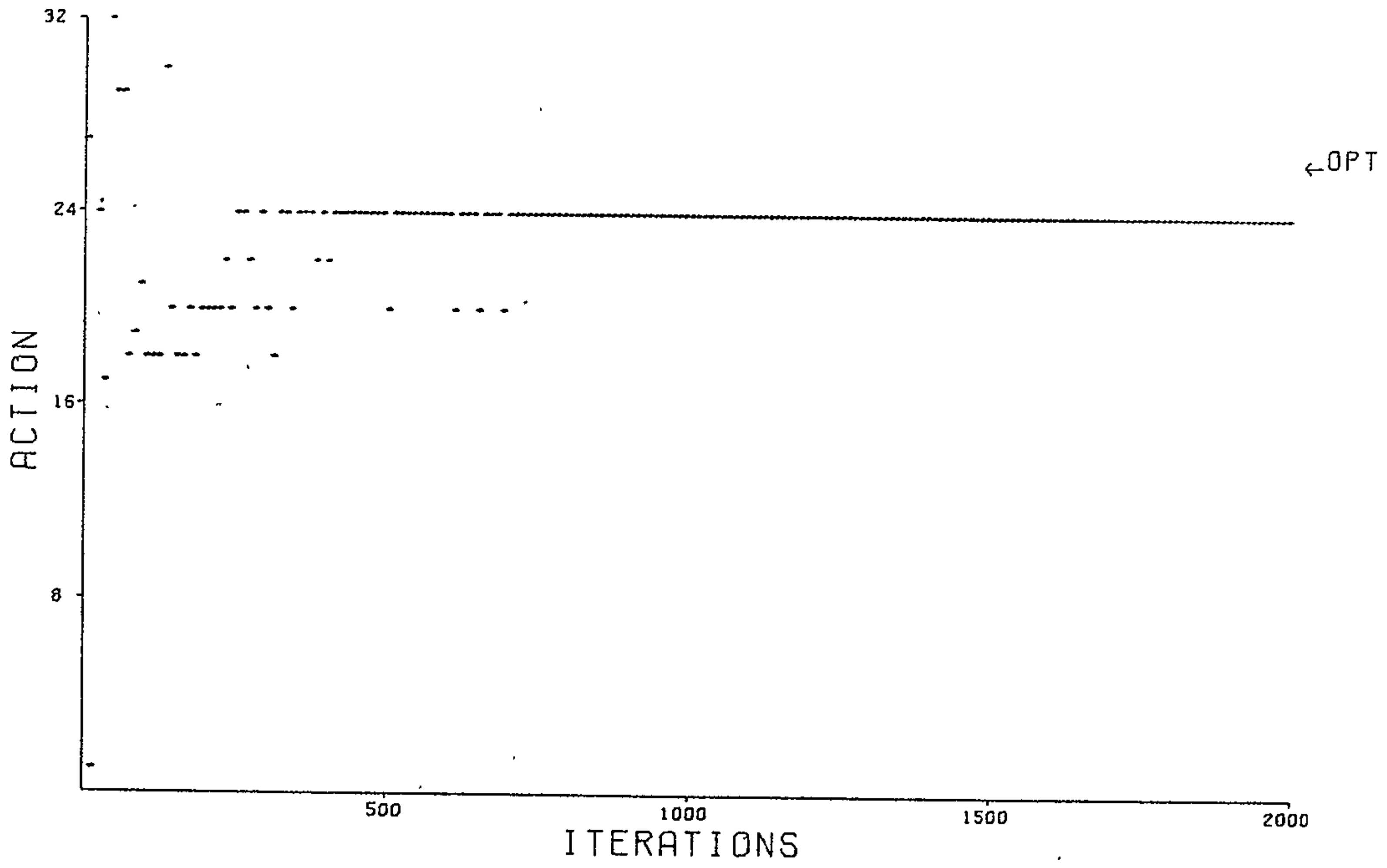


Figure 31 Output Map (6)

