Research Note 86-86

AD-A171 544

# ARTIFICIAL INTELLIGENCE THROUGH EVOLUTIONARY PROGRAMMING: PREDICTION AND IDENTIFICATION

Lawrence J. Fogel and David Fogel
Titan Systems, Inc.

Contracting Officer's Representative
George H. Lawrence



BASIC RESEARCH
Milton S. Katz, Director



U. S. Army
Research Institute for the Behavioral and Social Sciences

AUGUST 1986

# U. S. ARMY RESEARCH INSTITUTE

# FOR THE BEHAVIORAL AND SOCIAL SCIENCES

A Field Operating Agency under the Jurisdiction of the

Deputy Chief of Staff for Personnel


EDGAR M. JOHNSON
Technical Director

WM. DARRYL HENDERSON
COL, IN
Commanding



Research accomplished under contract for
the Department of the Army

Titan Systems, Inc.

Technical review by

Steve Kronheim


# REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: ARI Research Note 86-86
2. GOVT ACCESSION NO.:
3. RECIPIENT'S CATALOG NUMBER:

4. TITLE (and Subtitle): ARTIFICIAL INTELLIGENCE THROUGH EVOLUTIONARY PROGRAMMING: PREDICTION AND IDENTIFICATION
5. TYPE OF REPORT & PERIOD COVERED: Interim, February 1986 - April 1986
6. PERFORMING ORG. REPORT NUMBER:

7. AUTHOR(s): Lawrence J. Fogel and David Fogel
8. CONTRACT OR GRANT NUMBER(s): PO-9-X56-1102C-1

9. PERFORMING ORGANIZATION NAME AND ADDRESS: Titan Systems, Inc., P.O. Box 12139, La Jolla, California 92037
10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: 2Q161102B74F

11. CONTROLLING OFFICE NAME AND ADDRESS: U.S. Army Research Institute for the Behavioral and Social Sciences. 5001 Eisenhower Avenue, Alexandria, Virginia 22333-5600
12. REPORT DATE: August 1986
13. NUMBER OF PAGES: 191

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

Contracting Officer's Representative, George H. Lawrence.

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

evolutionary programming
single state machines
noisy environments
prediction and control
artificial intelligence
forecasting

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

This report examines the manner by which evolutionary programming treats an arbitrary prediction problem. Additional experiments were conducted to clarify uncertainties given increased machine size and the cost/benefit of retaining "offspring" programs, the impact of noise on the predictive capability of the evolutionary process, and the efficacy of crossover as a mechanism for improving simulated evolution. It was also found that some difficult combinatorial problems such as the classic Traveling Salesman Problem can be addressed through less complex logics.

Evolution, it is often said, is the most patient of engineers, redesigning organisms piece-by-piece in the tiniest of increments, using the grand and gradual sweep of geologic time as a drafting table.*

---

*Omni Magazine, March 1986, page 20

TABLE OF CONTENTS

# FOREWORD

Military decisions hinge on an ability to forecast the developing situation. A sufficient understanding of the enemy's response behavior sets the stage for causing the desired response. Clearly, prediction and control are essential aspects of the decision process. And yet, classical techniques treat only a limited portion of such real world situations. Least mean squared error reduction is rarely the appropriate criterion. The "plant" or transducer is neither linear nor passive.

The usual approach is to extend the classical techniques through quasi-linear approximations and higher ordered differential functions. But, such progress leads to ever more complex formulations that are increasingly unsuitable to primitive computation, and furthermore requires an in-depth understanding of the physical process which is not always available.

What is needed is an alternate formulation... a new beginning based on a different view of the logical process required for prediction, control, and other aspects of intelligence. Evolutionary programming constitutes such an approach... and simple high speed parallel processing will be notably suitable for such fast time simulation of the evolutionary process. The task at hand is to devise such programming in support of military prediction control and decision processes while, at the same time, to extend our intellectual capability.

# INTRODUCTION

The first Quarterly Progress Report described experiments in the prediction of diverse time series against various payoff functions through the use of Evolutionary Programming. It is now worthwhile to resolve some uncertainty concerning the structure of this program and to dimensionalize the use of this predictive capability. Given an arbitrary prediction problem, how can one determine the manner in which it should be treated through Evolutionary Programming?

Some additional experiments were performed to clarify uncertainties and to prepare for conducting a significant number of larger scale experiments to be performed on the NASA Ames Cray computer.

Various authors have suggested the use of crossover as a mechanism for improving simulated evolution. Some experiments were performed to explore the worth of this concept. Another series of experiments were conducted to demonstrate Evolutionary Programming within the context of identifying an arbitrary unknown transducer. Some difficult combinatorial problems such as the classic Traveling Salesman Problem can be addressed through the evolution of less complex logics (for example, single state machines). A demonstration in this regard is contained in Appendix.

## DISCUSSION

### Additional Experiments on Prediction

#### Increased Machine Size and the Saving of Equal-Worth Offspring

In view of the previous experiments, it remains unclear as to whether or not it is worthwhile to save offspring that are of the same worth as their parent. It is also of interest to enquire as to the worth of allowing the evolution of larger finite state machines. The previously used evolutionary program was altered to save equally worthwhile offspring and to permit finite state machines of up to fifty states, this in the hope that an increase in size will improve the predictive capability. In addition, offspring of equal worth to the parent were also saved.

The first experiment required prediction of the same binary cyclic environment used in the original experiments (101110011101). As expected, in the first experiment, a perfect predictor was found, but this logic was not discovered any faster. Figure 1 shows the predictive fit, while Figure 2 indicates the cumulative percent correct prediction. Note that perfect predictions were always made after the $170^{th}$ prediction. Figure 3 indicates that adding a state is a beneficial form of mutation. The rate of increase in the size of the machines follows the probability of adding a state. Note that as the evolutionary process proceeds, the percentage of such beneficial mutations (as compared with all possible mutations) gets smaller, and therefore the evolutionary process "slows down." This suggests that after a given complexity is reached, it might be appropriate to include a penalty for that complexity or an increase in the probability of deleting a state.

#### The Prediction of Noisy Environments

Ten experiments were conducted to examine the impact of noise on the predictive capability of the evolutionary process. The same binary cyclic environment was corrupted by having each symbol change, this with a

# 12 symbol cyclic with a maximum of 50 state machines

## General Conditions

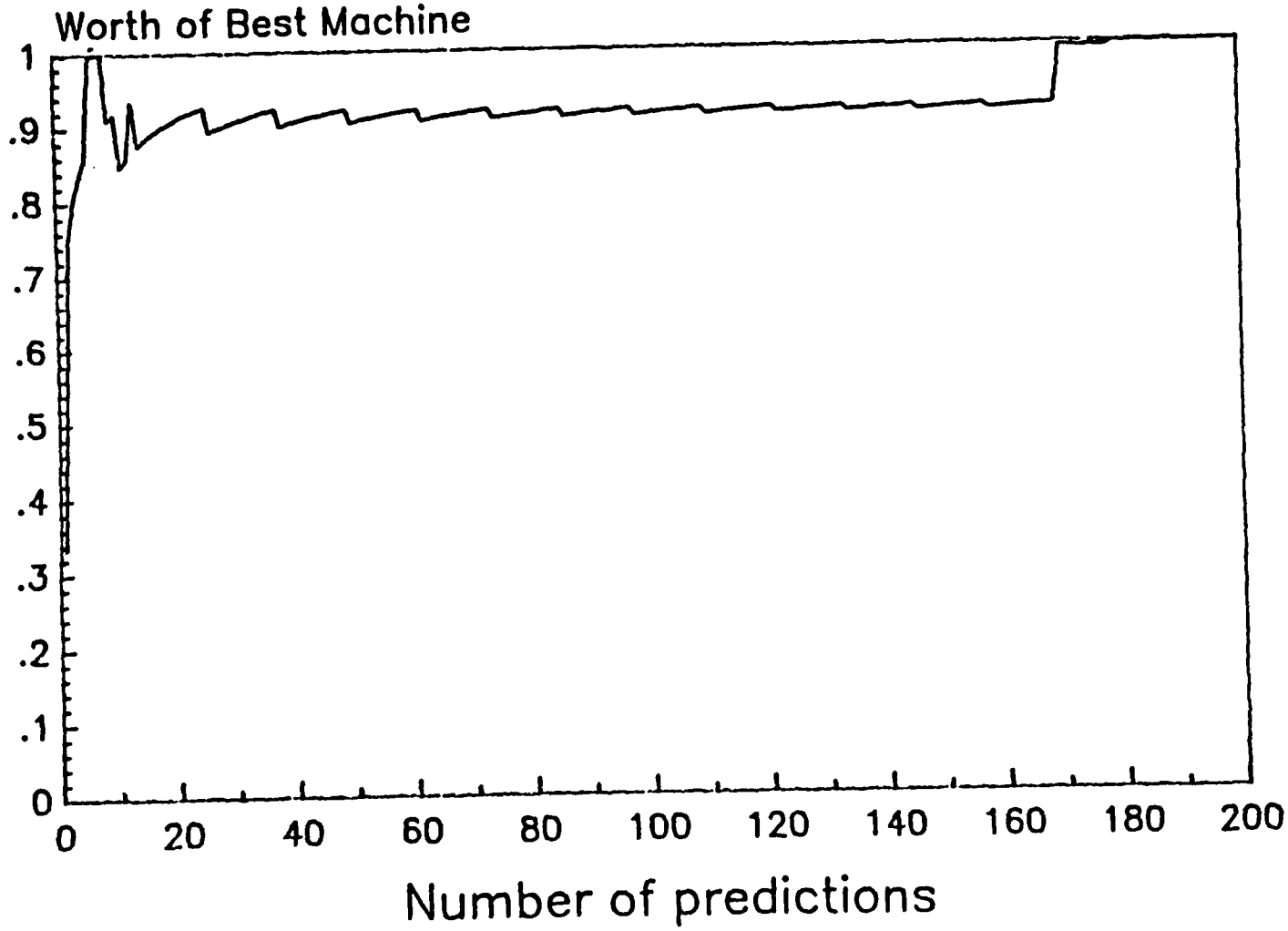


Figure 1

# 12 symbol cyclic with a maximum of 50 state machines

## General Conditions

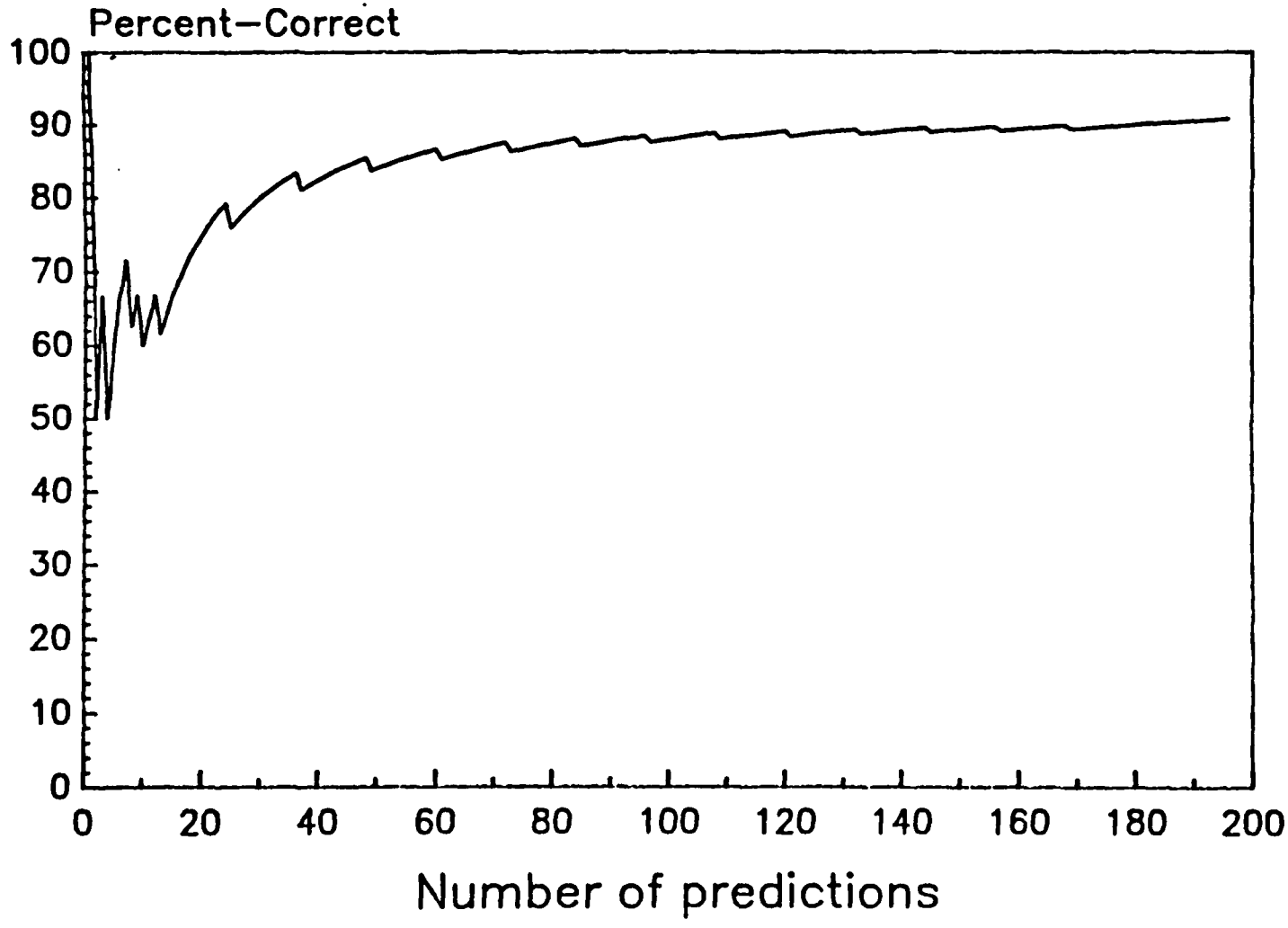


Figure 2

# 12 symbol cyclic with a maximum of 50 state machines

## General Conditions

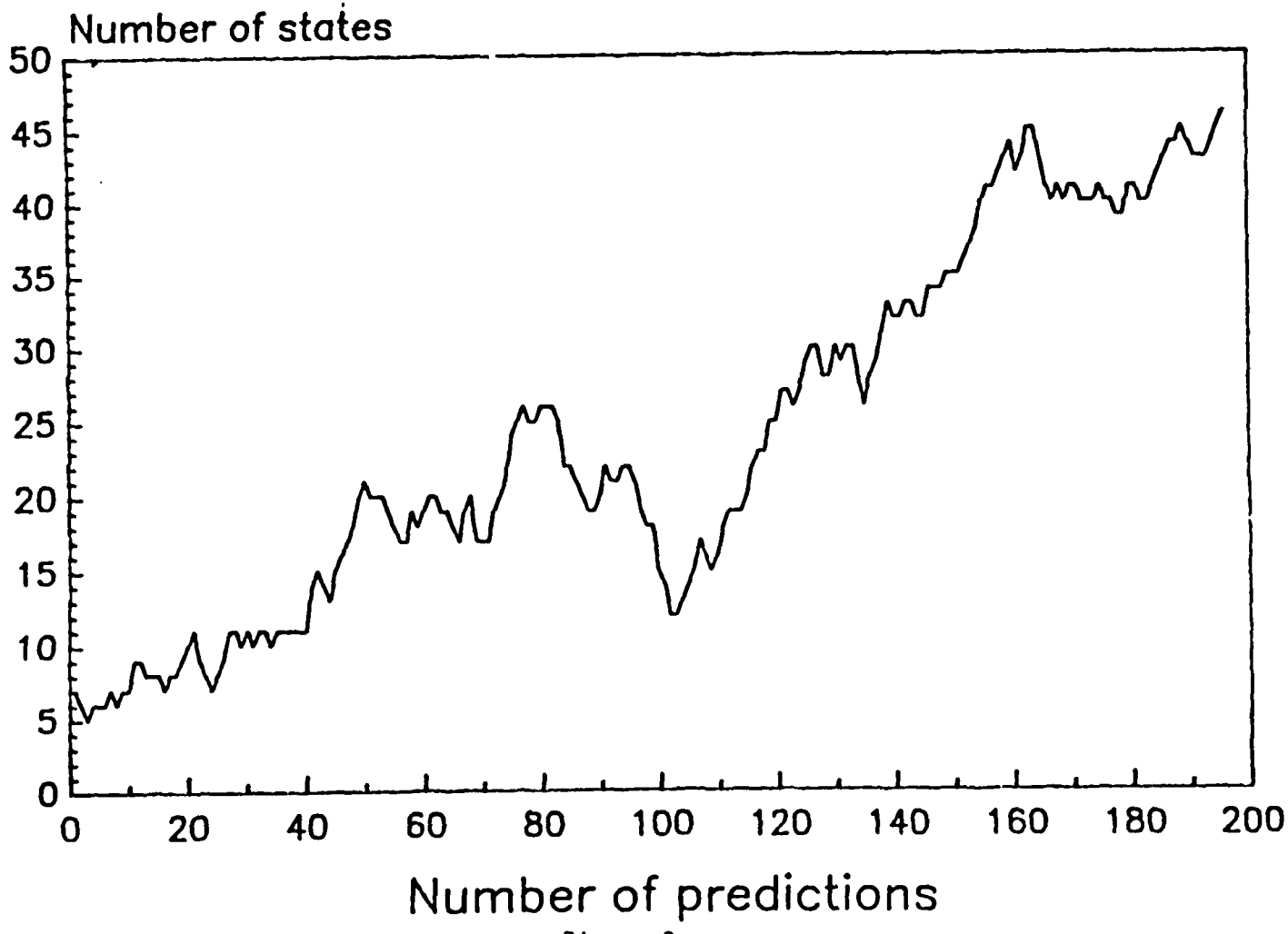


Figure 3

probability of one-sixth. Figure 4 shows the best of these experiments with the predictive fit score being very close to the expected value for a machine that perfectly fits the underlying cycle. Note that the excursion of the trace above the dashed line indicates that the evolving finite state machines were fitting the experienced noise. As shown in Figure 5, the evolution achieves about 65 percent correct predictions. The size of the machines grew rapidly to an upper limit to twenty-five states because of the noise, reference Figure 6.

Figure 7 shows the worst of the ten experiments. Here the predictive fit score was about ten percent below what could be expected. After 196 predictions, there was no evidence that the evolutionary process could predict better than fifty percent, see Figure 8. Figure 9 again reveals a rapid increase in the size of the evolving finite state machines.

Figures 10 and 11 indicate the mean and two sigma limits for the ten experiments. The mean shows a steady but slow increase (probably to 83 1/3 percent, the highest possible percent correct for this noisy environment). The two sigma "confidence limits" converge as expected. The mean worth (average predictive fit) was very close to the maximum expected value and had relatively narrow confidence limits, see Figures 12 and 13. Note that if the environment is noisy and an all-or-none payoff function is imposed, the evolutionary process constructs machines that fit both signal and noise with equal weight. This is particularly apparent if the environment is binary.

A brief experiment required the prediction of a noisy four-symbol environment. Here the noise was quasi-Gaussian... a 67 percent chance of altering each symbol ± 1 and a 33 percent chance of altering each symbol ± 2. See Figure 14. A linear payoff function was used to provide some "incentive" for predicting close to the correct true symbol. This encourages discovering the signal as opposed to the noise. Figures 15 and 16 indicate the predictive fit and prediction capability generated through use of this less severe payoff function. Since the four-symbol environment was known to be simple (a two-state machine would perfectly predict the cycle), a 0.01 penalty per state was imposed.

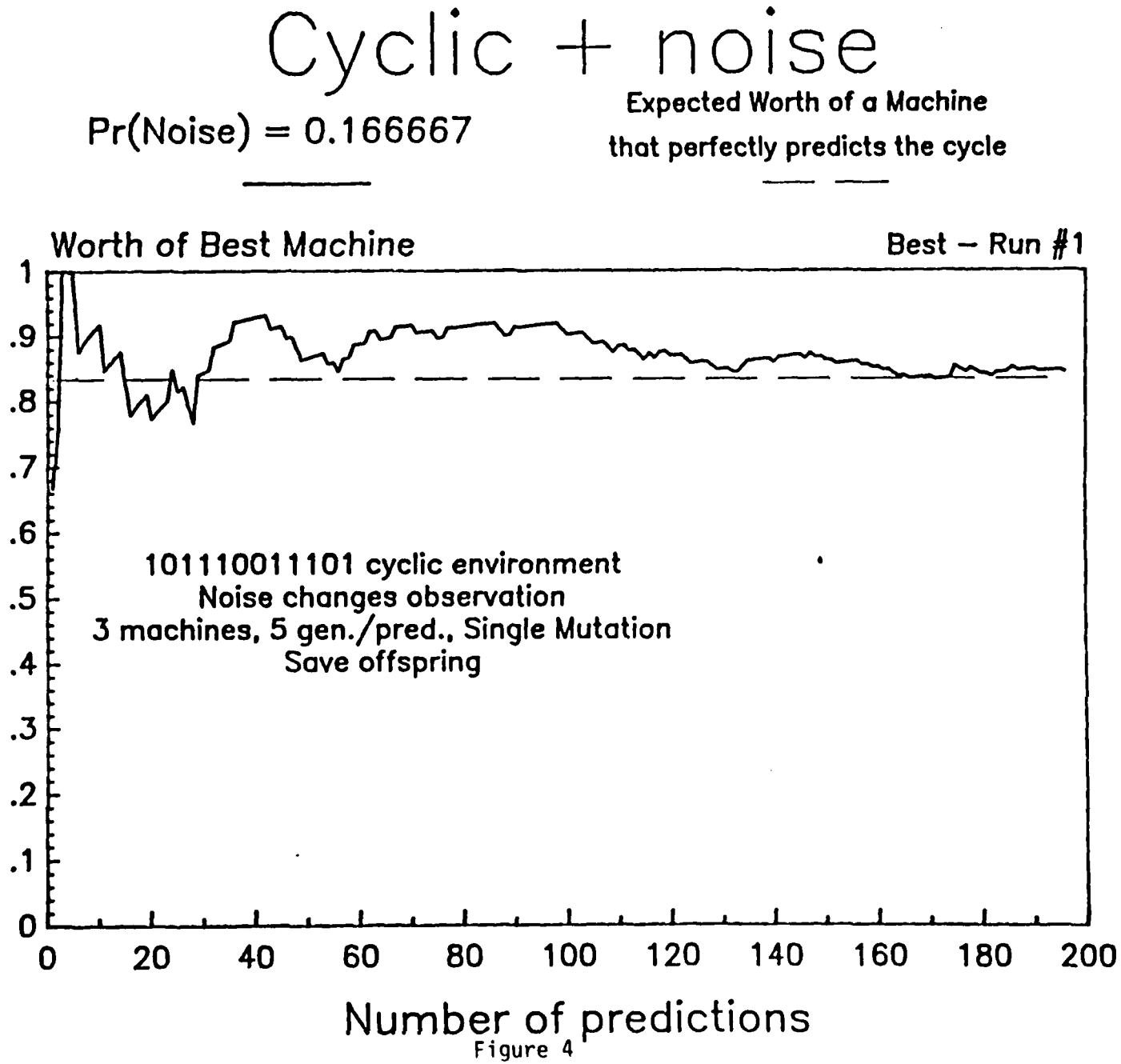
Cyclic + noise
Pr(Noise) = 0.166667
Expected Worth of a Machine
that perfectly predicts the cycle
Worth of Best Machine
Best – Run #1
1
.9
.8
.7
.6
.5
.4
.3
.2
.1
0
101110011101 cyclic environment
Noise changes observation
3 machines, 5 gen./pred., Single Mutation
Save offspring
0
20
40
60
80
100
120
140
160
180
200
Number of predictions