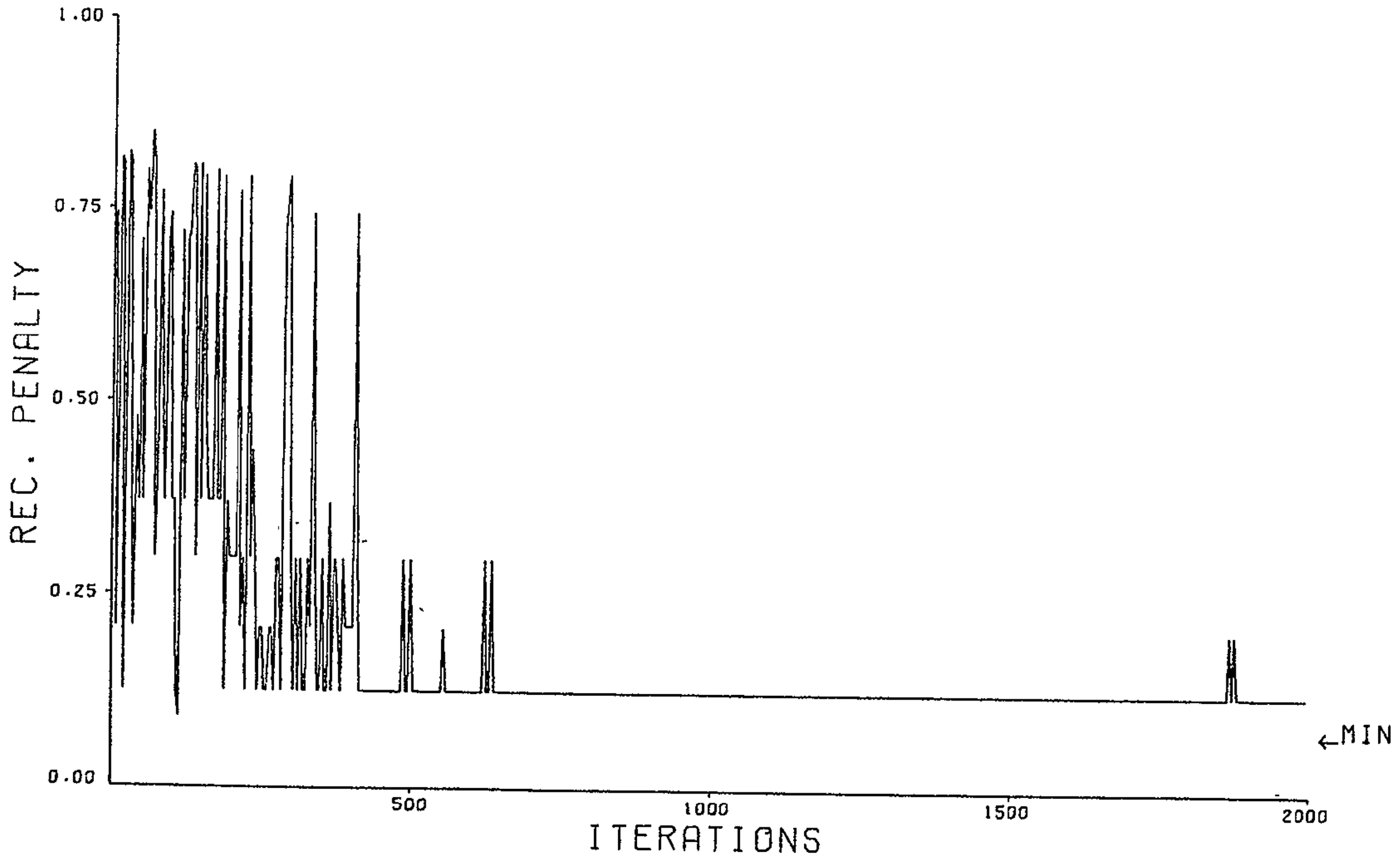


Figure 32 Penalty Curve (6)