Figure 4

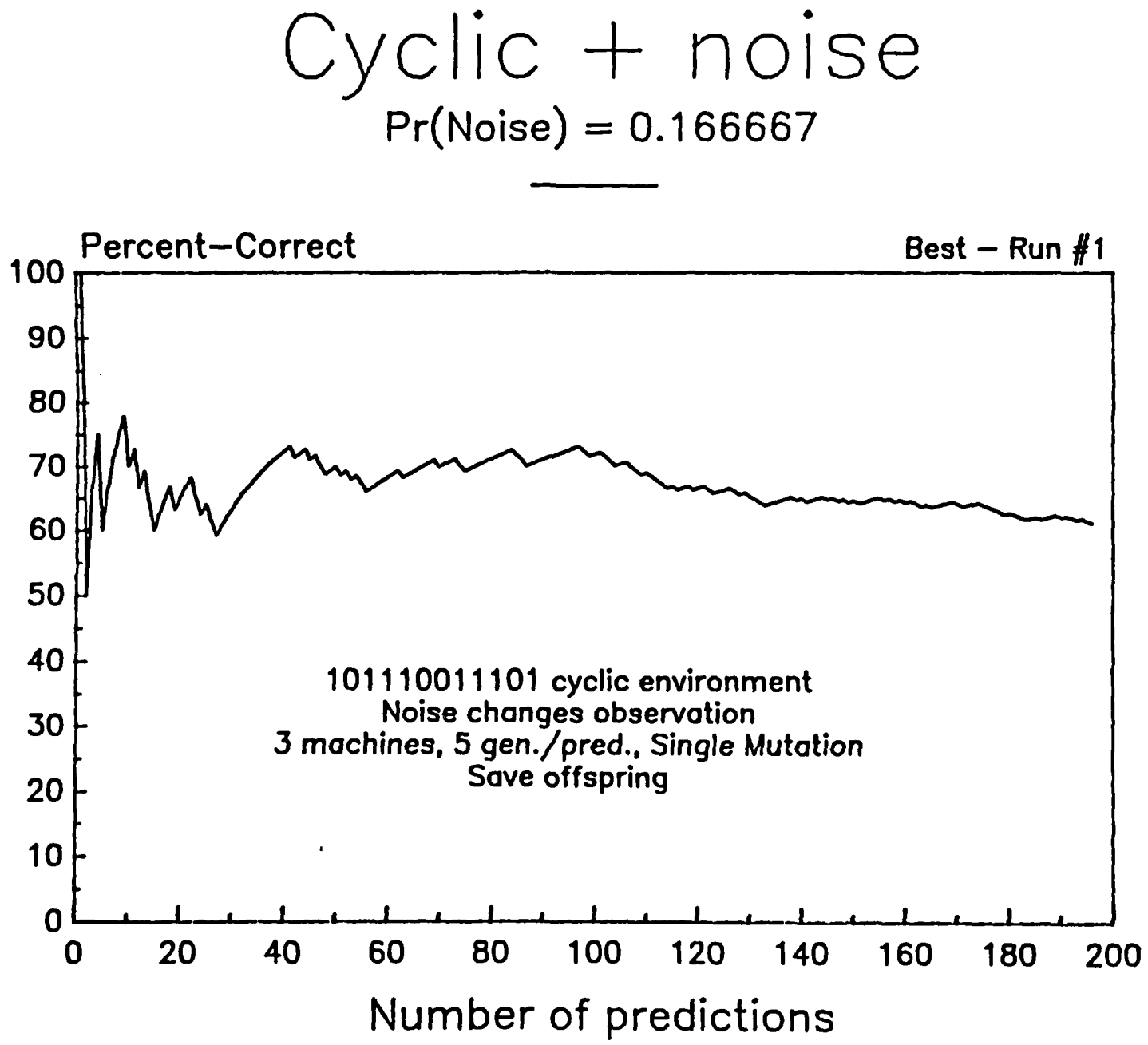
Cyclic + noise
Pr(Noise) = 0.166667
Percent–Correct
Best – Run #1
100
90
80
70
60
50
40
30
20
10
0
0
20
40
60
80
100
120
140
160
180
200
101110011101 cyclic environment
Noise changes observation
3 machines, 5 gen./pred., Single Mutation
Save offspring
Number of predictions

Figure 5

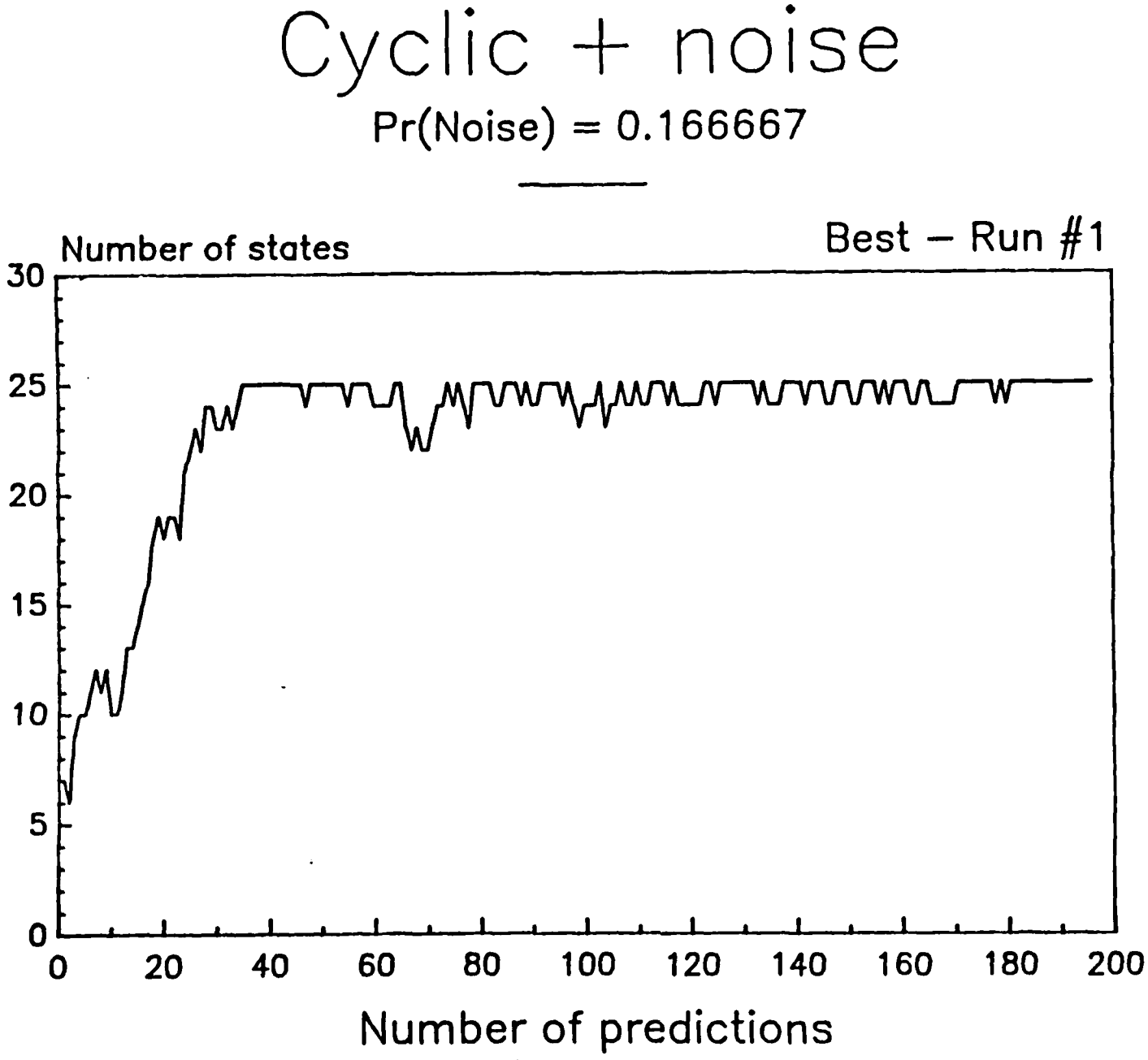
Cyclic + noise
Pr(Noise) = 0.166667
Number of states
Best – Run #1
30
25
20
15
10
5
0
0
20
40
60
80
100
120
140
160
180
200
Number of predictions

Figure 6

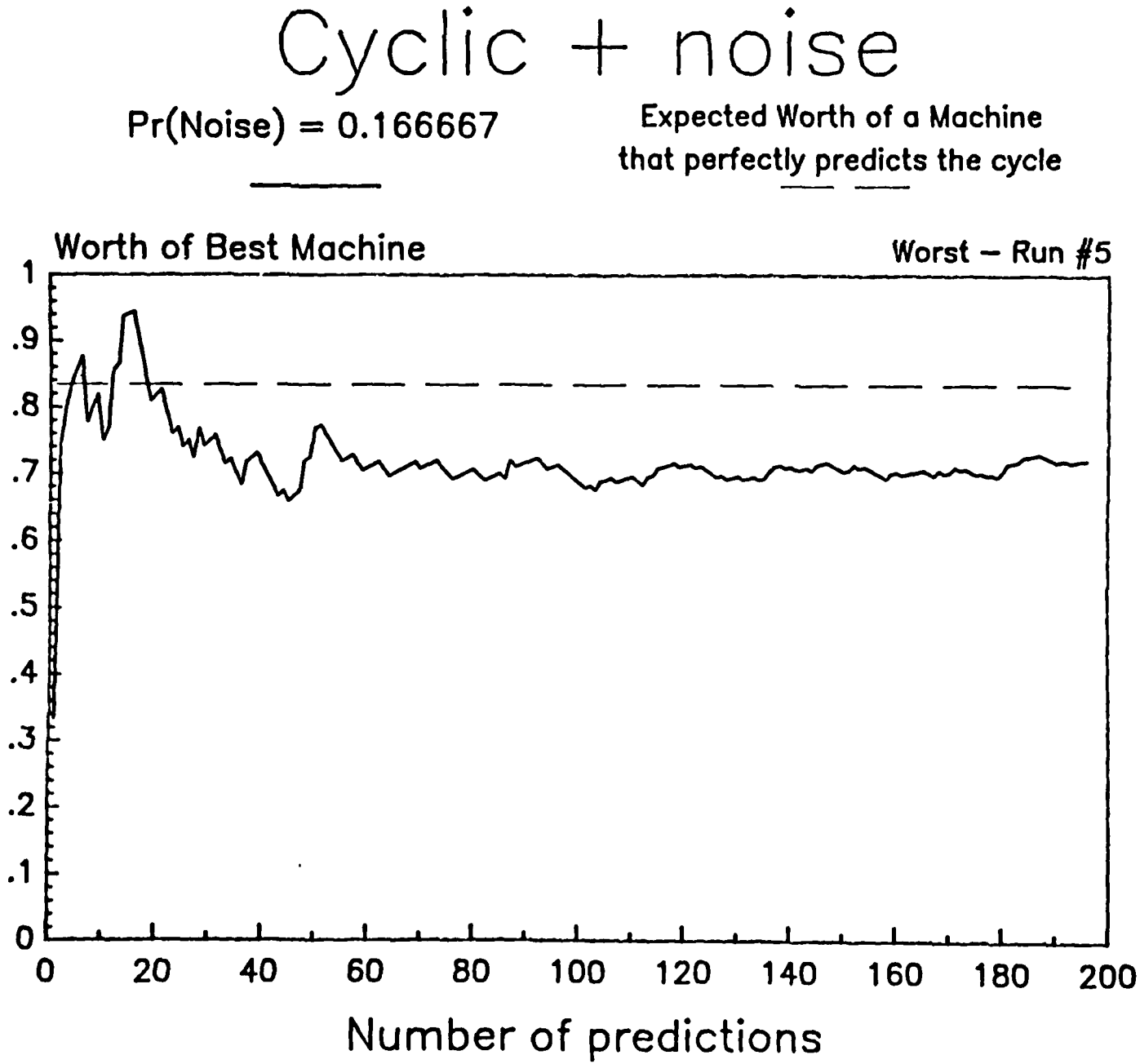
Cyclic + noise
Pr(Noise) = 0.166667
Expected Worth of a Machine
that perfectly predicts the cycle
Worth of Best Machine
Worst – Run #5
1
.9
.8
.7
.6
.5
.4
.3
.2
.1
0
0
20
40
60
80
100
120
140
160
180
200
Number of predictions

Figure 7

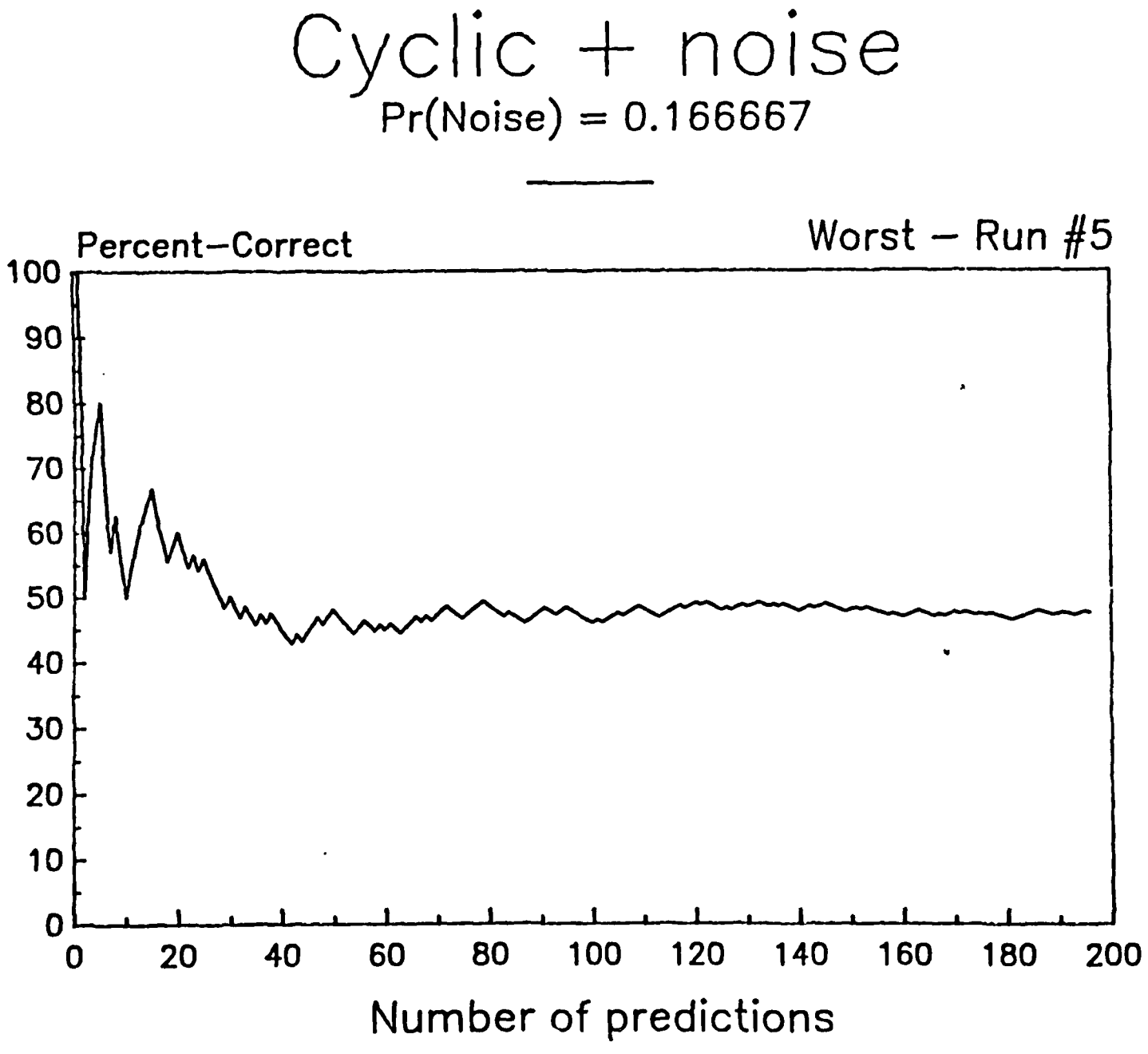
Cyclic + noise
Pr(Noise) = 0.166667
Percent–Correct
Worst – Run #5
100
90
80
70
60
50
40
30
20
10
0
0
20
40
60
80
100
120
140
160
180
200
Number of predictions

Figure 8

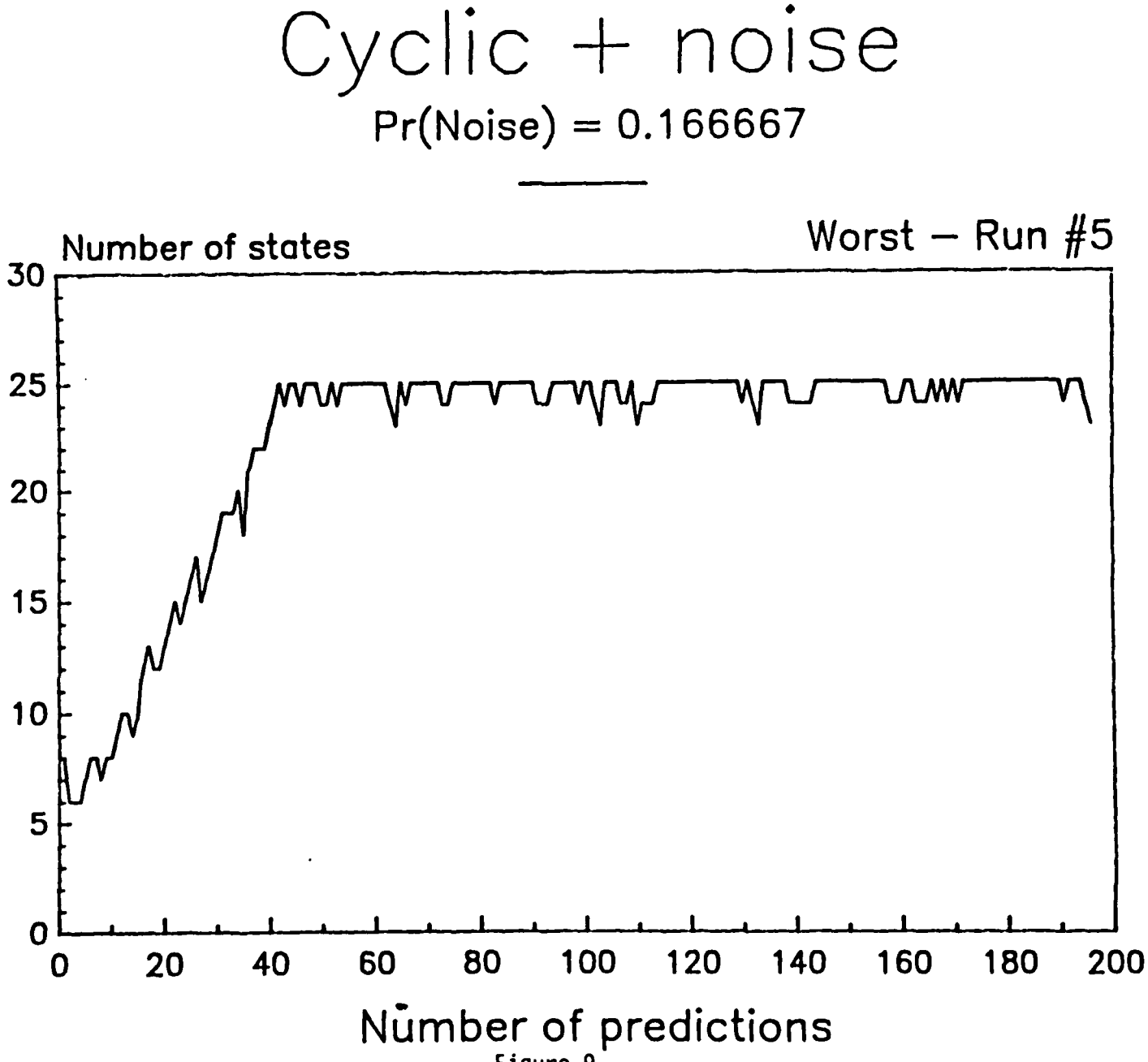
Cyclic + noise
Pr(Noise) = 0.166667
Number of states
Worst – Run #5
30
25
20
15
10
5
0
0
20
40
60
80
100
120
140
160
180
200
Number of predictions

Figure 9

# Cyclic + noise

## Mean Percentage

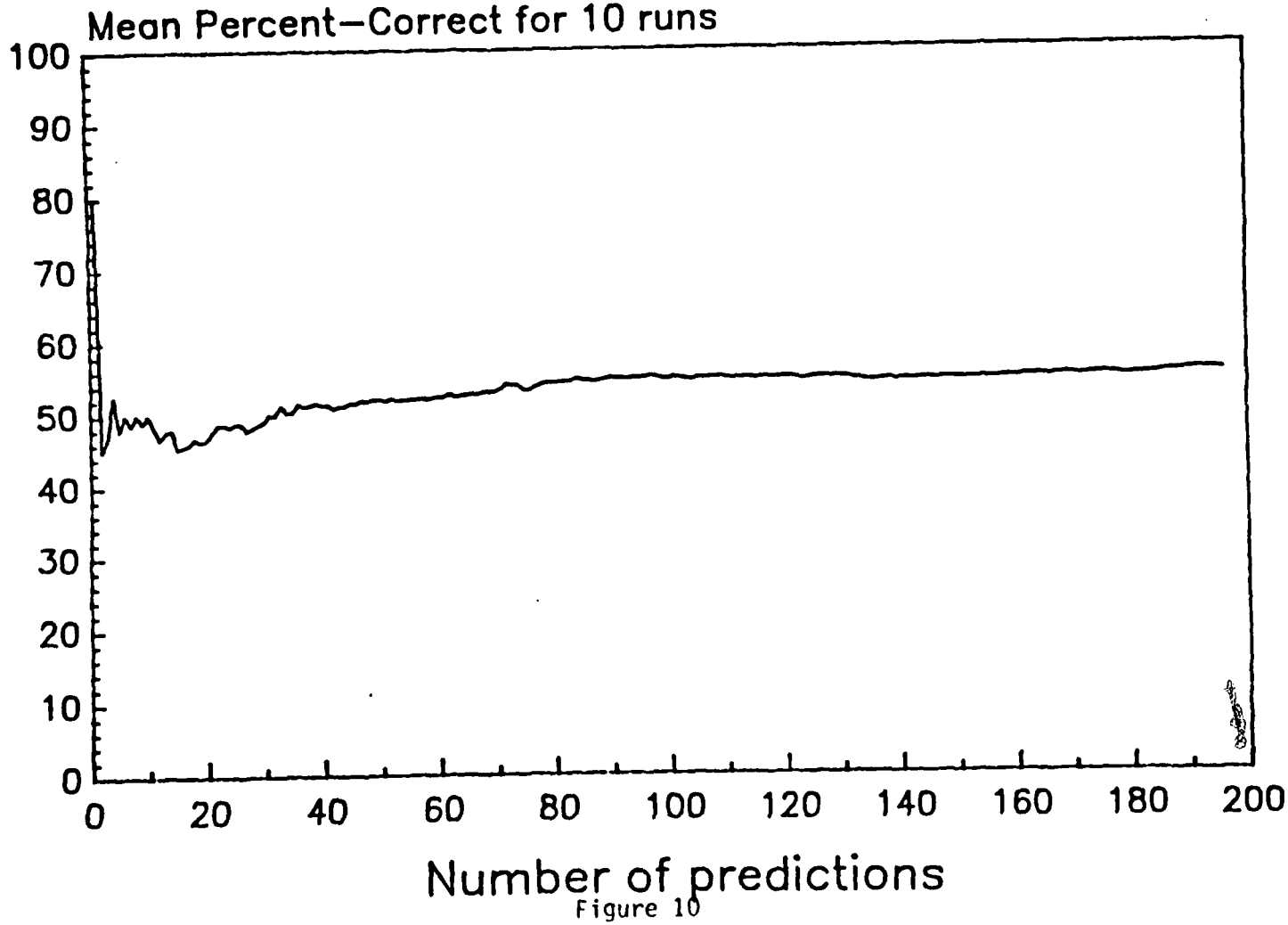


Figure 10



Figure 10

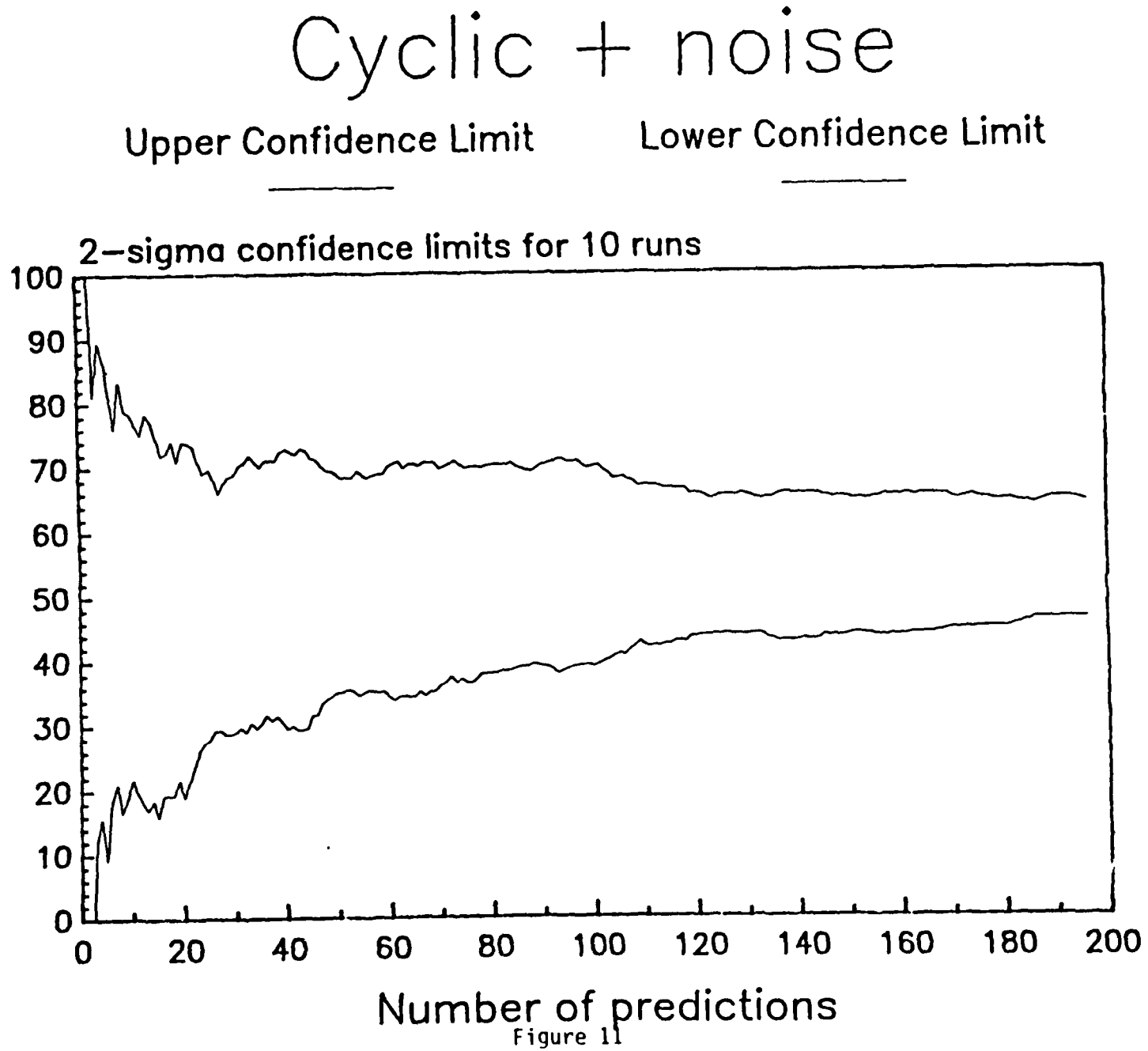
Cyclic + noise
Upper Confidence Limit
Lower Confidence Limit
2–sigma confidence limits for 10 runs
100
90
80
70
60
50
40
30
20
10
0
0
20
40
60
80
100
120
140
160
180
200
Number of predictions

Figure 11

# Cyclic + noise

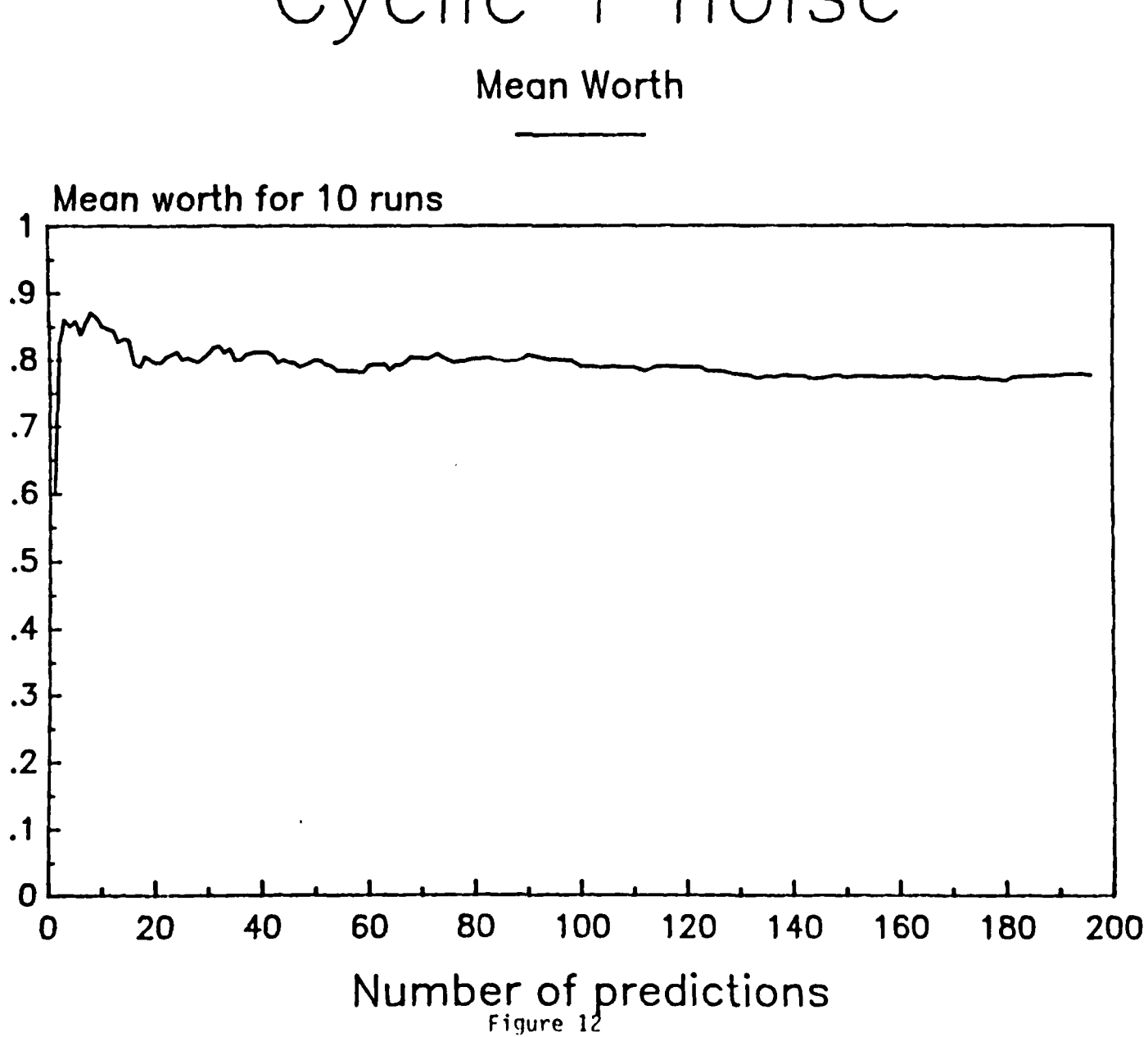


Figure 12



Figure 12

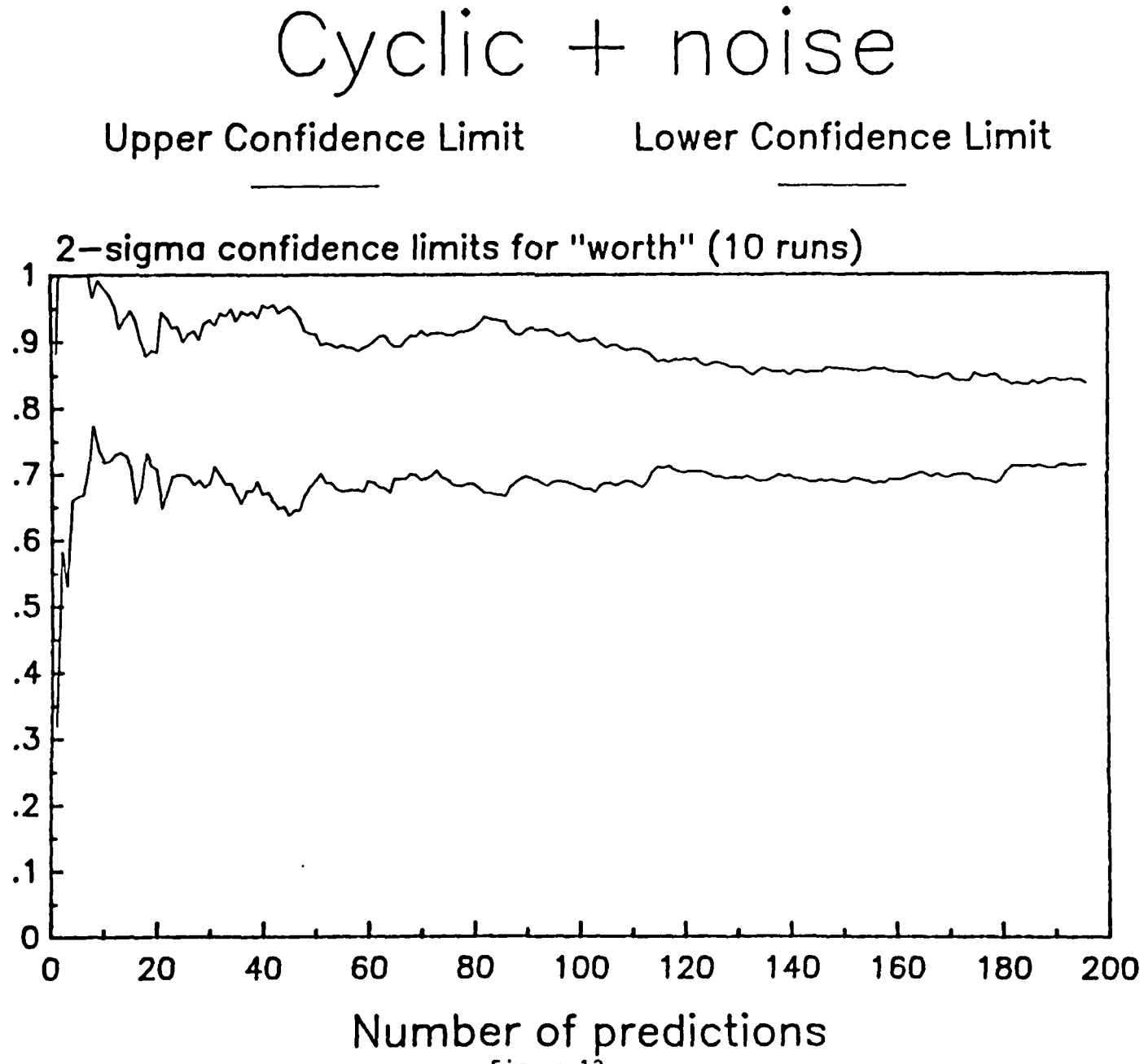
Cyclic + noise
Upper Confidence Limit
Lower Confidence Limit
2−sigma confidence limits for "worth" (10 runs)
1
.9
.8
.7
.6
.5
.4
.3
.2
.1
0
0
20
40
60
80
100
120
140
160
180
200
Number of predictions

Figure 13

# Graphic representation of the environment

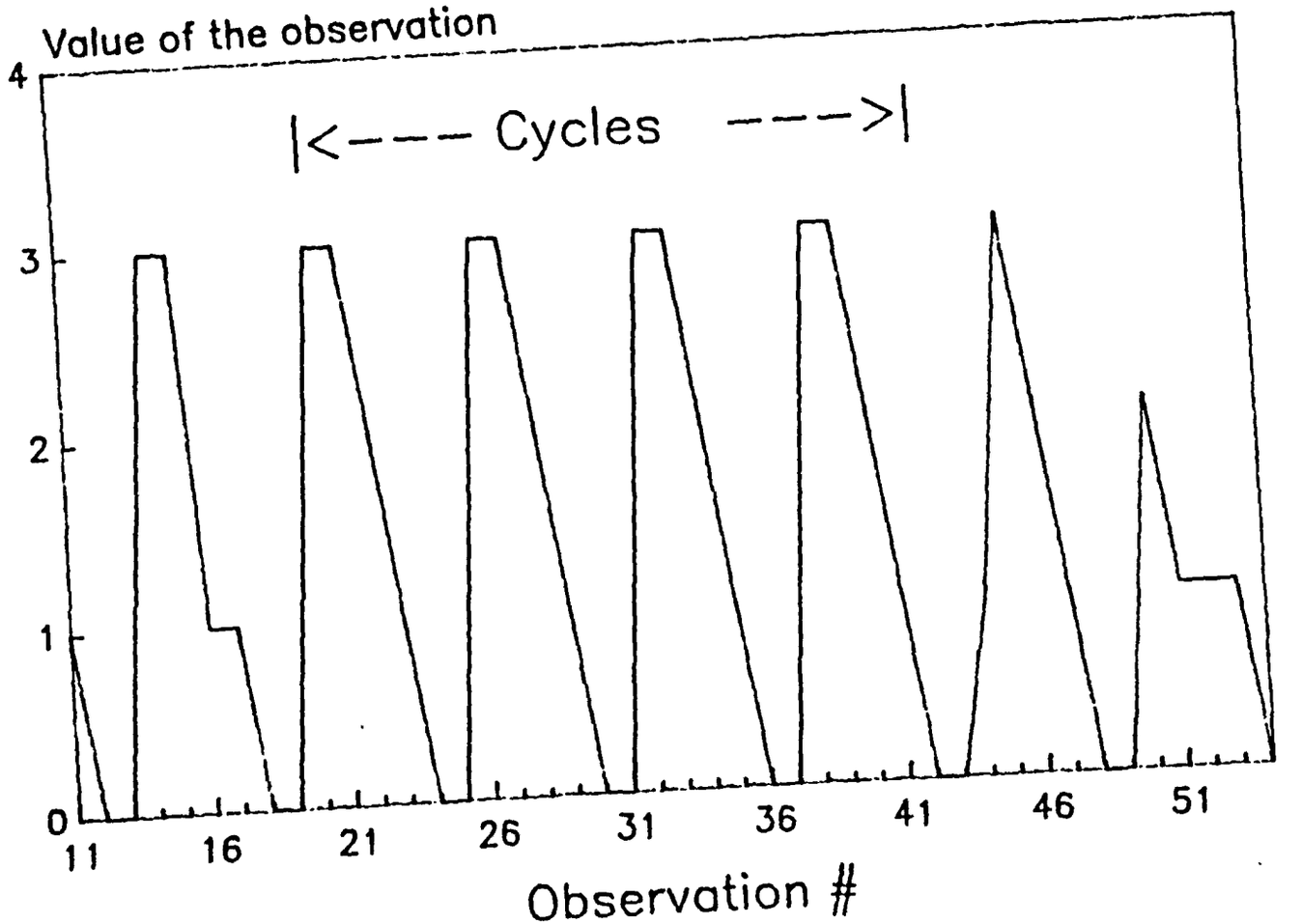


Figure 14



Figure 14

(033210) cyclic with Linear Payoff Function – Gaussian Noise (+/1 1,2)

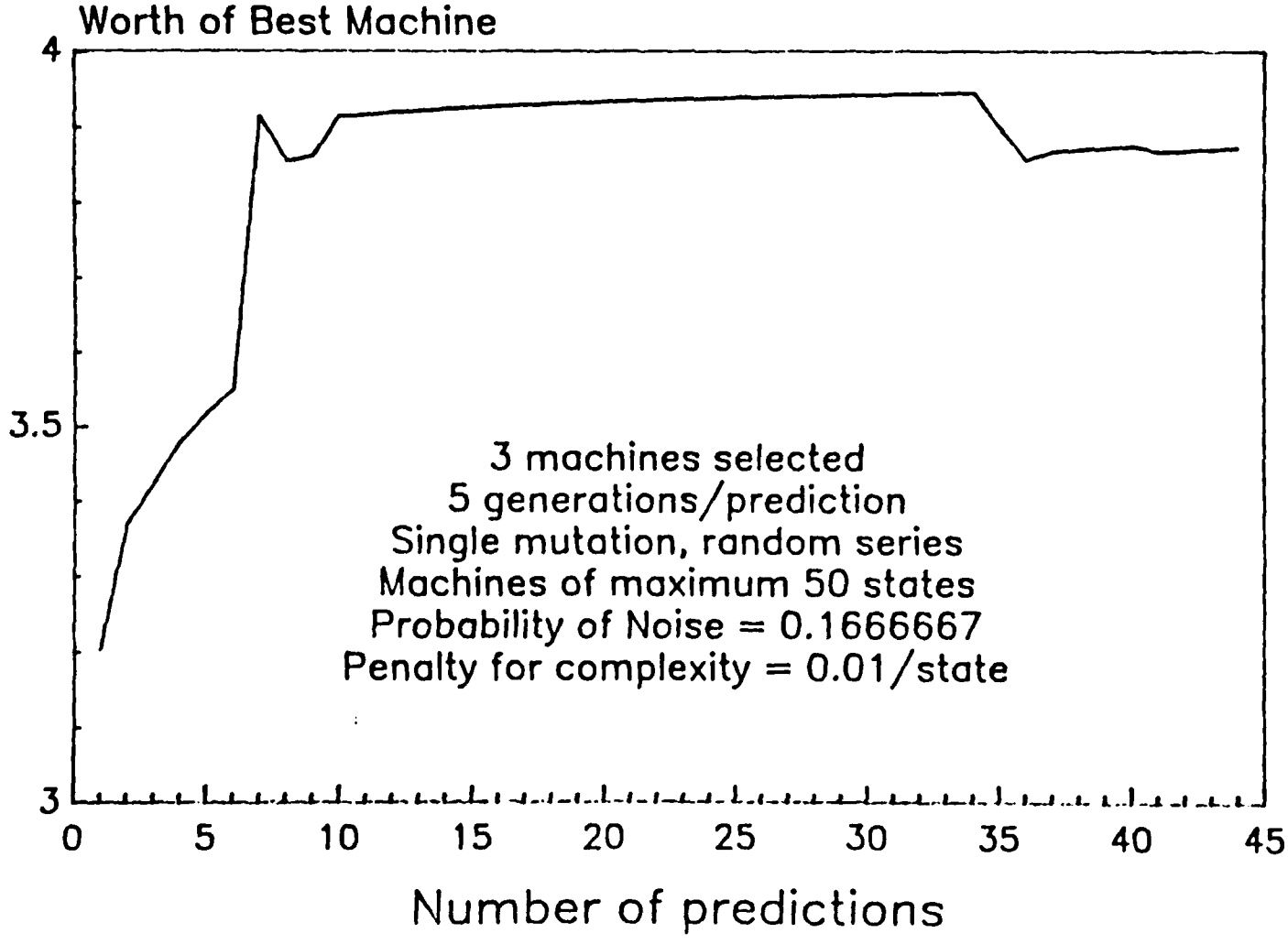


Figure 15

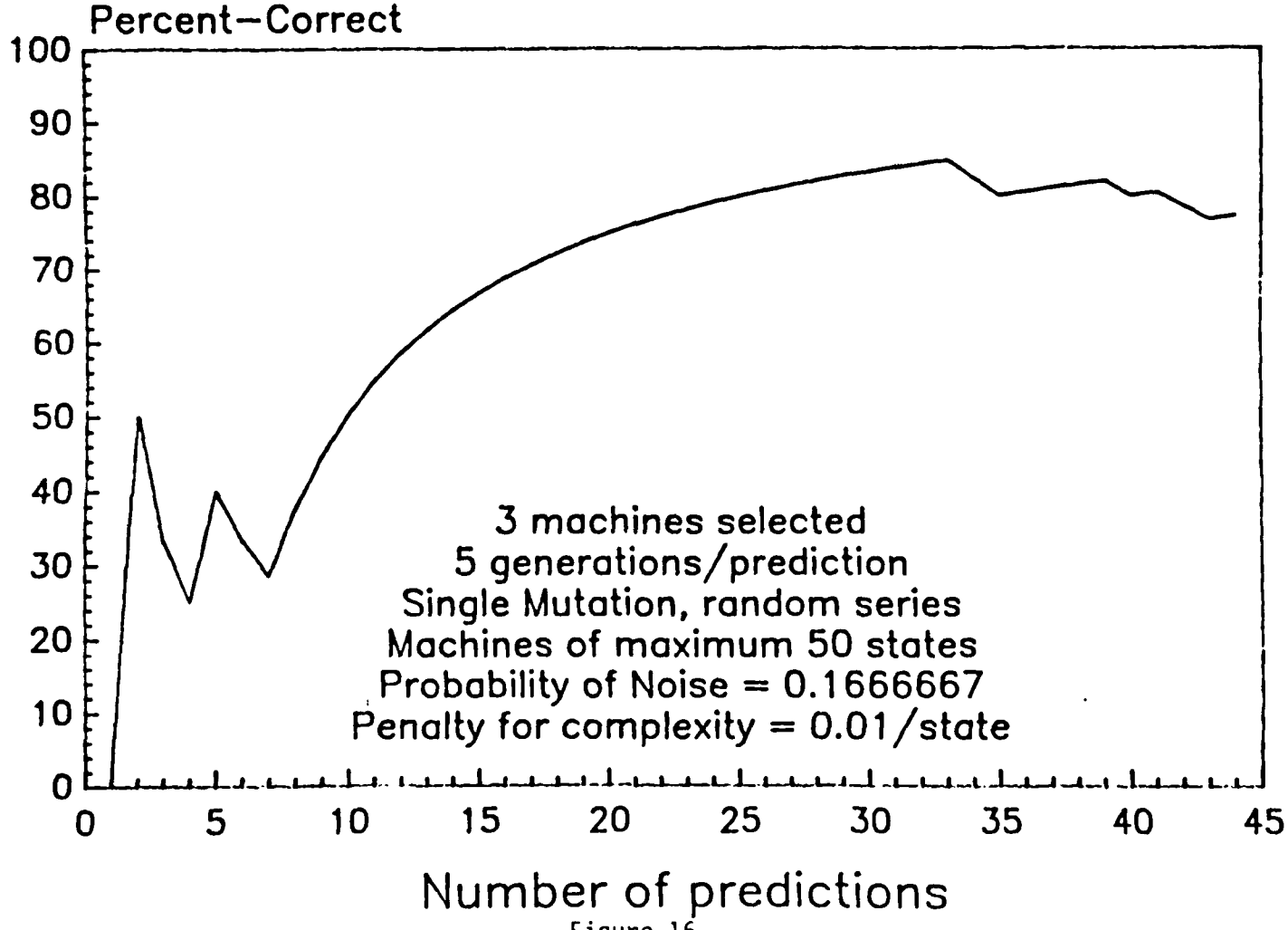
(033210) cyclic with Linear Payoff Function – Gaussian Noise (+/– 1,2)
Percent–Correct
100
90
80
70
60
50
40
30
20
10
0
0
5
10
15
20
25
30
35
40
45
3 machines selected
5 generations/prediction
Single Mutation, random series
Machines of maximum 50 states
Probability of Noise = 0.1666667
Penalty for complexity = 0.01/state
Number of predictions

Figure 16

## Saving Equal Offspring with a Penalty for Complexity

Nine experiments were performed on the above-referenced binary cyclic environment wherein offspring equal in predictive fit to the parent were saved, and a 0.01 penalty per state was imposed. In one experiment, an eight-state perfect predictor machine was found on the $63^{rd}$ prediction, see Figure 17. 12,603 offspring were evaluated in the 196 predictions. Figure 18 shows that the machines greatly increase in complexity as required to "solve this problem." Figures 19 and 20 show that the mean and two sigma limits were slightly better than those previously obtained, reference the first Quarterly Progress Report.

## Predicting the Fibonacci Series, Modulo-10

Having determined that saving offspring of equal worth to the parent can be of benefit, an experiment was performed to reveal the predictive capability of the evolutionary program with respect to an environment generated by the Fibonacci Series, Modulo-10. Figures 21 and 22 indicate significant improvement over the previously reported results. The evolutionary process was now able to "learn" more of the 60-symbol long cycle. After 3,020 predictions, a 42 percent cumulative predictive score was achieved. In principle, this environment would eventually be perfectly predicted by a ten-state machine.