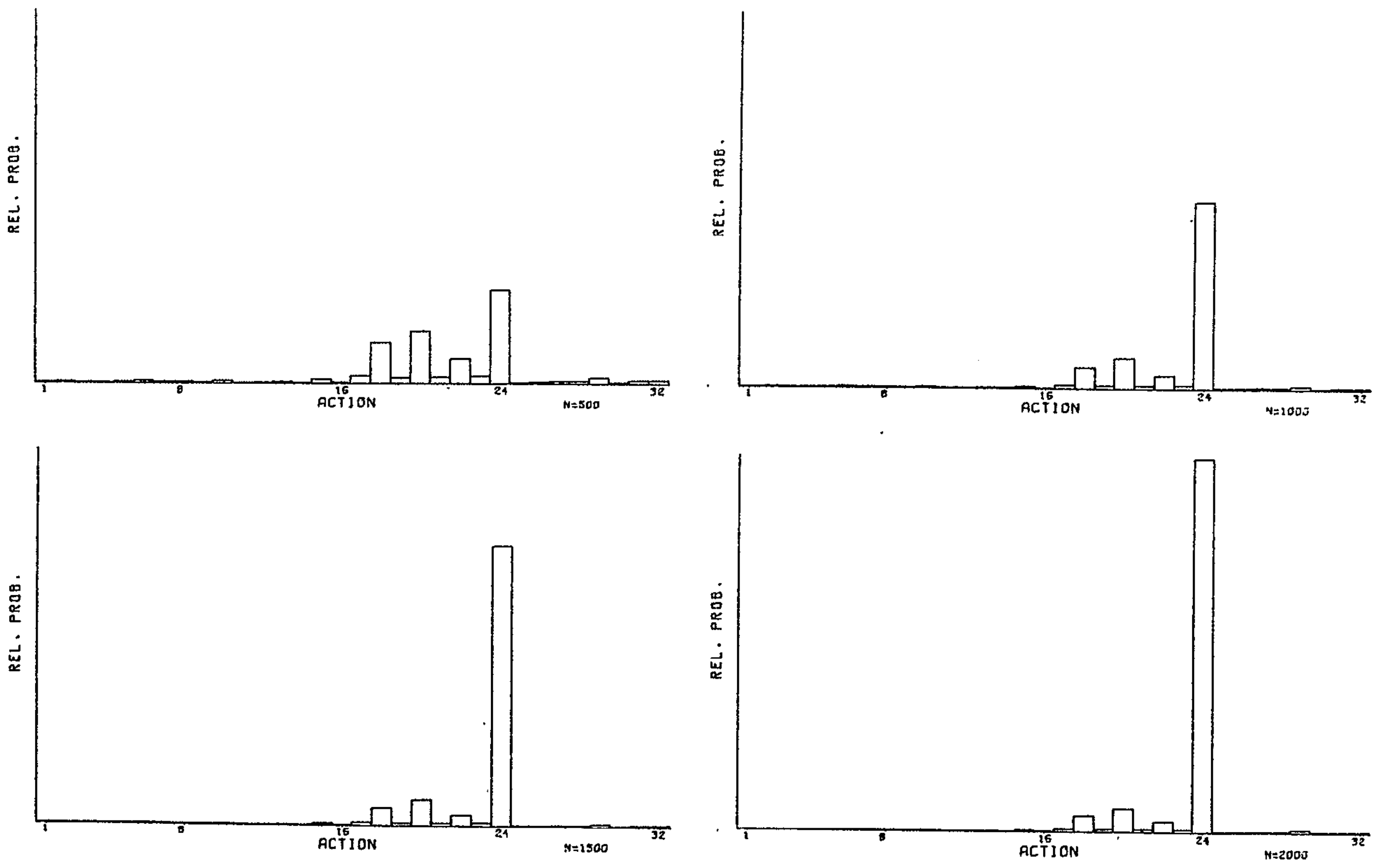


Figure 33 Distribution Curves (6)