## Predicting the "I.Q. Test" Environment

Seven experiments were performed on the typical "I.Q. test" environment 101001000... again saving offspring of equal worth; all other conditions being the same as those indicated in the first Quarterly Progress Report. Figures 23 through 29 show these experiments in terms of percent correct and demonstrate the adaptation with respect to an ever-changing environment. In the first experiment, the "discovery" that it is better to predict the zeros was not yet demonstrated because it had successfully predicted the early one's. The second experiment shows a distinct

# 101110011101 cyclic environment

## With a penalty for complexity

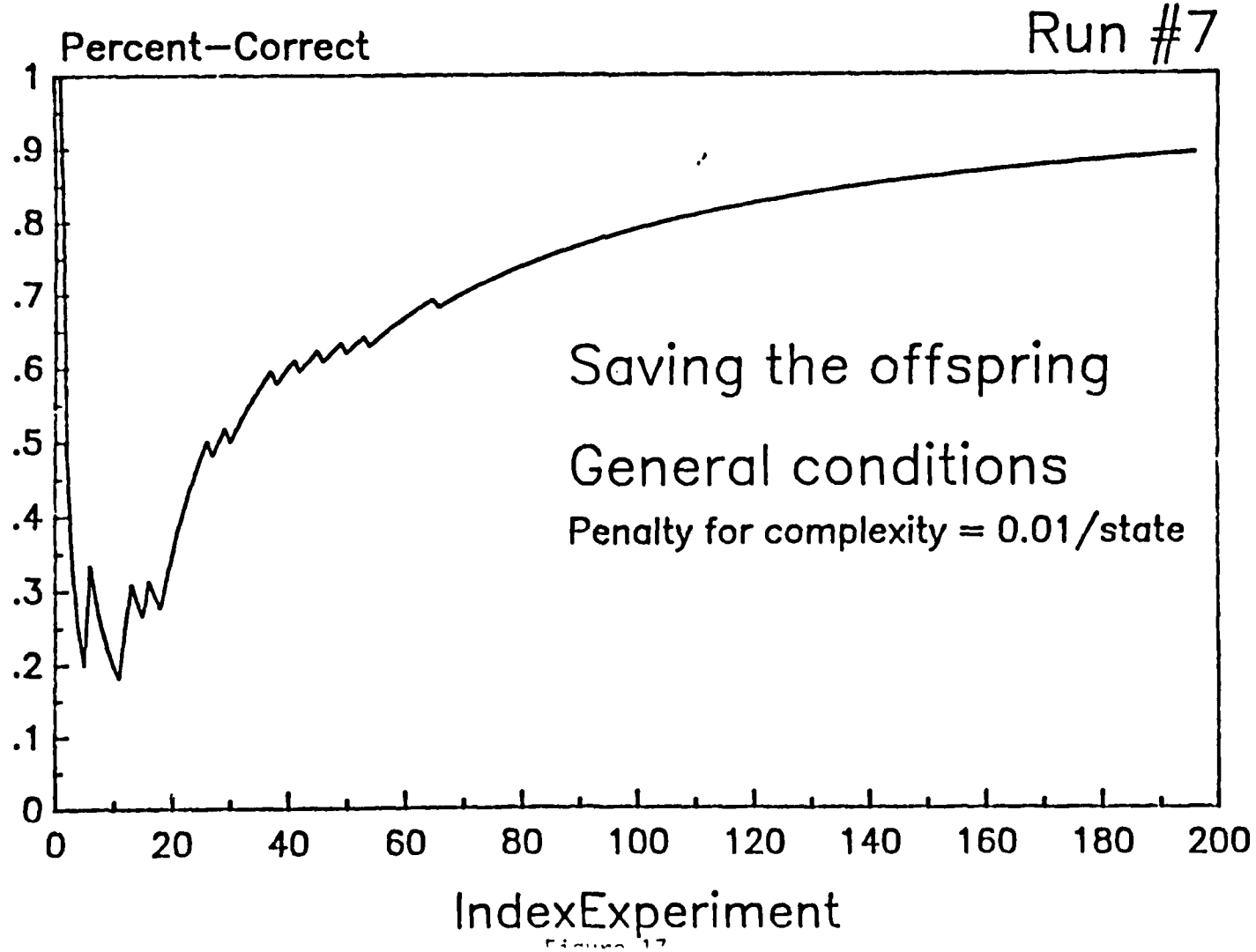


Figure 17

# 101110011101 cyclic environment

## With a penalty for complexity

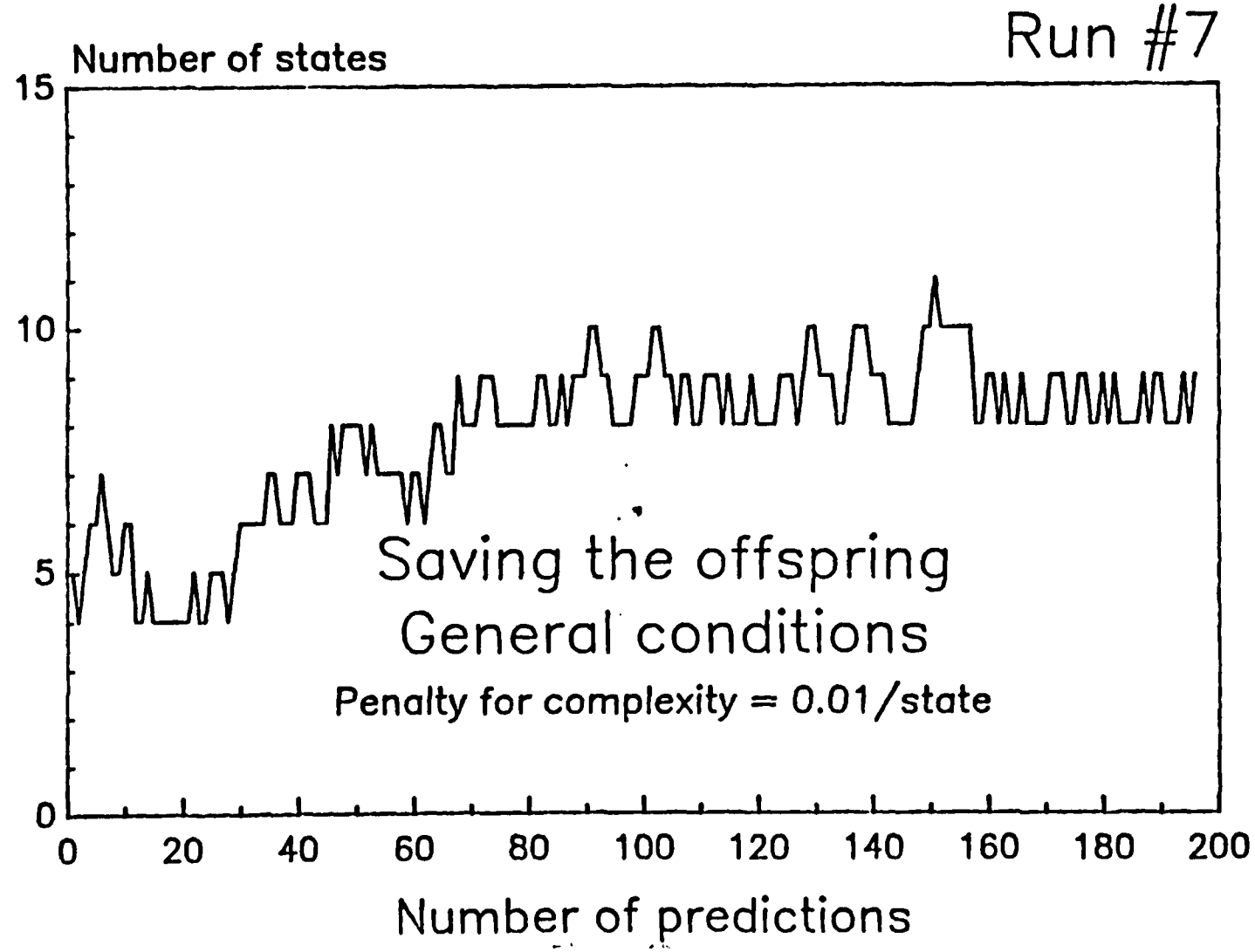


Figure 18
22

# 101110011101 cyclic environment

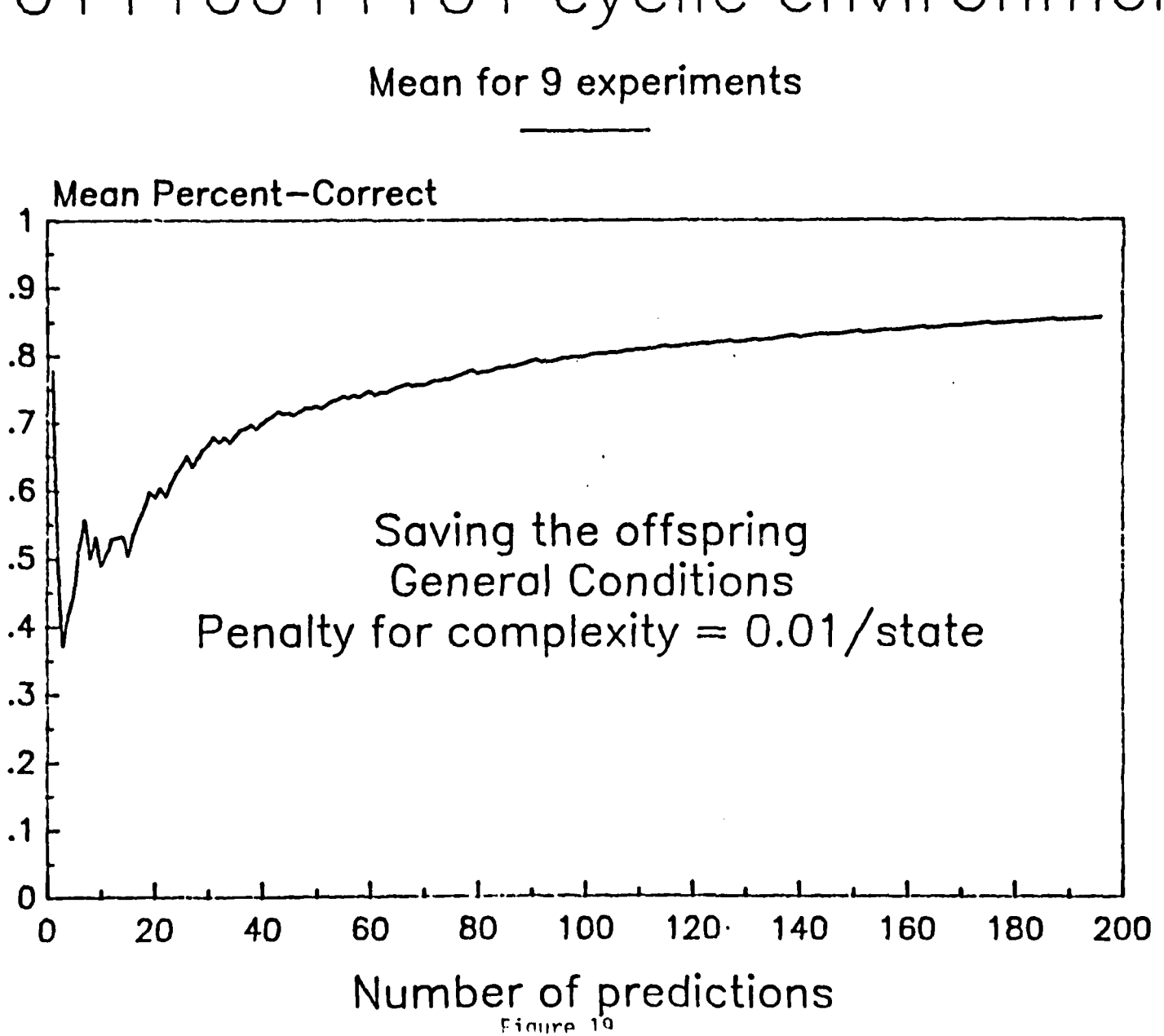


Figure 19

# 101110011101 cyclic environment

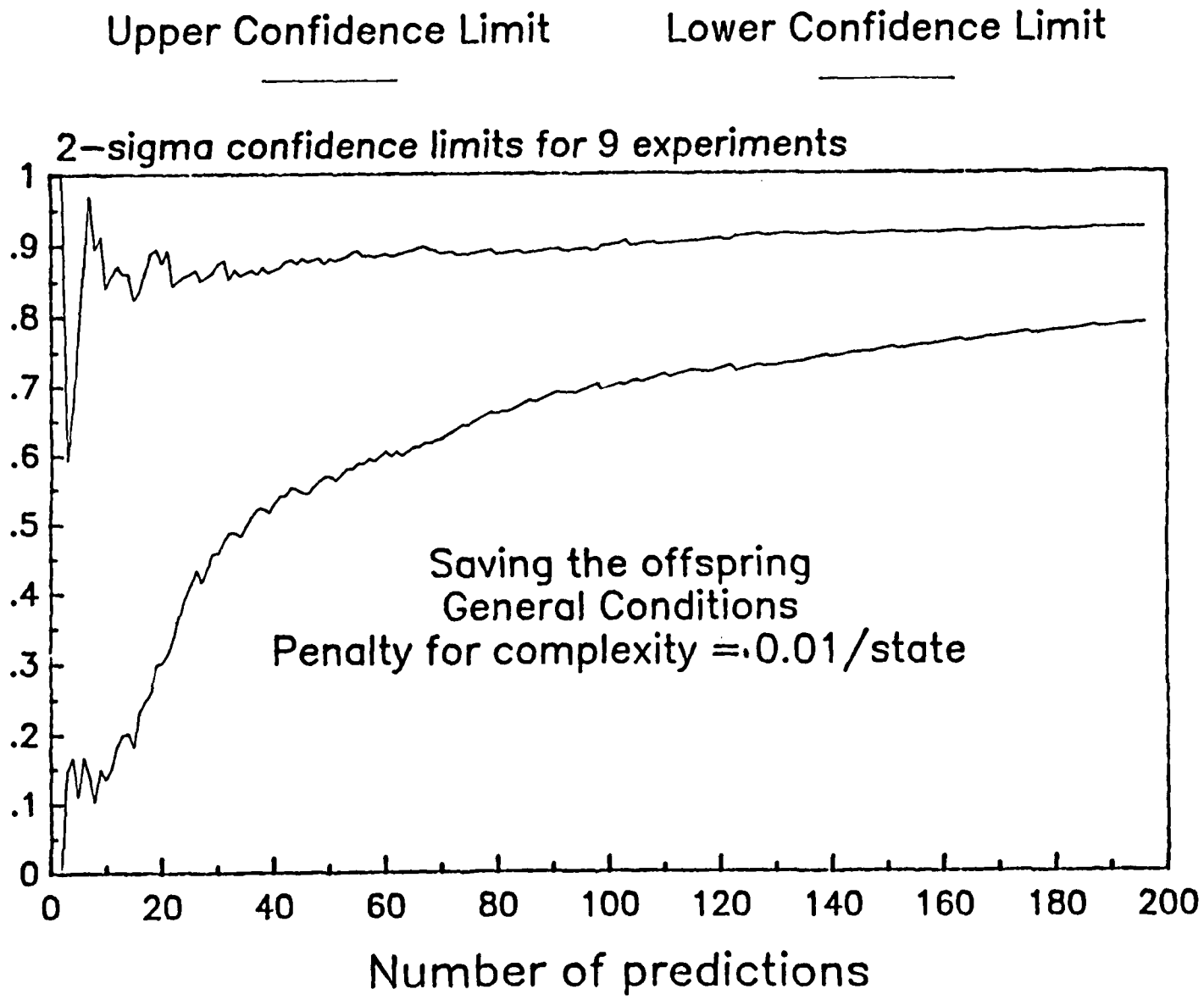


Figure 20

# Fibonacci mod 10 – Save offspring (50 state machines)

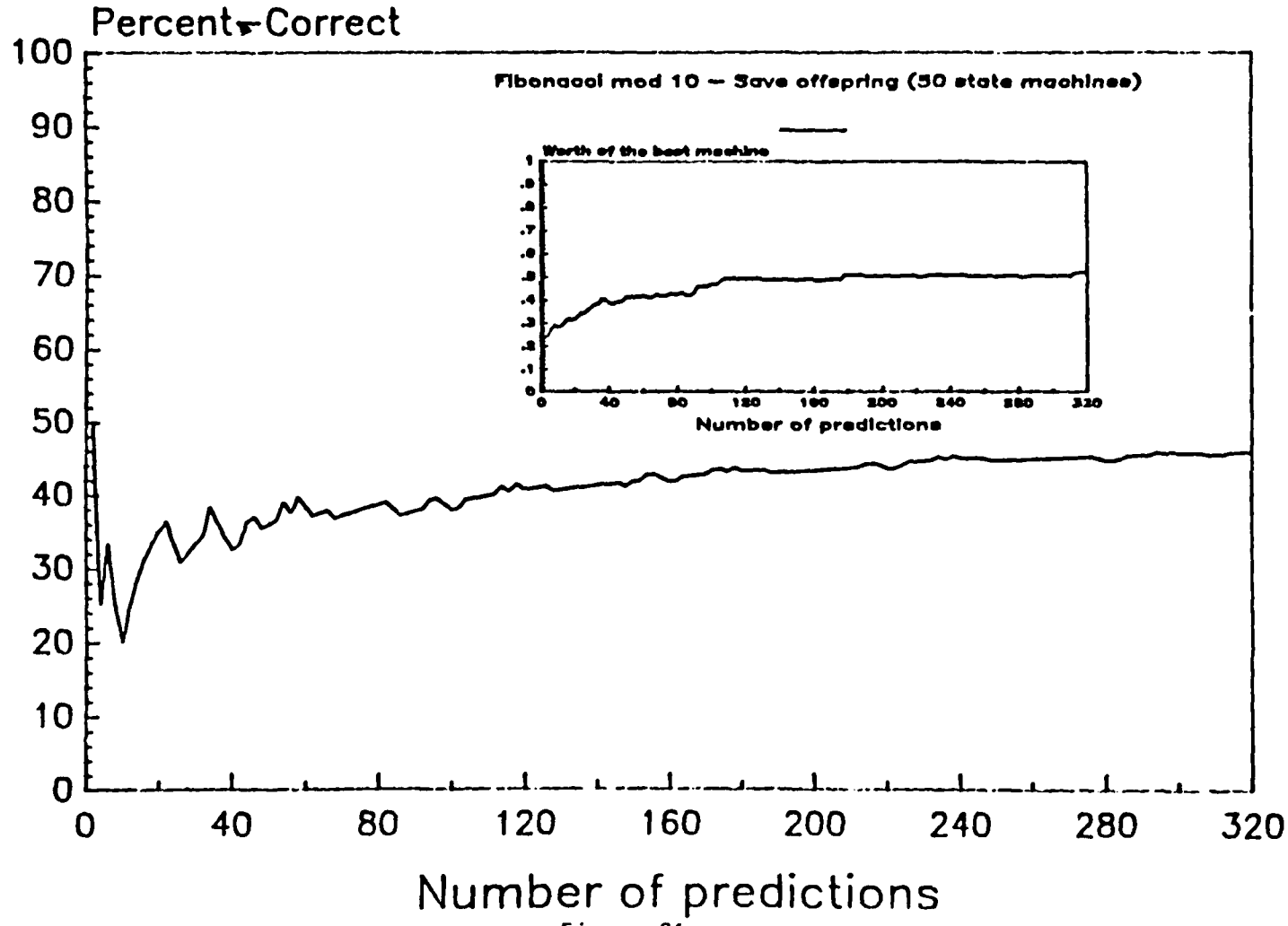


Figure 21

# Fibonacci mod 10 – Save offspring (50 state machines)

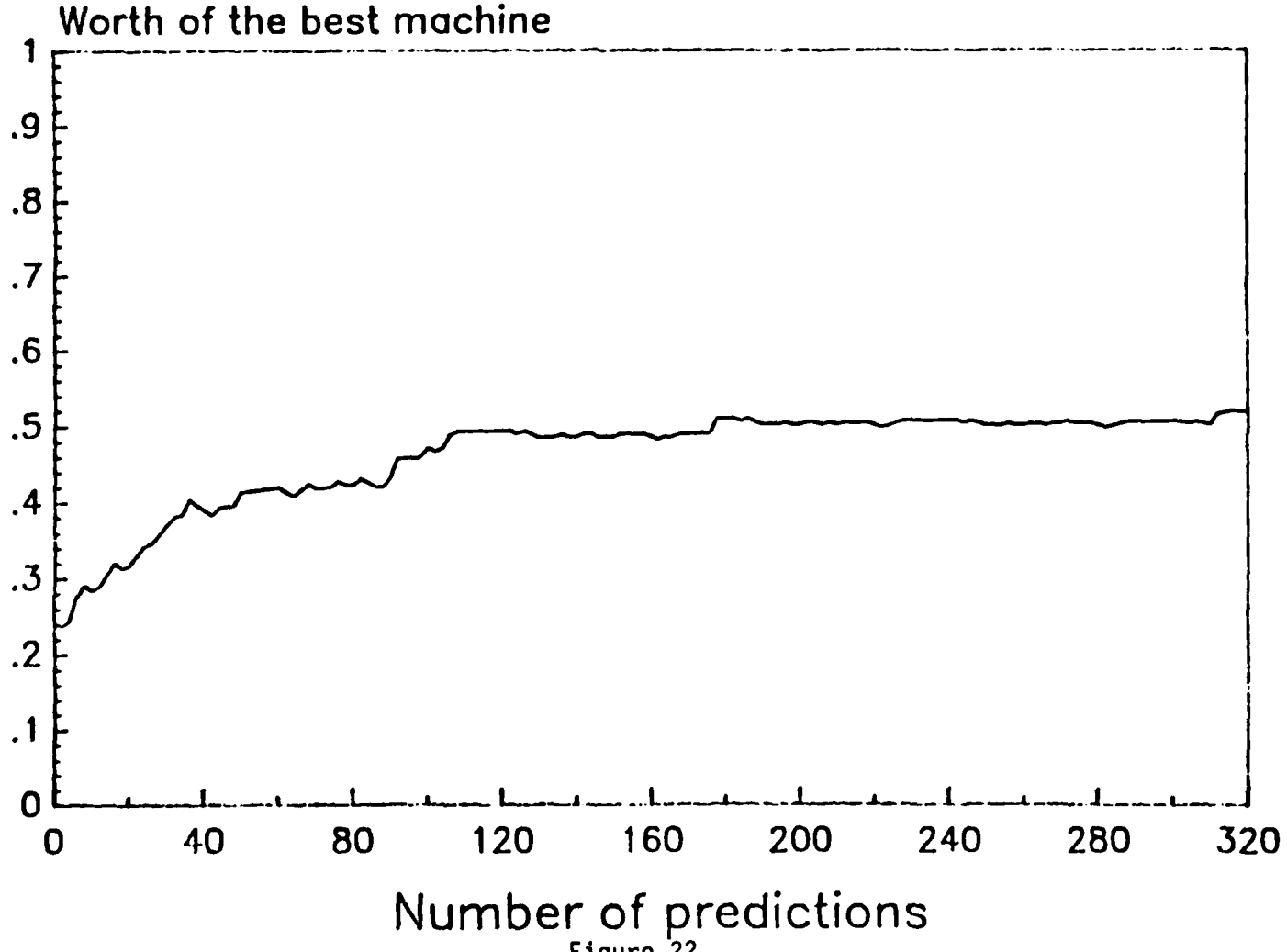


Figure 22

# Sequence of Internally Growing Zeroes

## Saving the offspring instead of parent

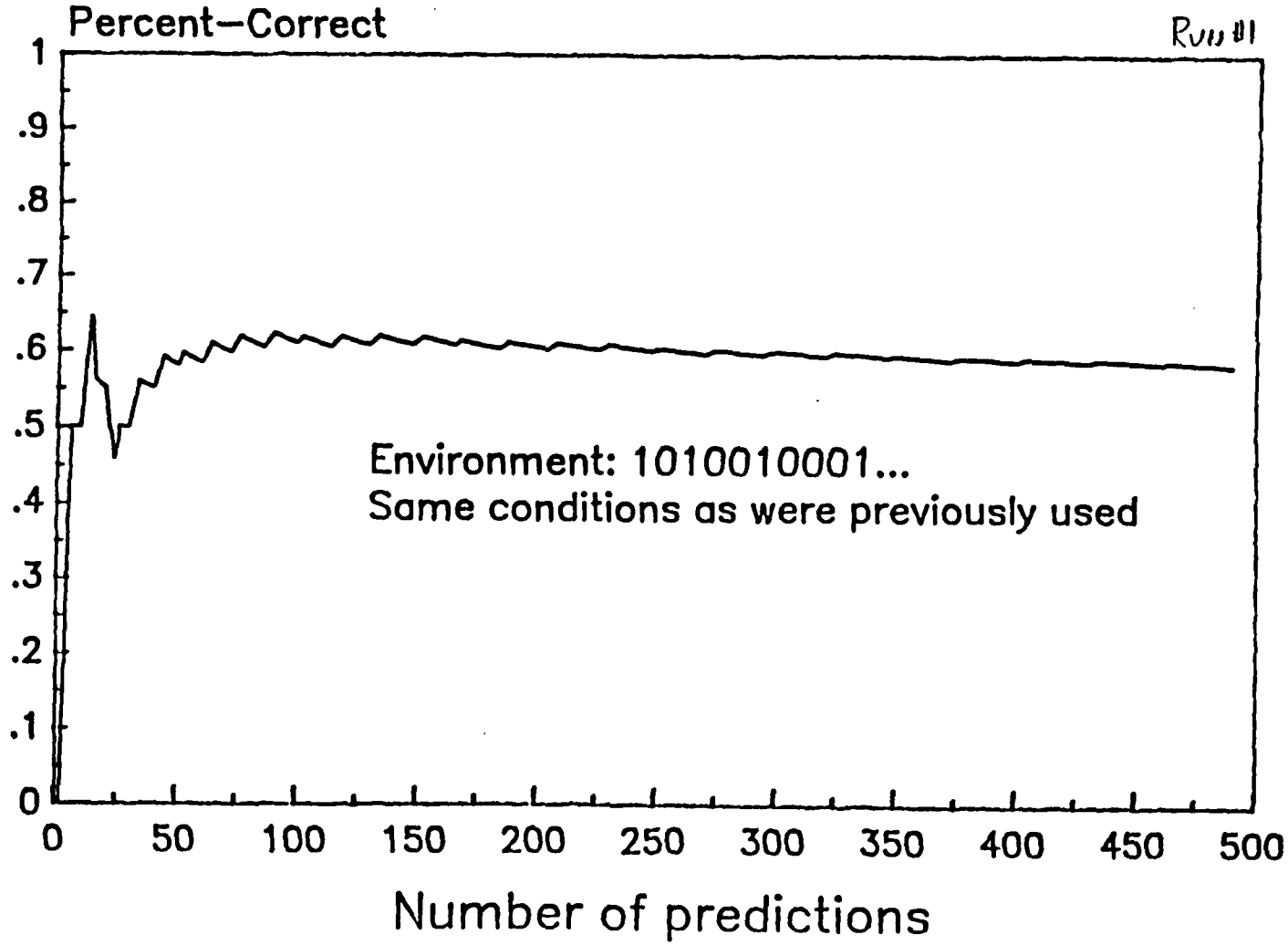


Figure 23

# Sequence of Internally Growing Zeroes

## Saving the offspring instead of parent

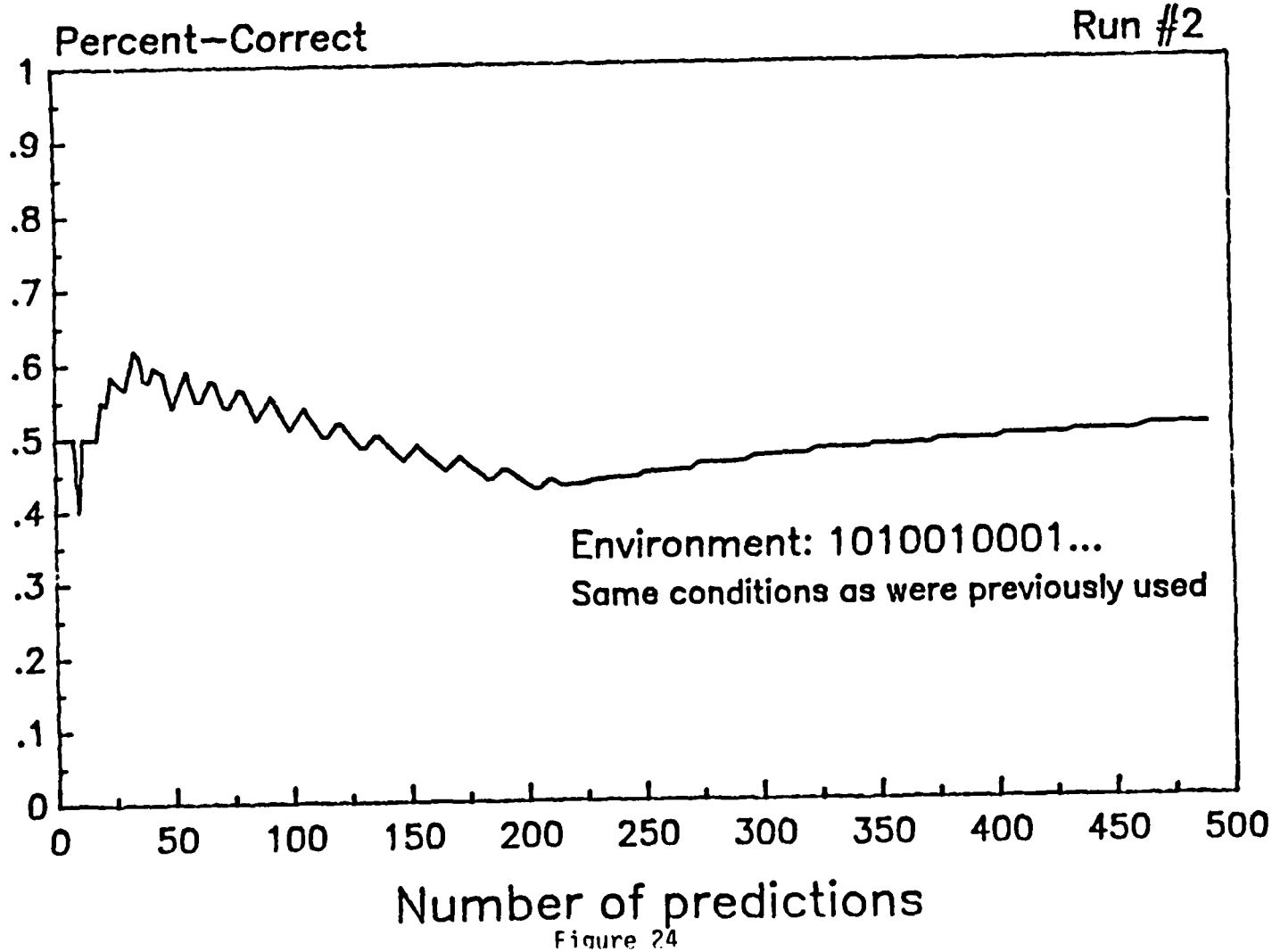


Figure 24

# Sequence of Internally Growing Zeroes

## Saving the best offspring instead of parent

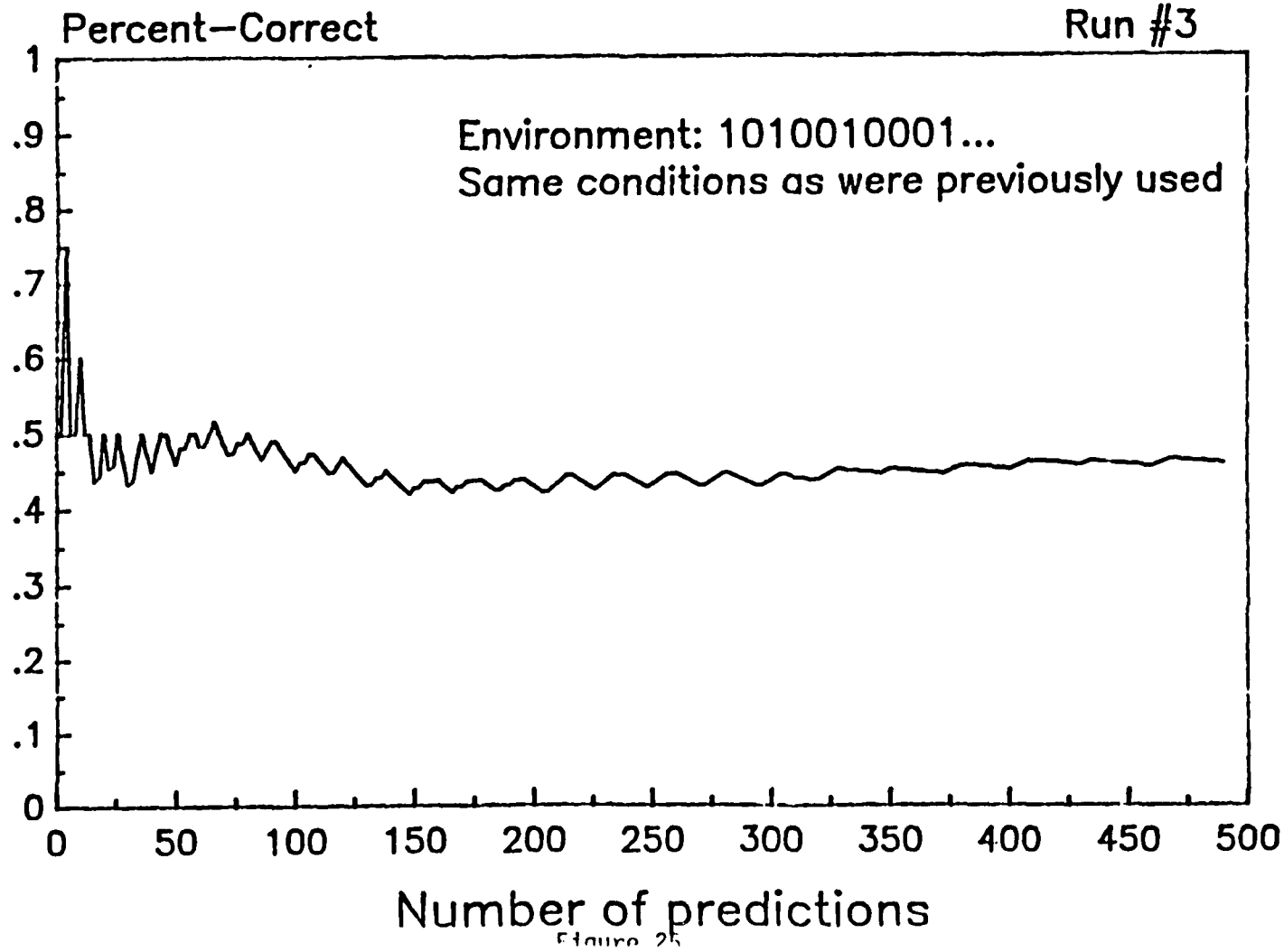


Figure 25



Figure 25

# Sequence of Internally Growing Zeroes

## Saving the best offspring instead of parent

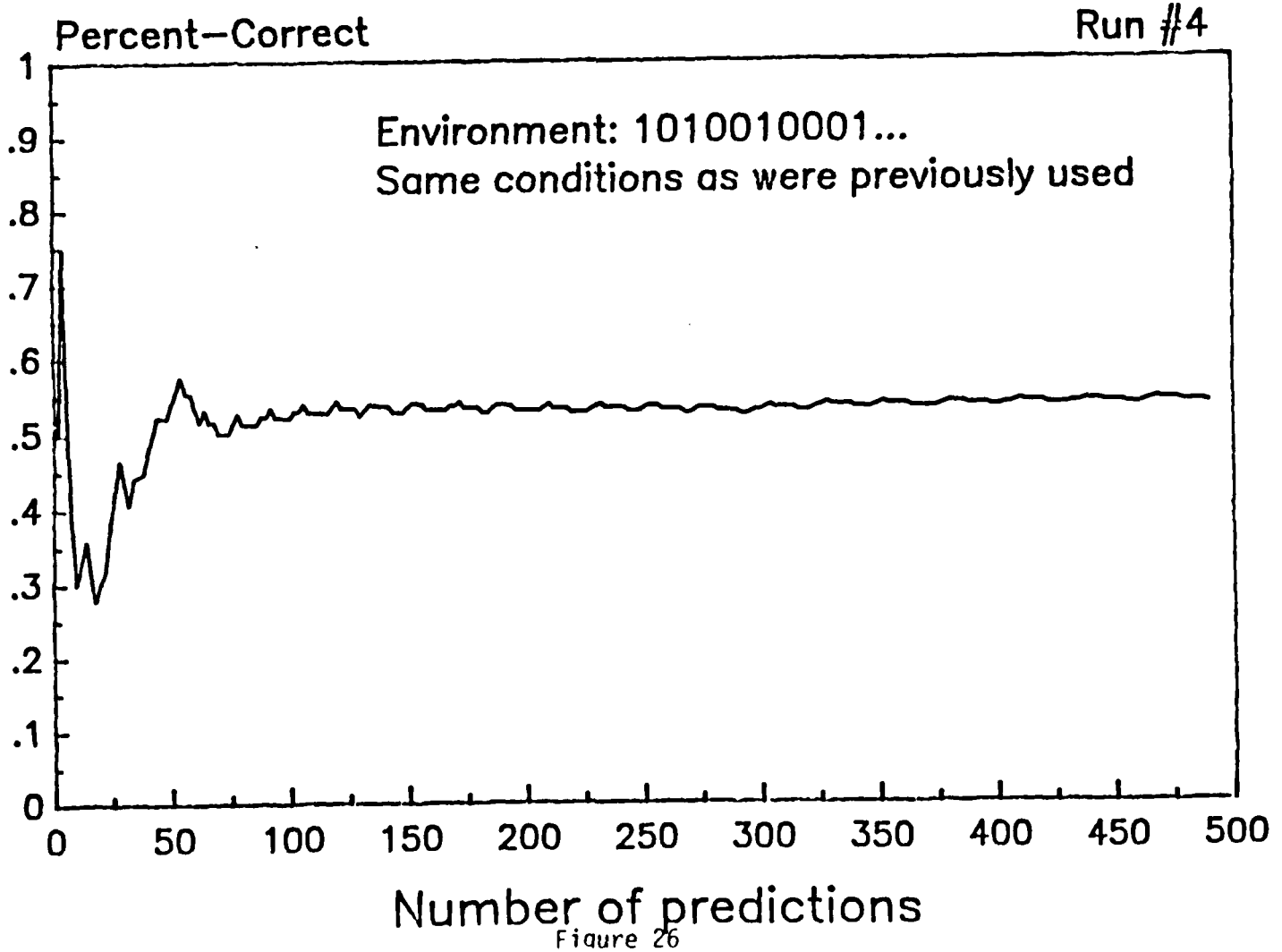


Figure 26

# Sequence of Internally Growing Zeroes

Saving the best offspring instead of parent

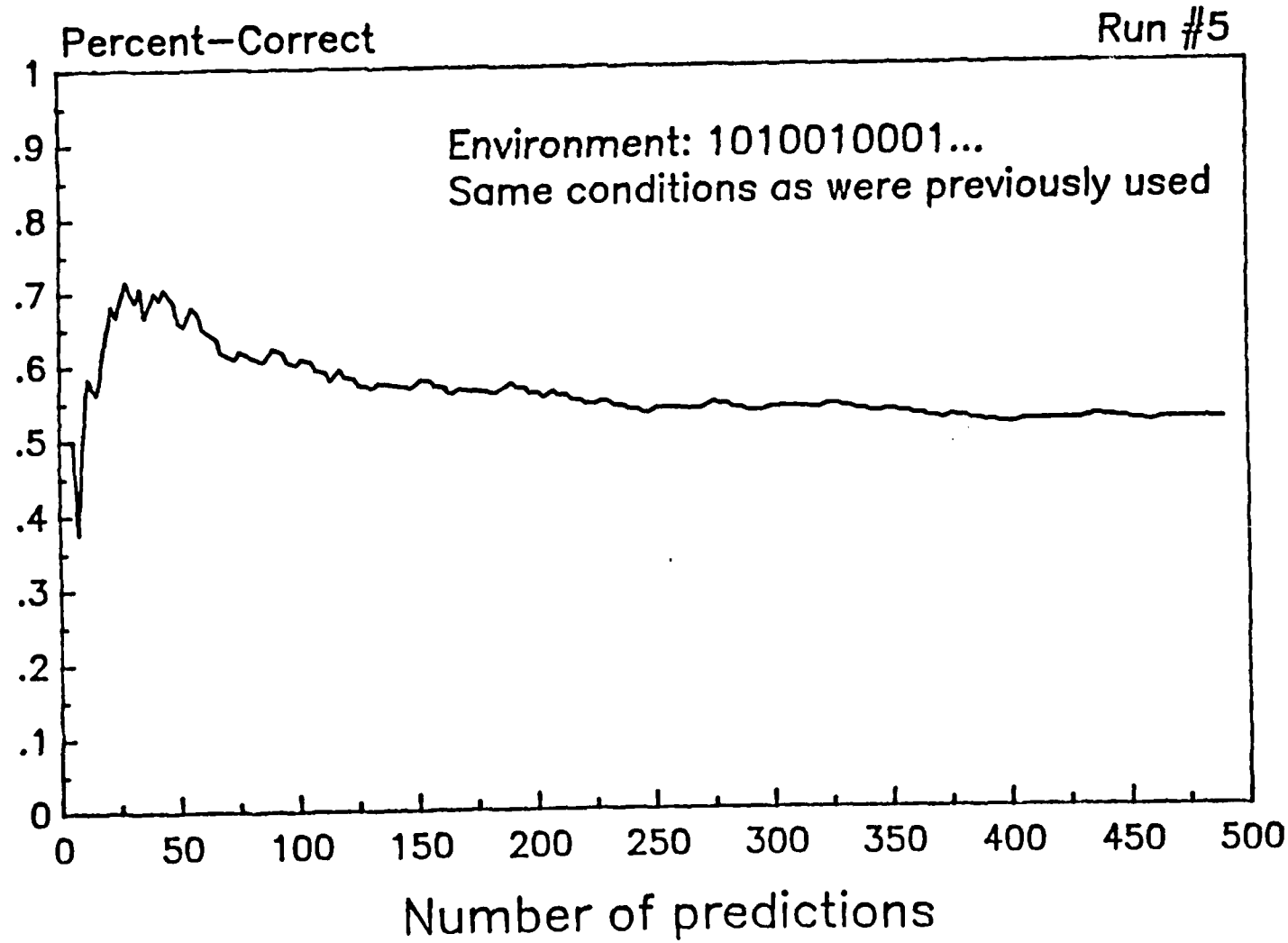


Figure 27



Figure 27

# Sequence of Internally Growing Zeroes

## Saving the offspring instead of parent

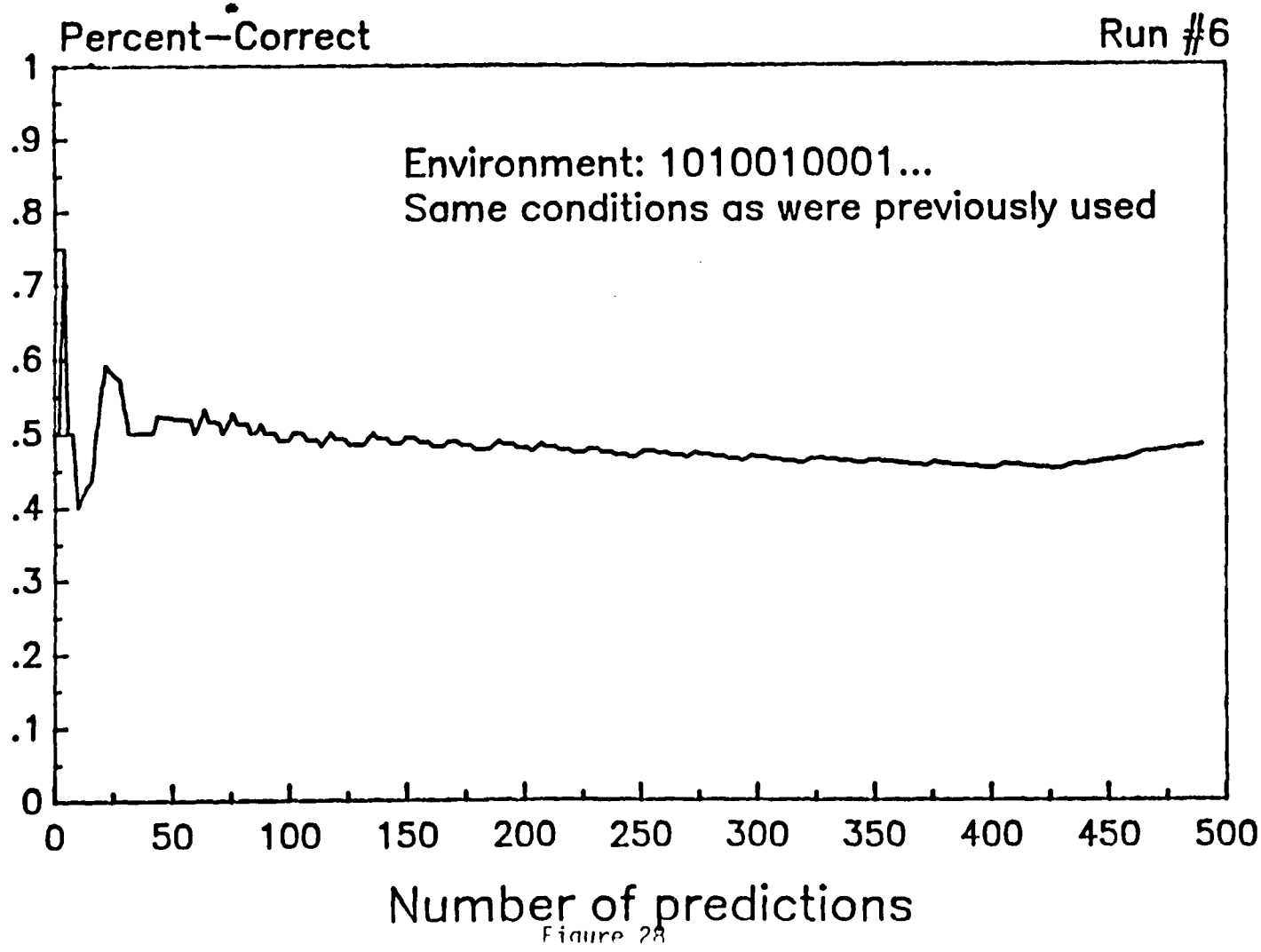


Figure 28

# Sequence of Internally Growing Zeroes

## Saving the offspring instead of parent

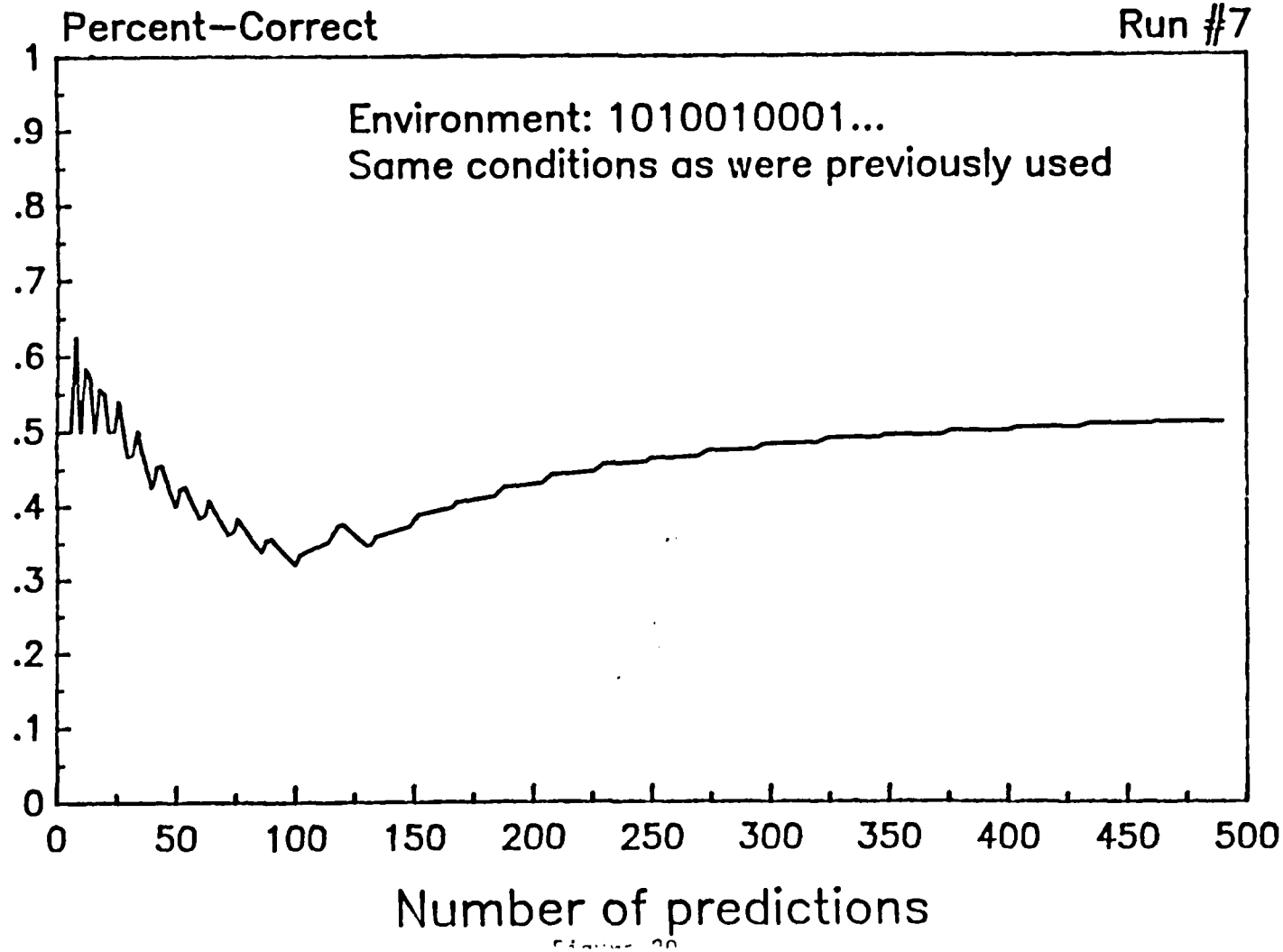


Figure 29

improvement after 220 predictions. Clearly, the evolving finite state machines now primarily predict the zeros. Figures 30 through 36 show these seven experiments in terms of the size of the machines. In every case, there was a rapid build-up to the imposed limit. Conceivably, a higher limit would have allowed for a greater predictive fit score; however, in the limit, the results would be the same. As the machines increase in size, each state would correspond to a given symbol in the environment. Eventually, there would be a steady state that predicts only zeros. Figures 37 and 38 show the mean and two sigma limits for these seven experiments.

## Altered Mutation Variation

Ten experiments were performed to examine the worth of altering the probabilities of mutation in order to improve the effectiveness of the evolutionary process. Here the same binary cyclic environment was used with a thirty percent chance of adding a state and ten percent chance of deleting a state. Figures 39 through 44 indicate the cumulative percent correct. In these ten experiments, the evolutionary process generated five perfect predictors. As stated in the previous Progress Report, no perfect predictors were found in thirty such experiments. Figure 45 indicates the manner in which the states bulid up in these experiments. Figures 46 and 47 indicate the mean cumulative percent correct scores and two sigma limits, respectively. The average number of evaluated offspring was 3,238.8 with only a slight variance.