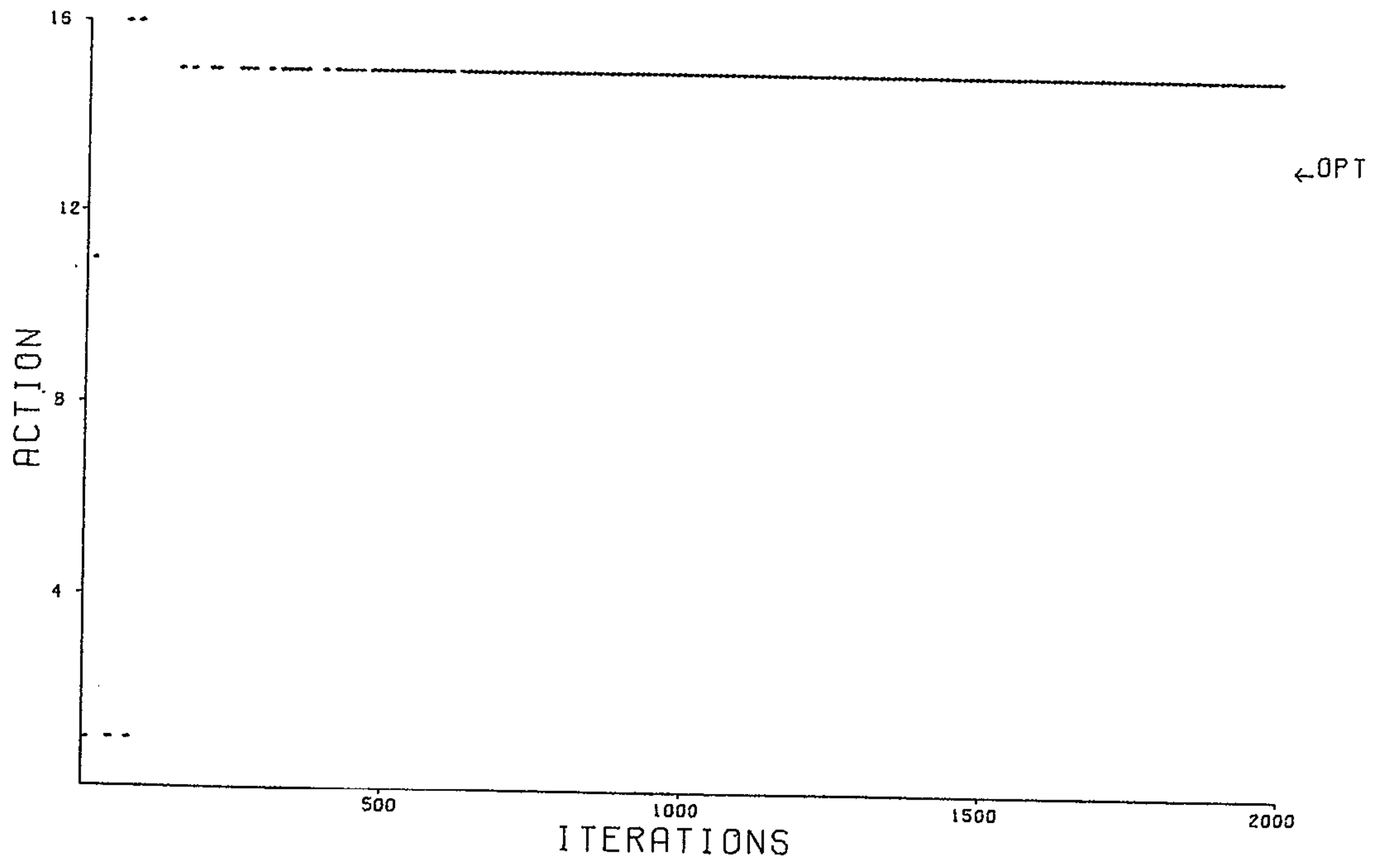


Figure 34 Output Map (7)

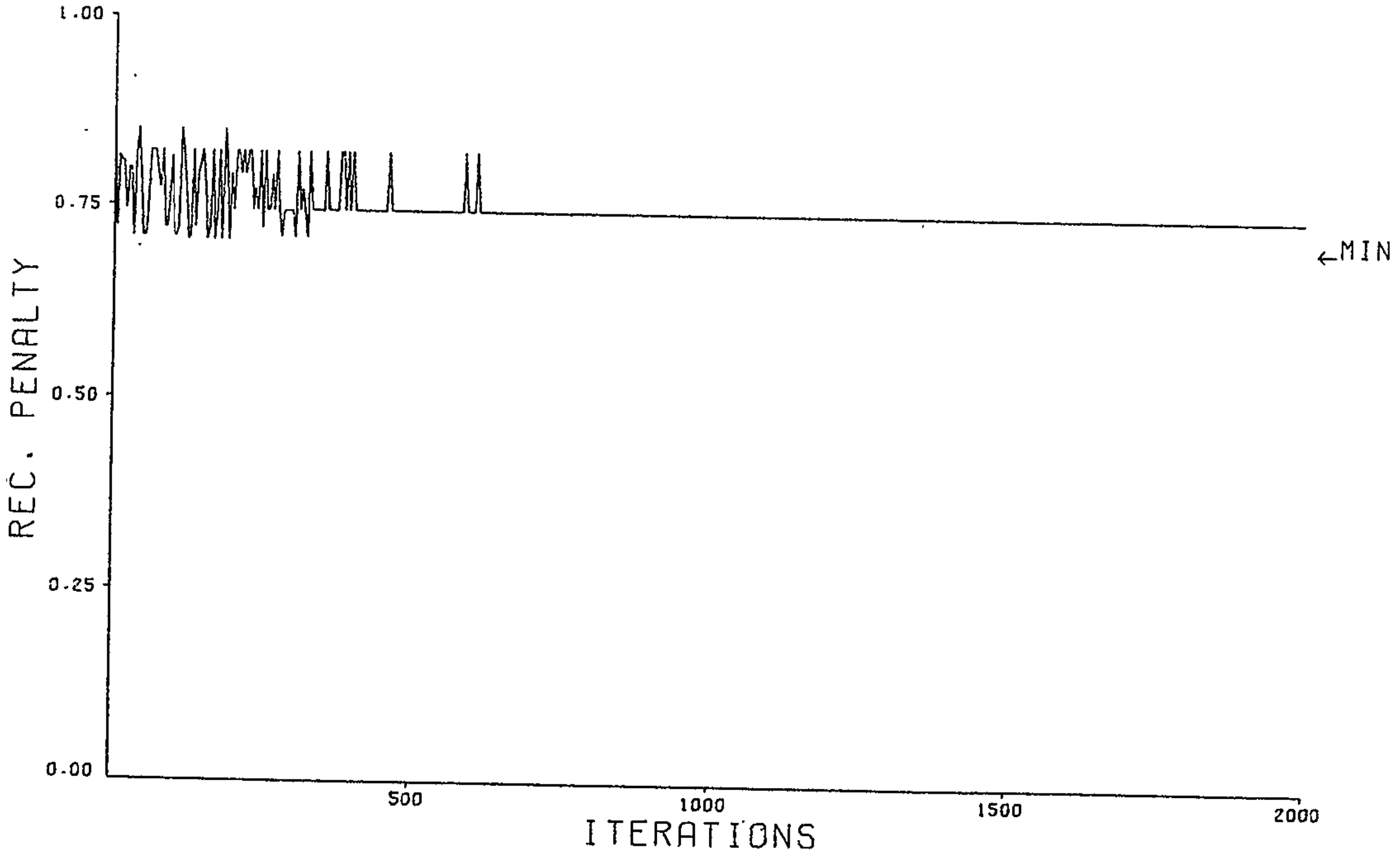


Figure 35 Penalty Curve (7)