## Predicting Cycles within Cycles

A "double obverse" environment was constructed consisting of ten repetitions of the above binary cycle followed by ten repetitions of 223445, this being followed by another ten cycles of the original binary sequence. Offspring of equal worth were saved. Five experiments were conducted demonstrating early learning. The prediction score then

# Sequence of Internally Growing Zeroes

## Saving the offspring instead of parent

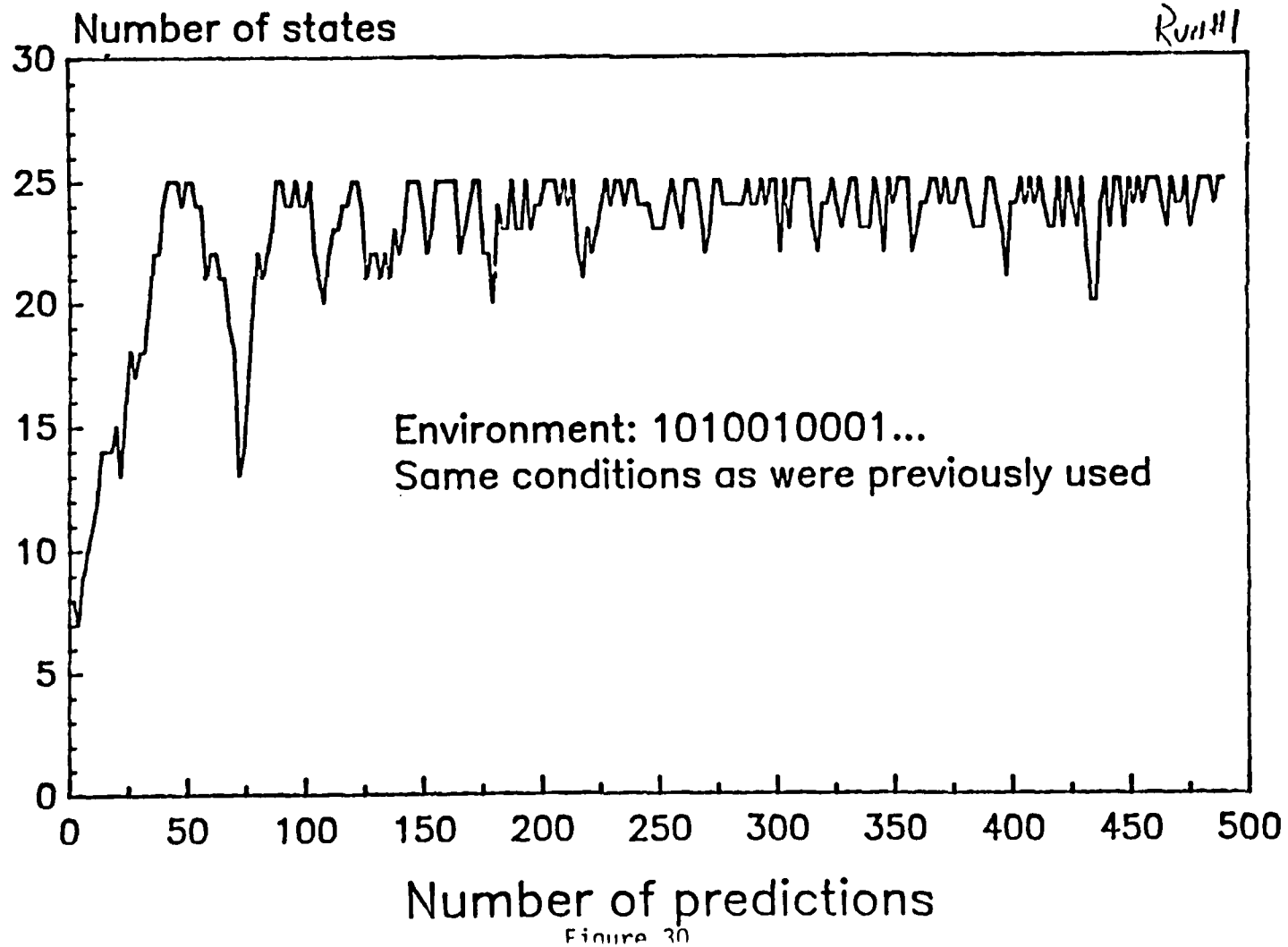


Figure 30

# Sequence of Internally Growing Zeroes

## Saving the best offspring instead of parent

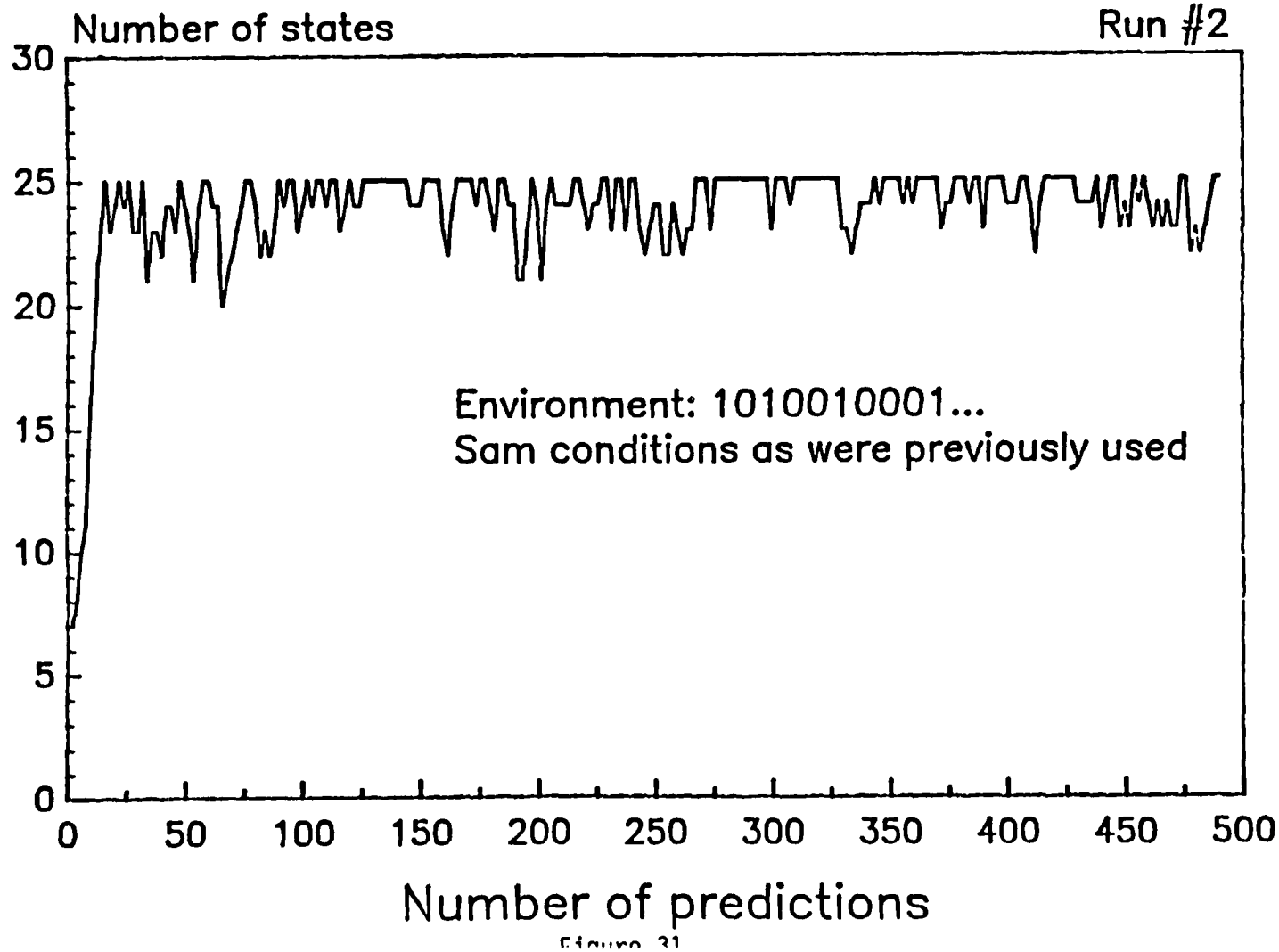


Figure 31

# Sequence of Internally Growing Zeroes

## Saving the best offspring instead of parent

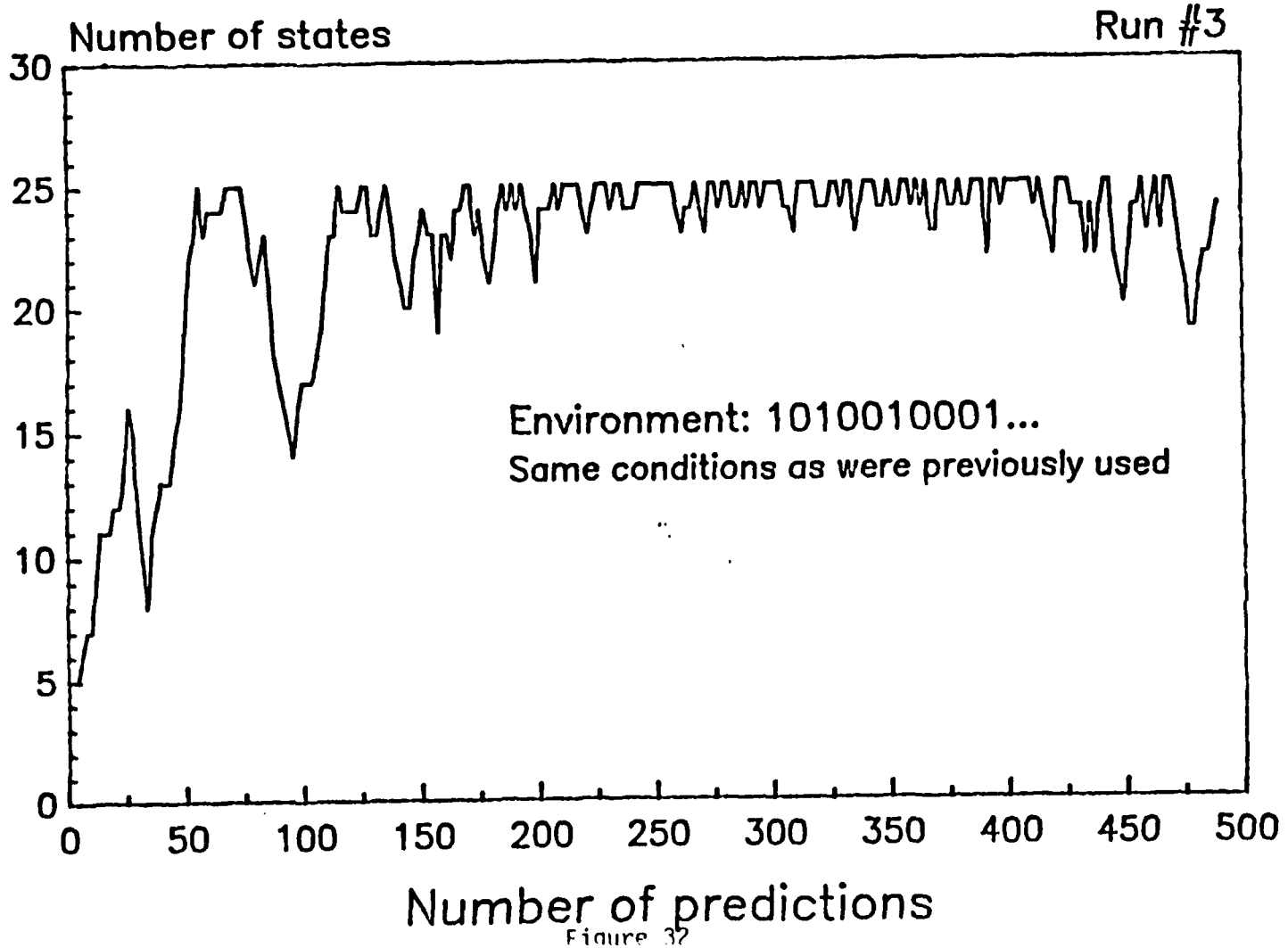


Figure 32

# Sequence of Internally Growing Zeroes

Saving the best offspring instead of parent

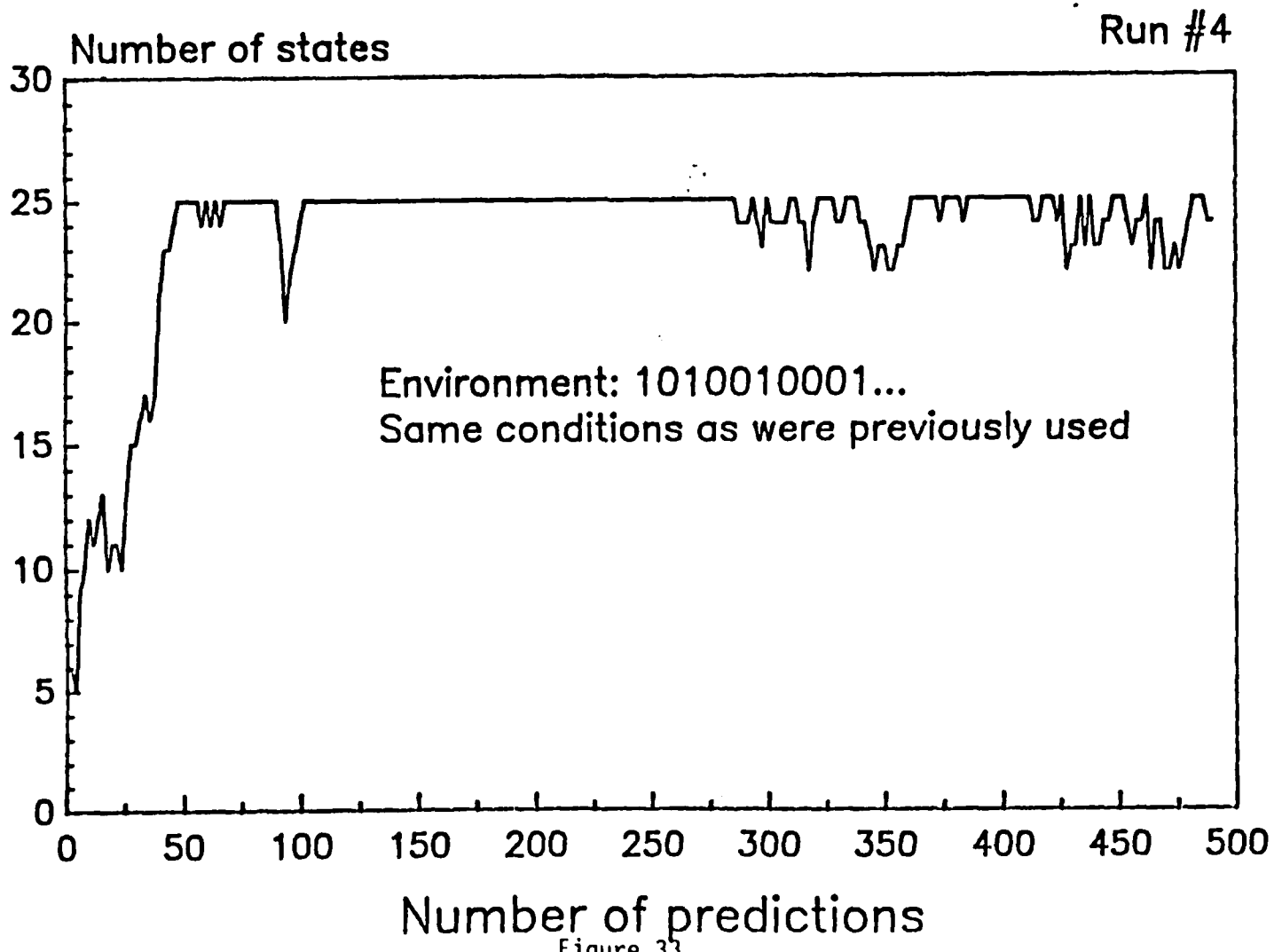


Figure 33

# Sequence of Internally Growing Zeroes

## Saving the best offspring instead of parent

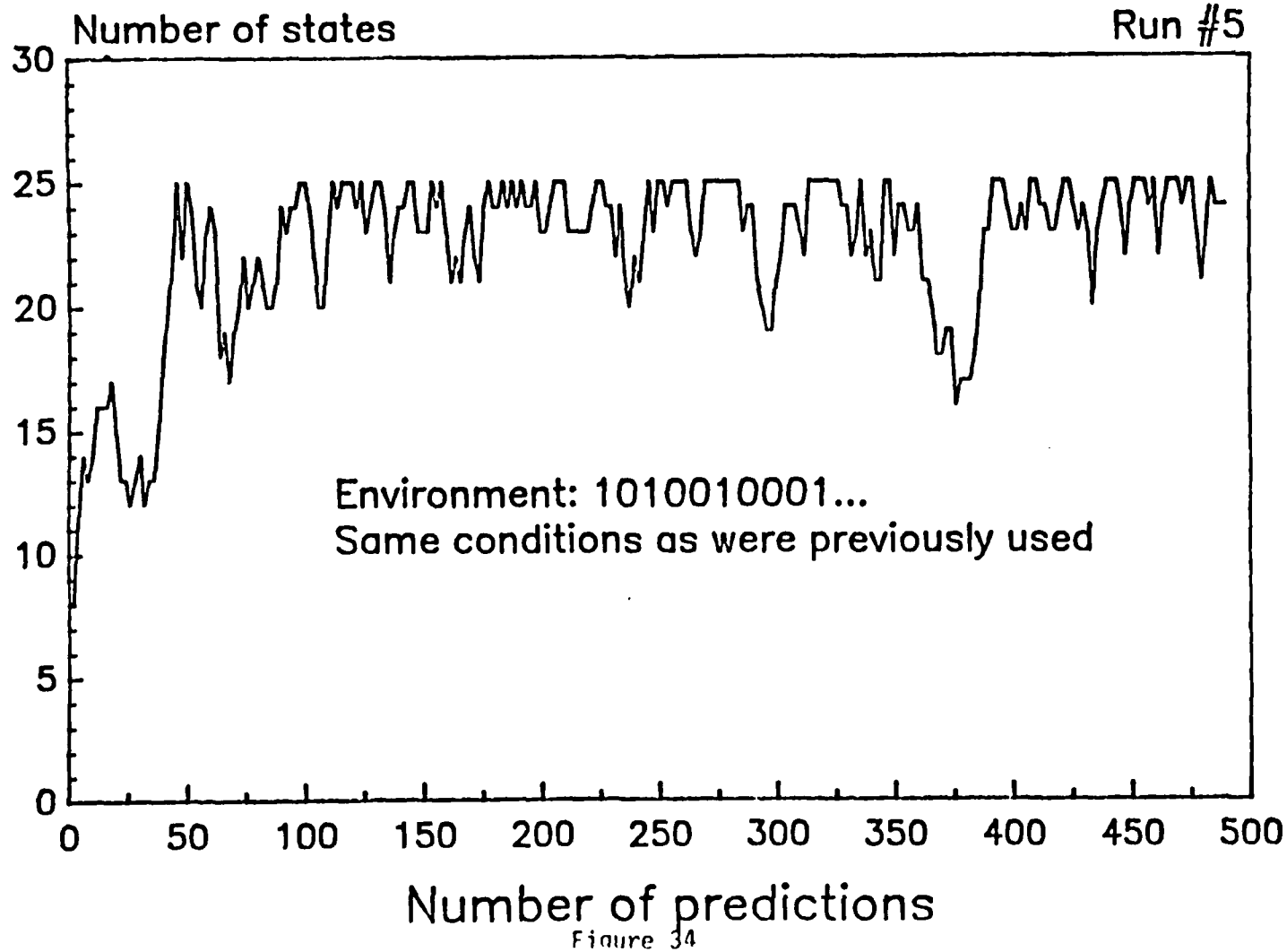


Figure 34

# Sequence of Internally Growing Zeroes

## Saving the offspring instead of parent

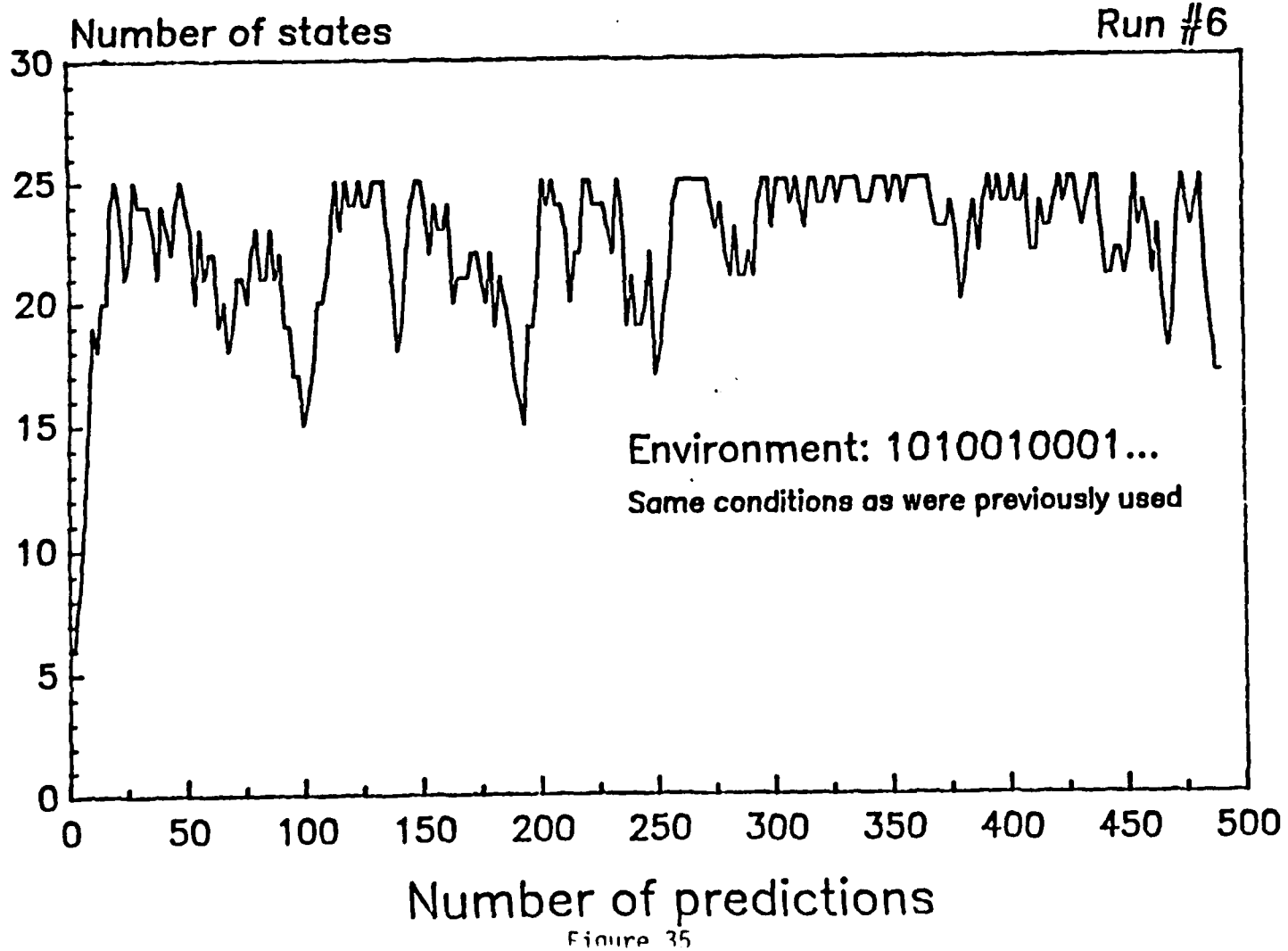


Figure 35

# Sequence of Internally Growing Zeroes

## Saving the offspring instead of parent

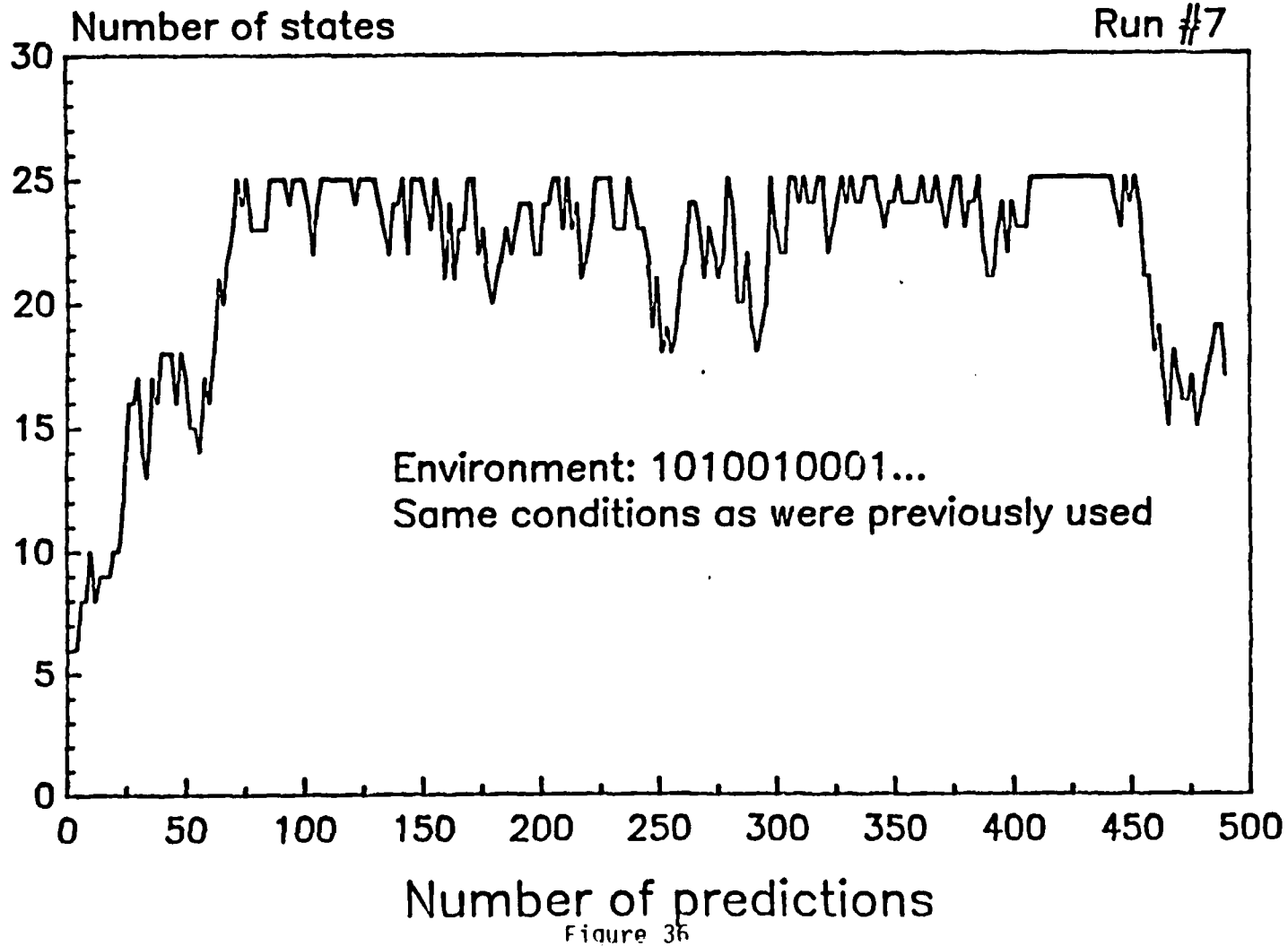


Figure 36



Figure 36

# Sequence of Internally Growing Zeroes

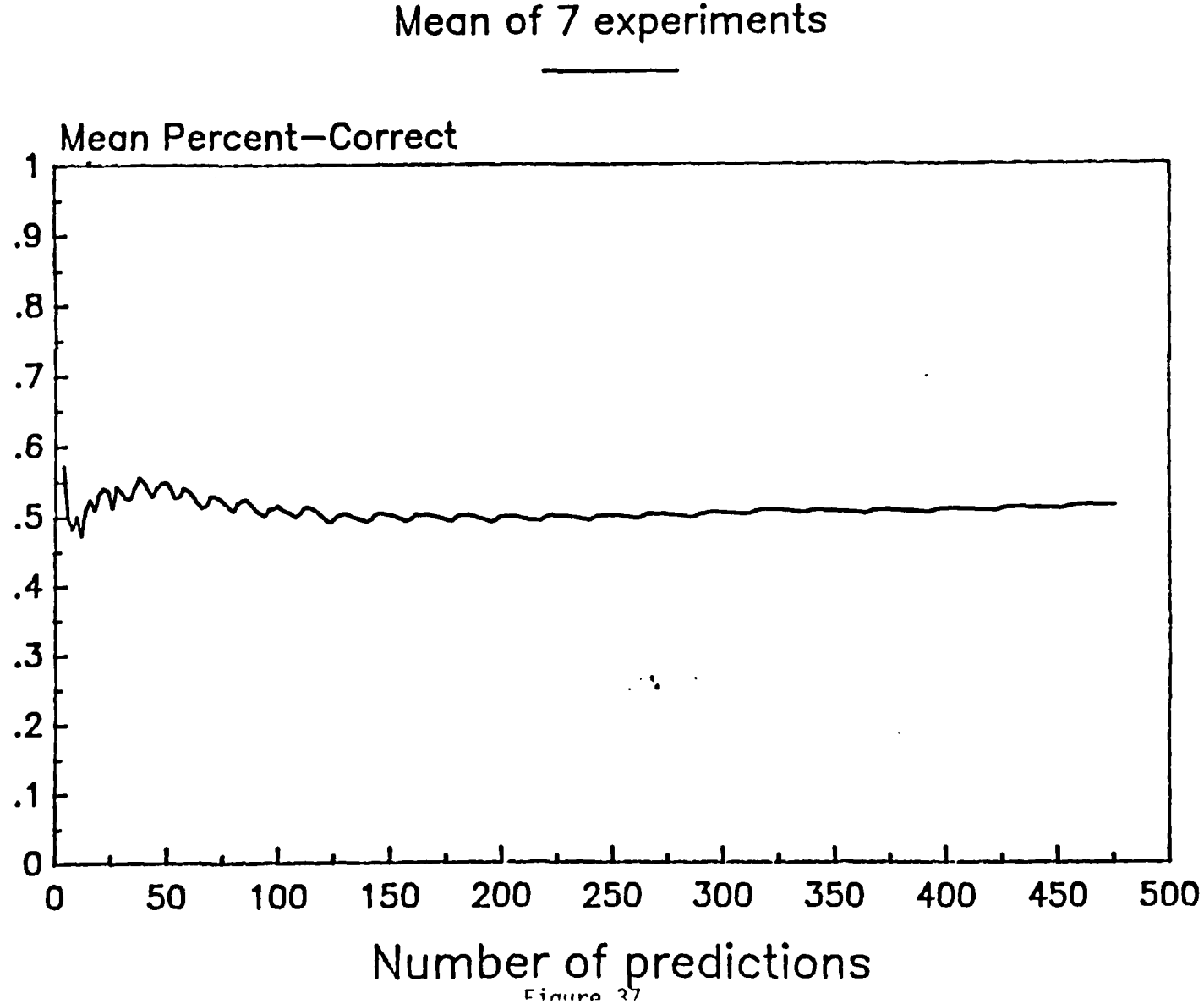


Figure 37



Figure 37

# Sequence of Internally Growing Zeroes

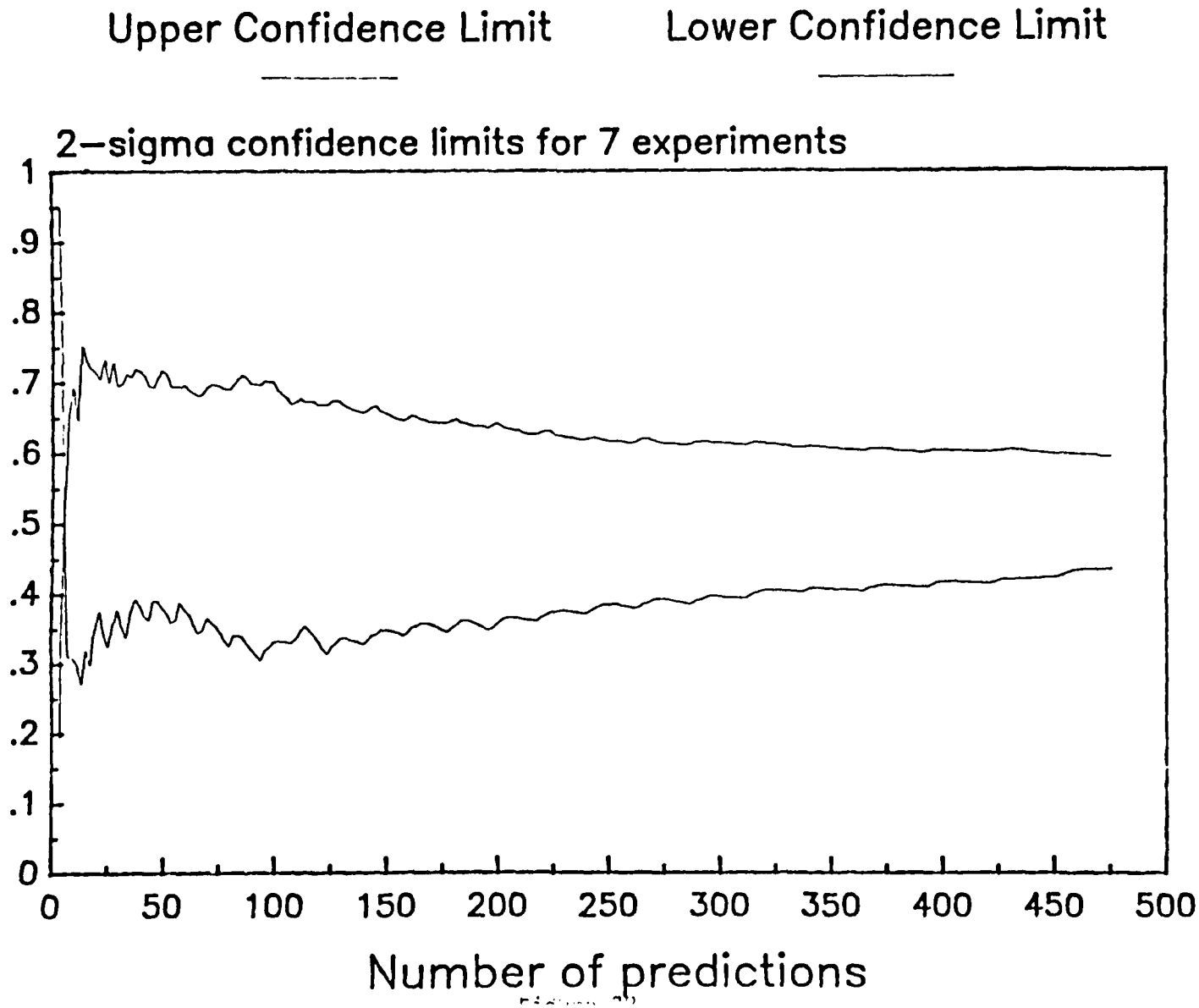


Figure 38
43

Cyclic environment with non–equal mutation variation

(101110011101)