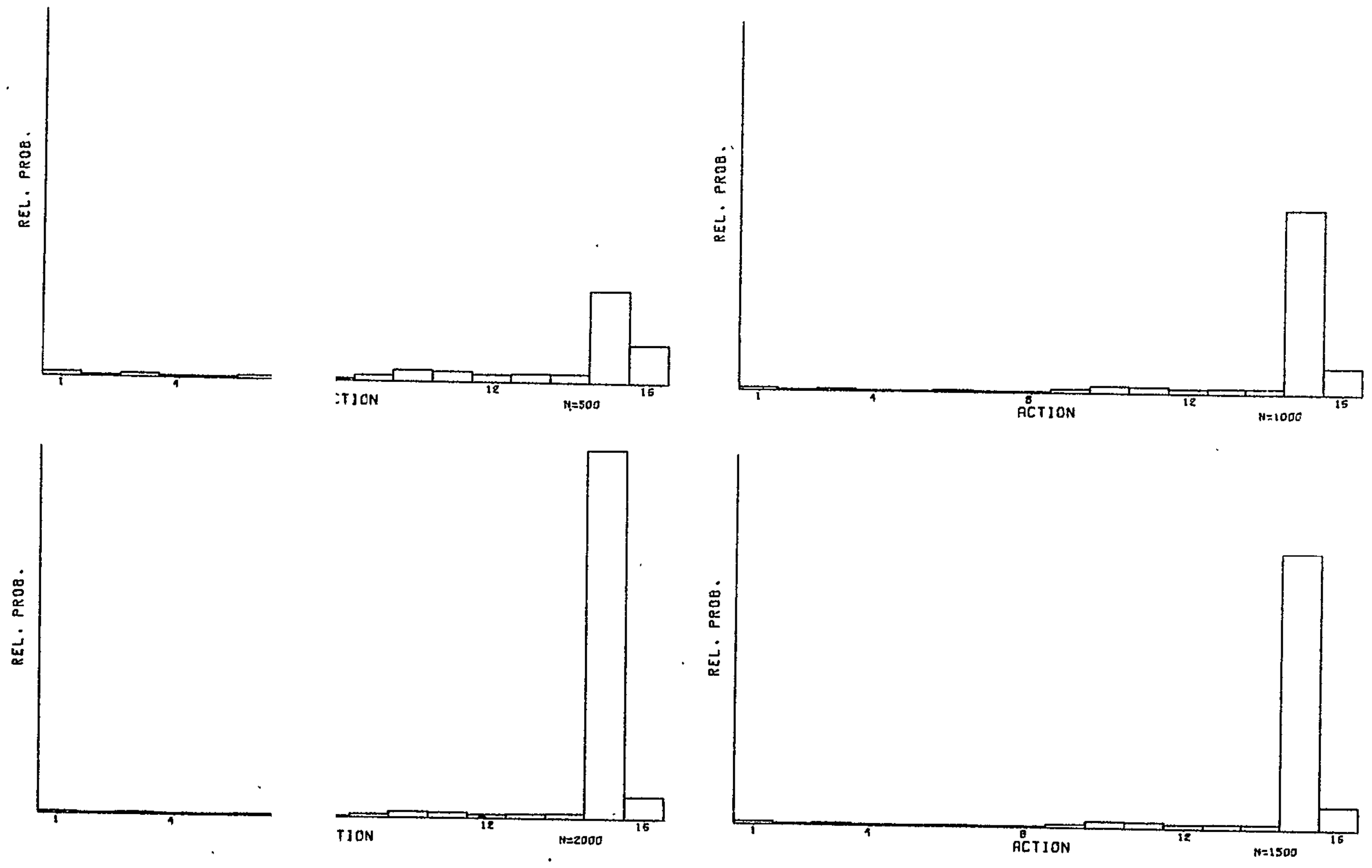


Figure 36 Distribution Curves (7a)