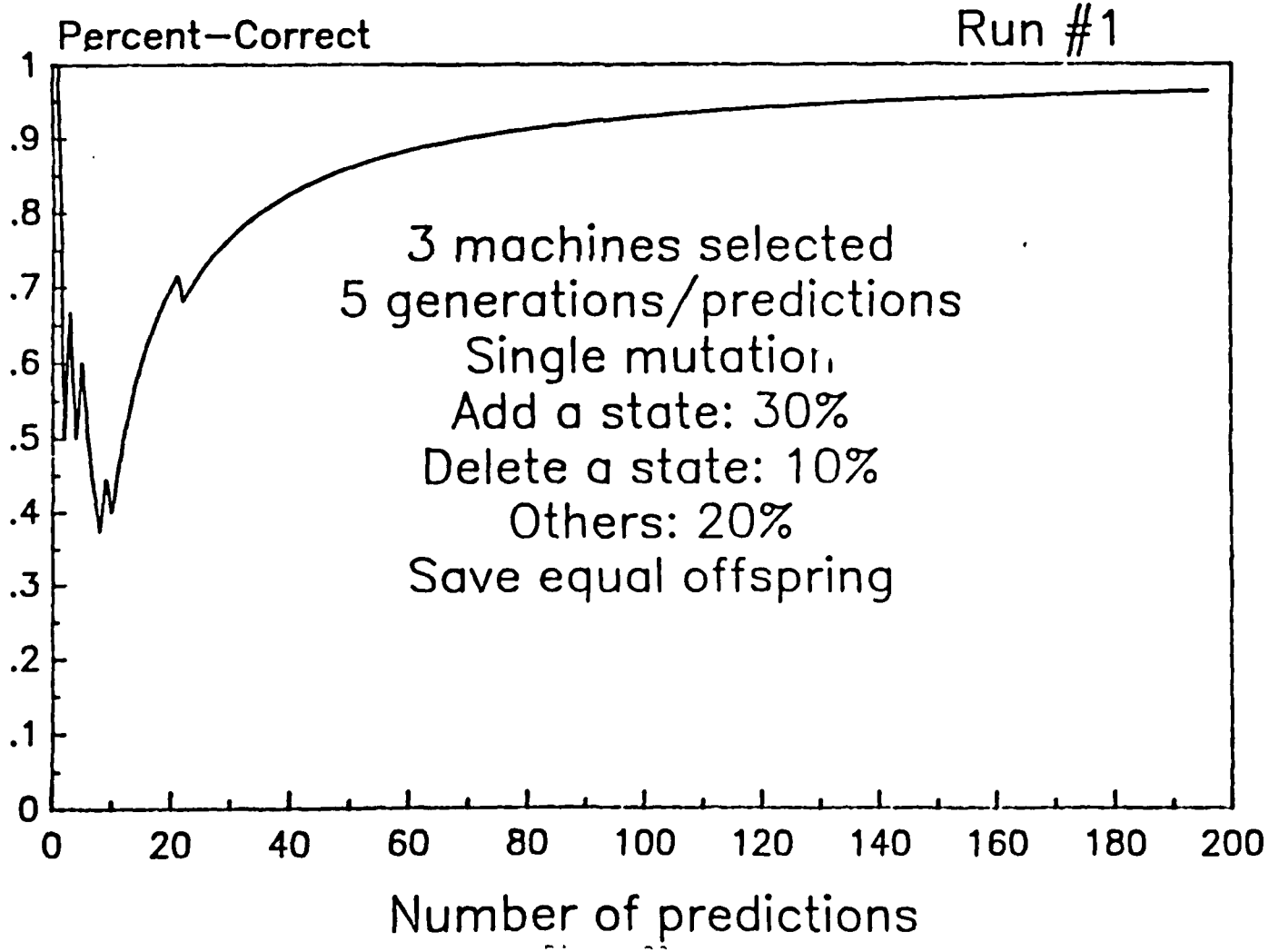


Figure 39

# Cyclic environment with non–equal mutation variation (101110011101)

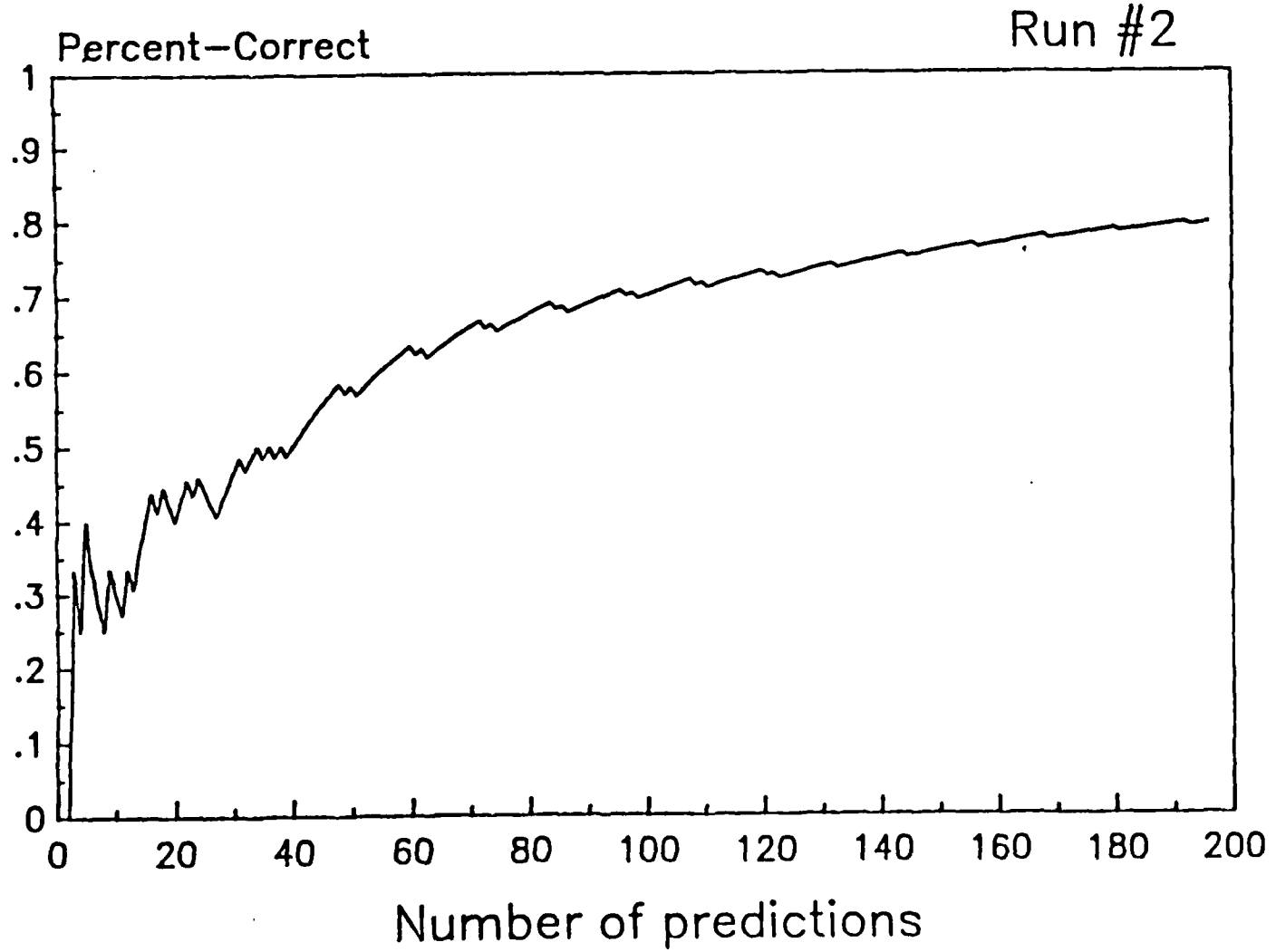


Figure 40

# Cyclic environment with non–equal mutation variation (101110011101)

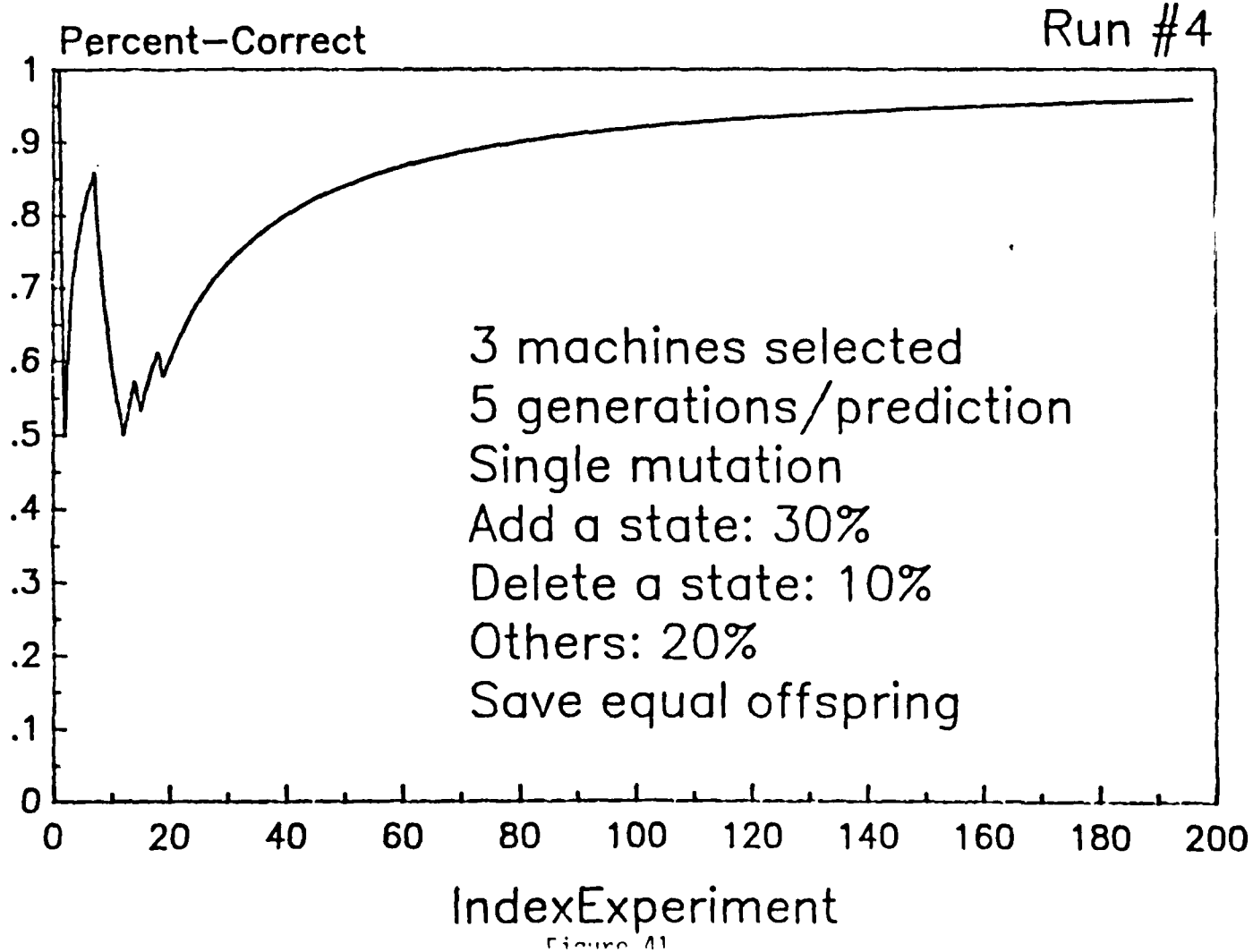


Figure 41

# Cyclic environment with non−equal mutation variation (101110011101)

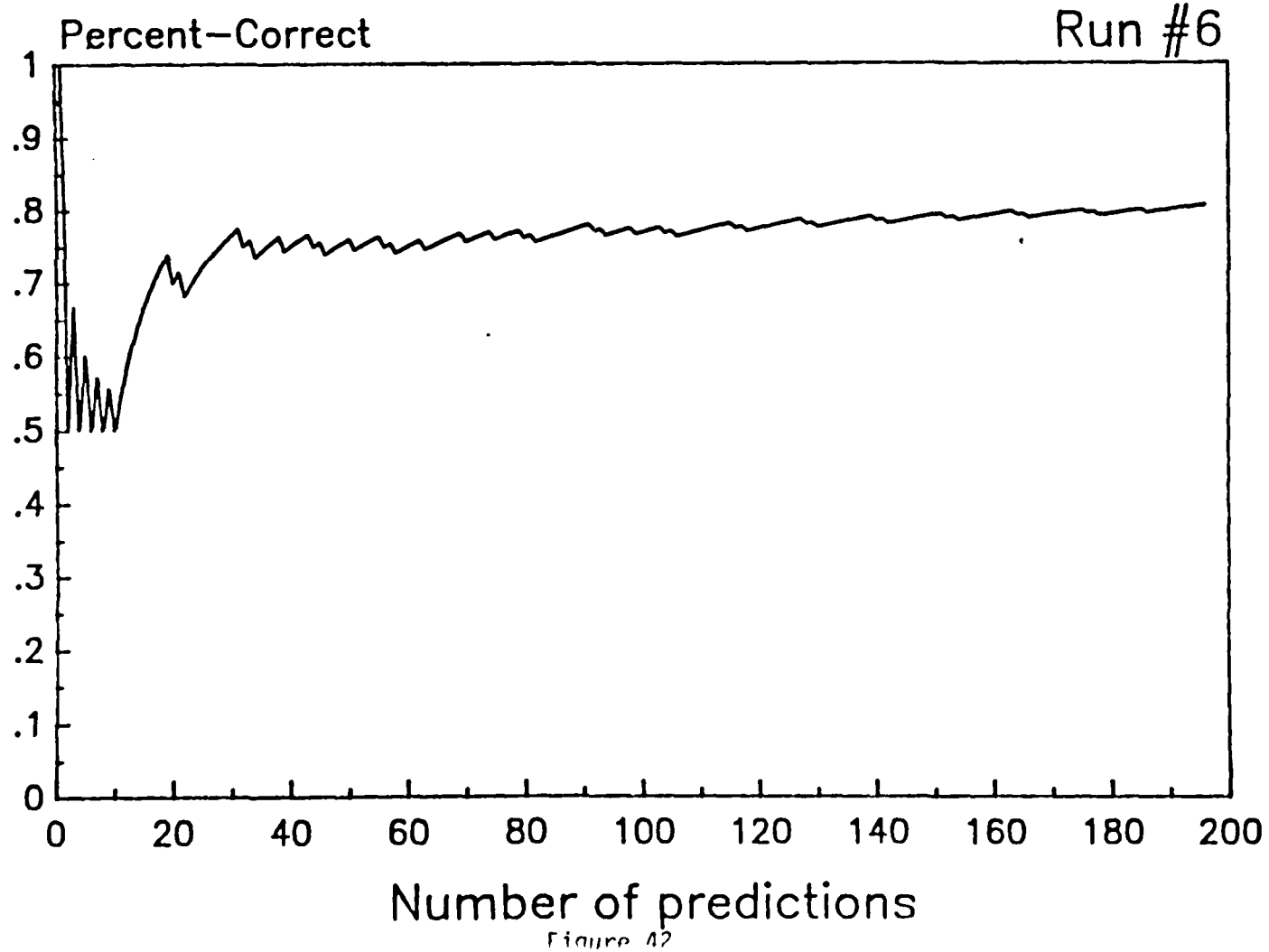


Figure 42



Figure 42

# Cyclic environment with non–equal mutation variation (101110011101)

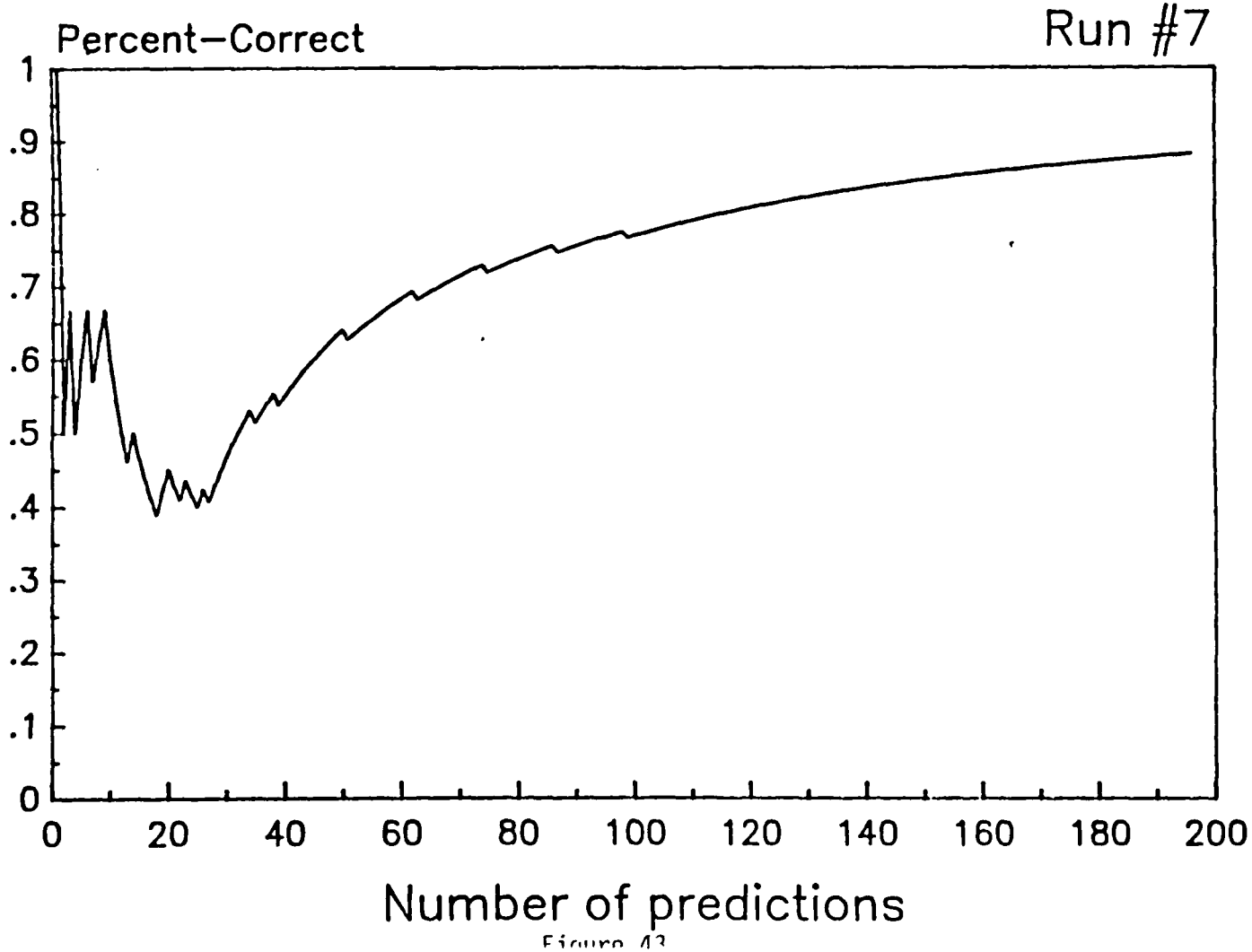


Figure 43

# Cyclic environment with non–equal mutation variation (101110011101)

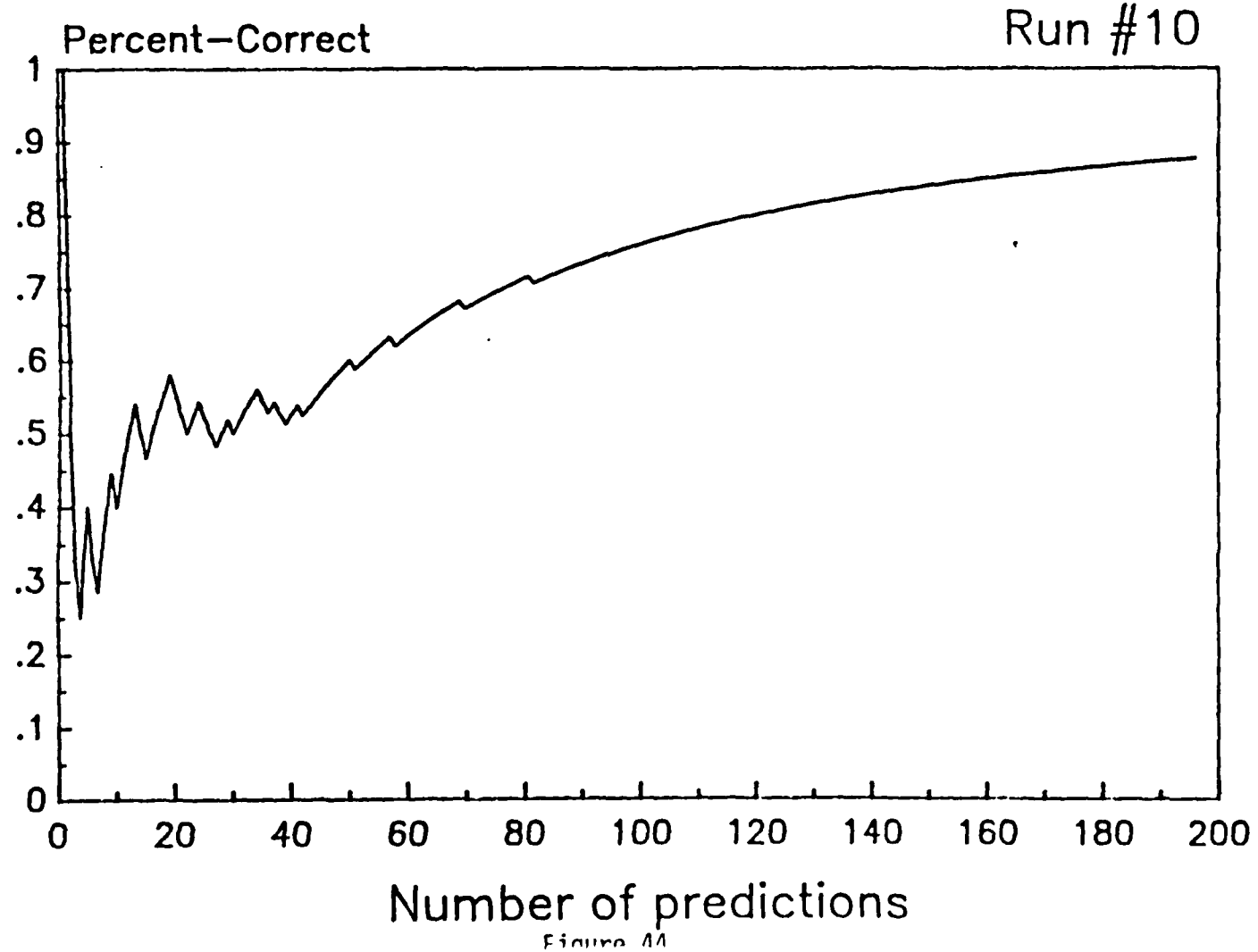


Figure 44

# Cyclic environment with non−equal mutation variation (101110011101)

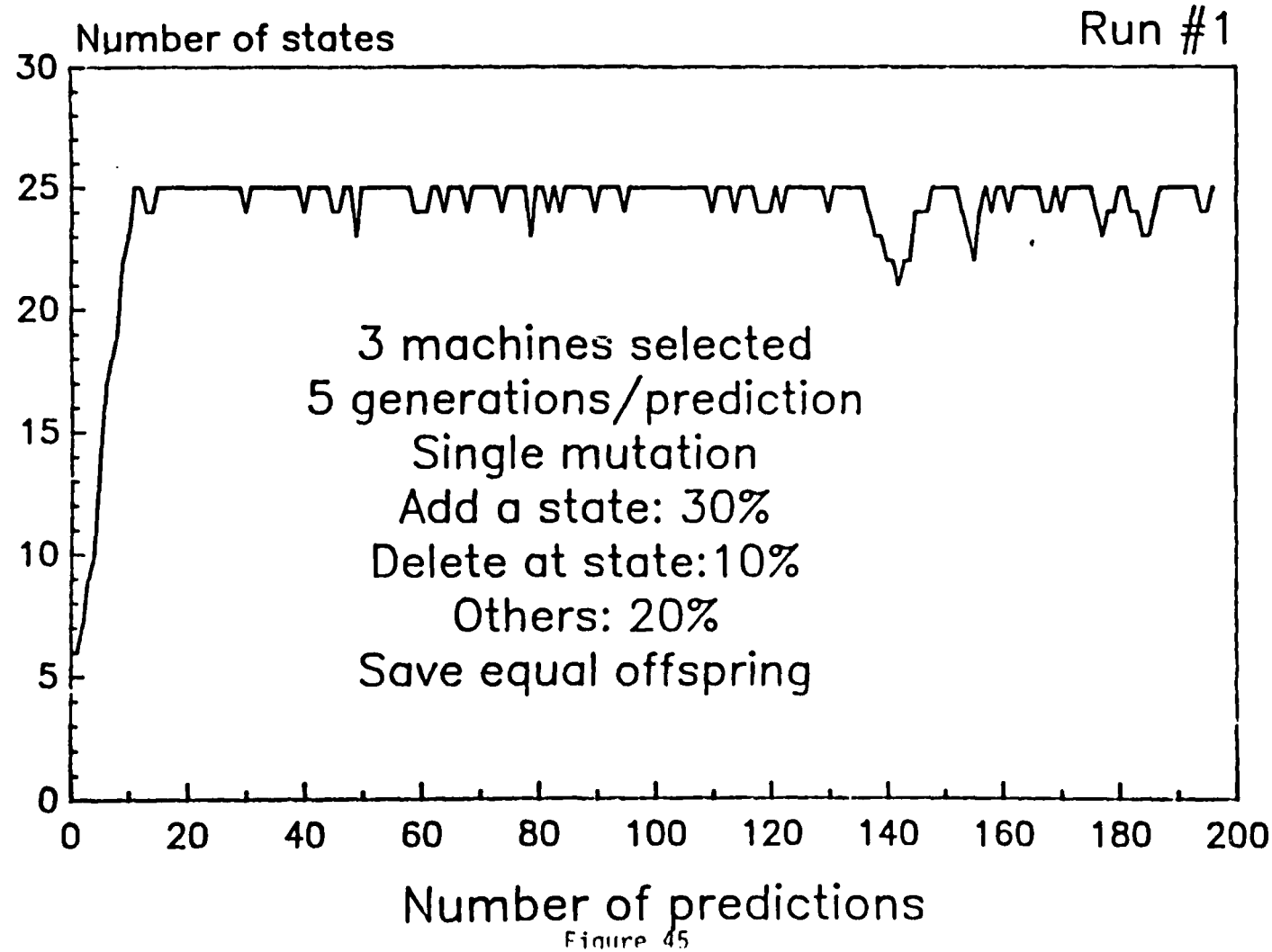


Figure 45



Figure 45

# Cyclic environment with unequal mutation variation

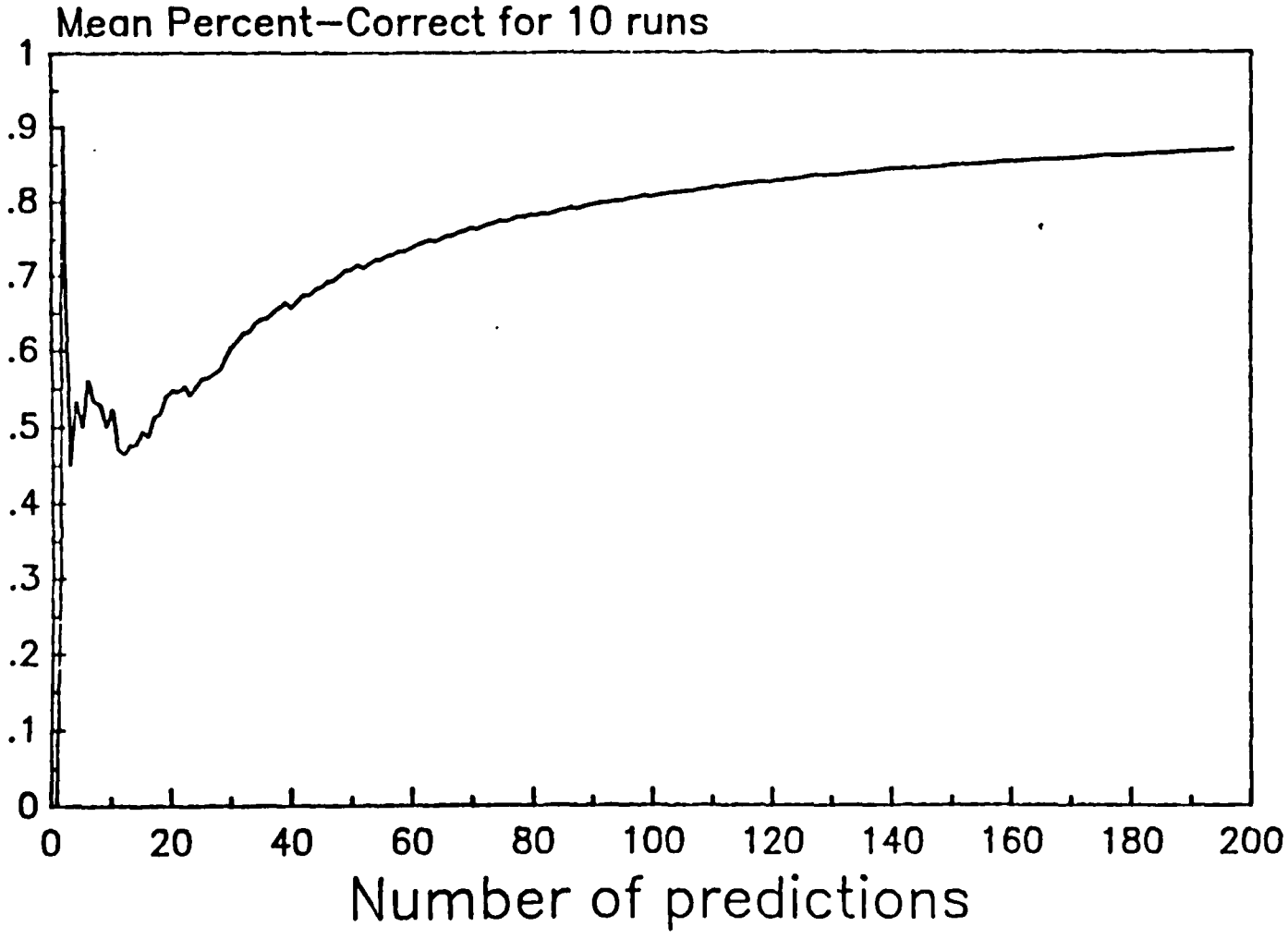


Figure 46

# Cyclic environment with unequal mutation variation

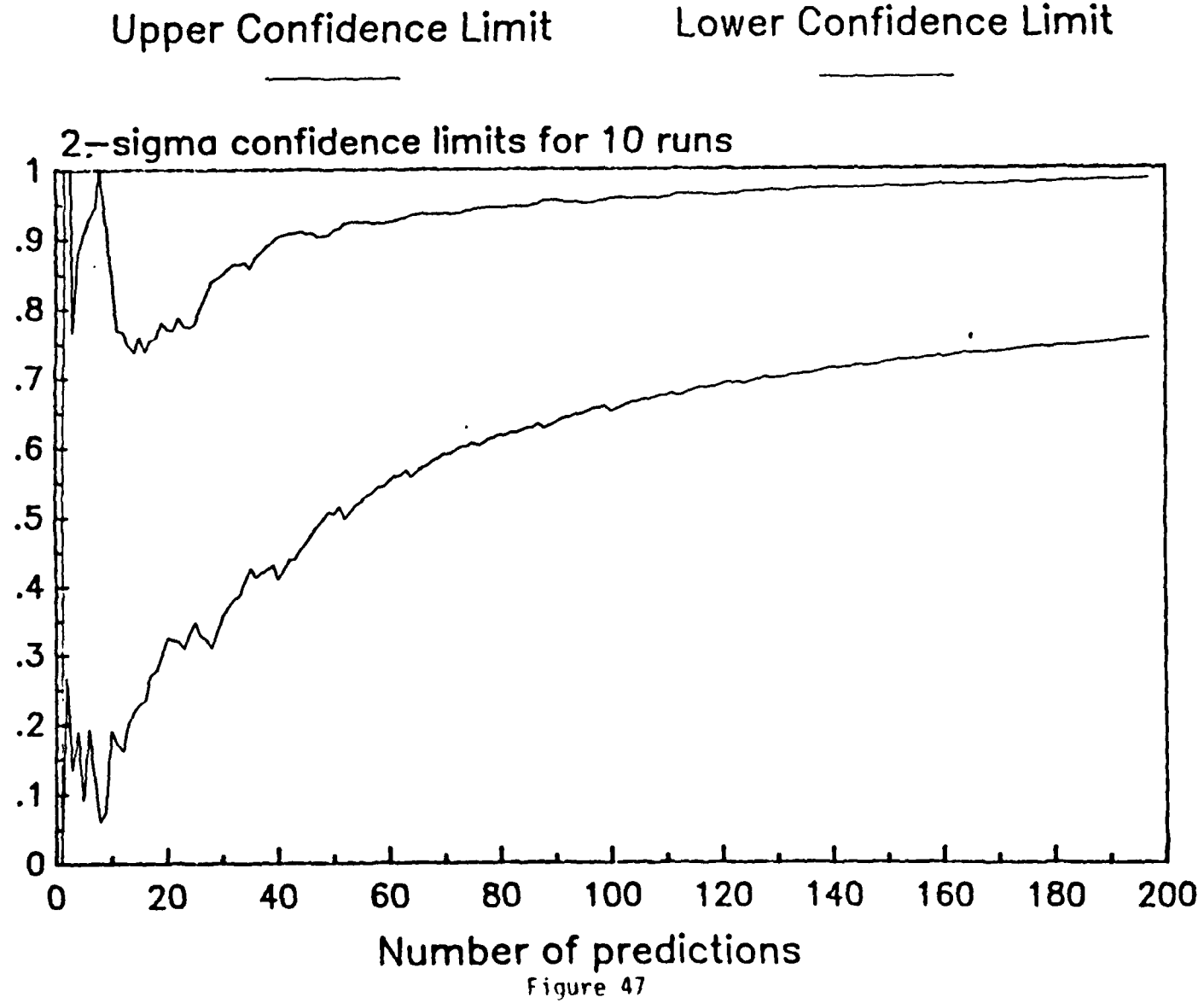


Figure 47

reflected difficulty in learning the second cyclic component of the environment, see Figures 48 through 52. Saving equally worthwhile offspring leads to a rapid increase in machine complexity, see Figures 53 through 57. It is believed that the added complexity that results from saving offspring of equal worth acts as a buffer to the coding that predicts the original environment. As expected, therefore, when the original environment was re-introduced, the evolutionary process quickly redemonstrated its previous ability. Again, note that an increase in size can markedly slow down the evolutionary process when it encounteres a new environment. Also note that the initial