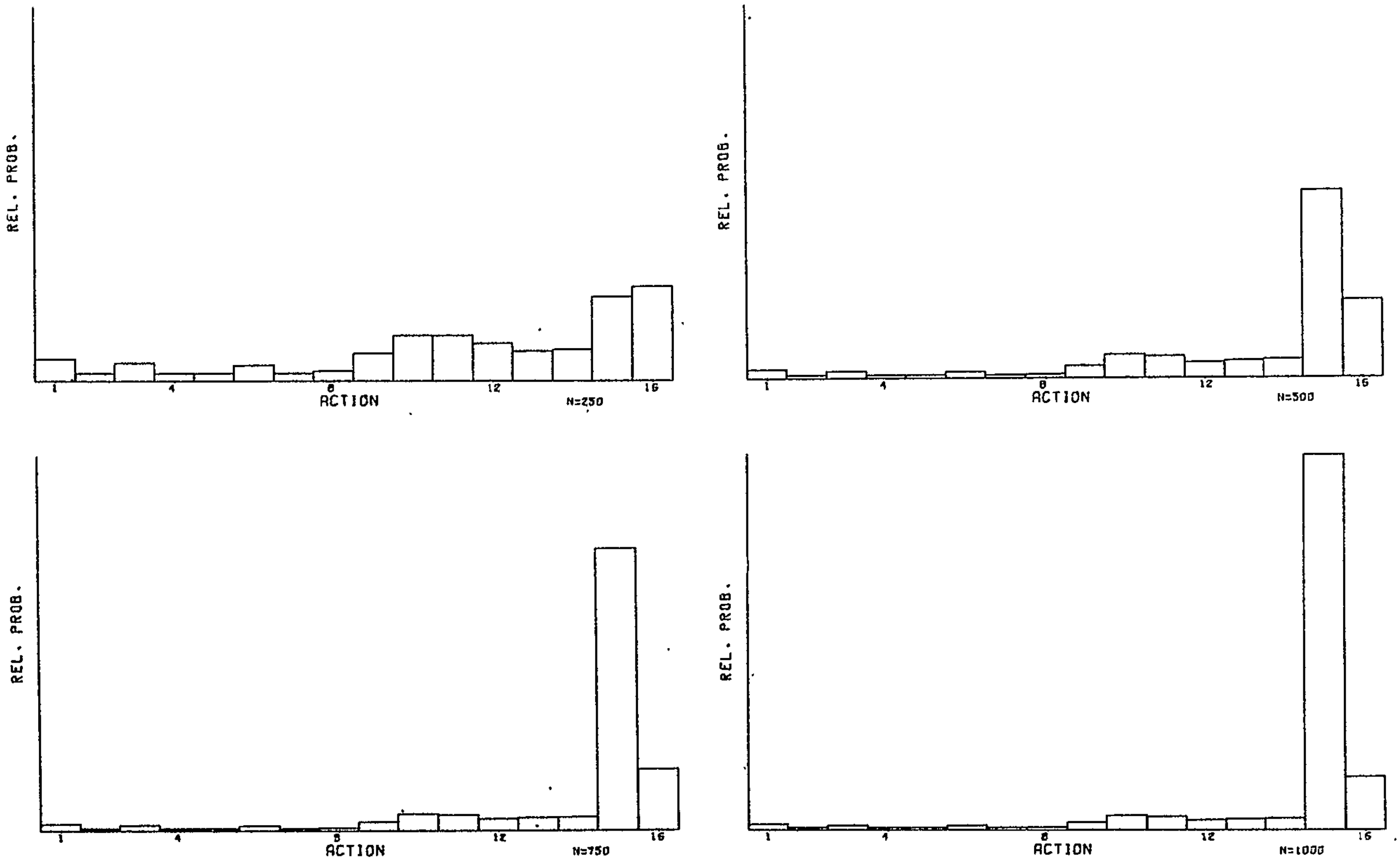


Figure 37 Distribution Curves (7b)