learning of the first sequence was slower than previously demonstrated. This was due to the fact that the evolutionary process "believed" it was dealing with a six-symbol environment instead of merely a two-symbol environment (the initial machine was experssed in a six-symbol language).

Similar experiments were also conducted wherein the cycles were all within the same four-symbol language. Here the sequence 0132 was repeated fifteen times followed by 331022 repeated fifteen times with a return to the original sequence repeated fifteen times. In the first experiment, only offspring that were superior to their parents in predictive fit were saved. Here the evolutionary process discovered a perfect predictor of the first cyclic sequence at the $20^{th}$ prediction, see Figure 58. 348 offspring were evolved to yield a four-state machine shown in Figure 59. Note that only a single state machine is necessary to predict this first sequence. After the environment changed at the $51^{st}$ prediction, the evolutionary process showed a sharp drop in success but regained to a prediction capability of five correct out of the six symbols. At the $141^{st}$ prediction, the original environment returned, and a perfect predictor was still evident. 200 predictions evaluated 6,585 offspring. Throughout the experiment, the machines remained of small size (three to four states) because here only offspring superior to the parent were saved.

In the second experiment, the offspring were saved if they scored equal to or better than the parent. Here again, the evolution indicated

# Double obverse: Two sub environments

(101110011101)x10 (223445)x10 (101110011101)x10

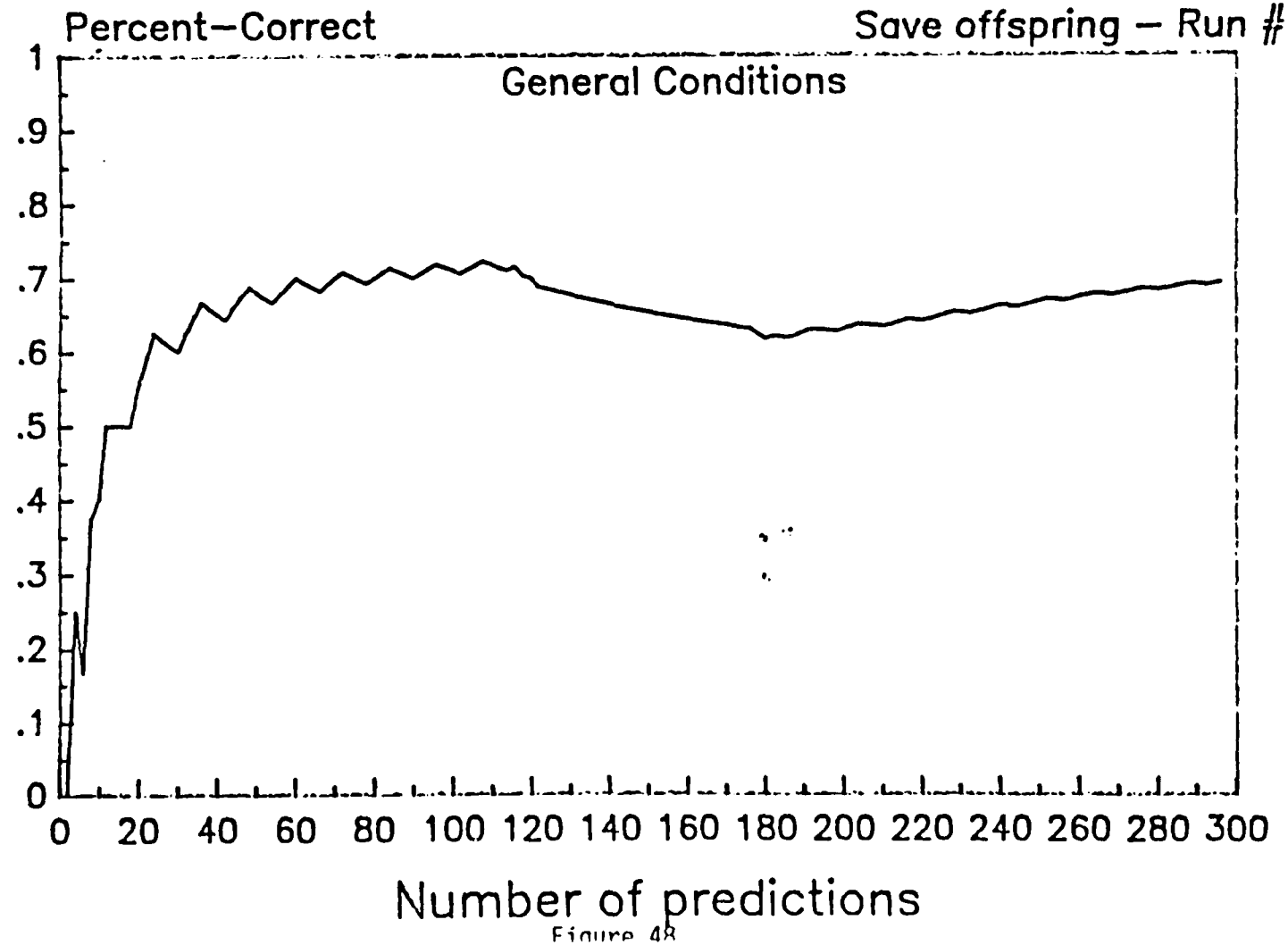


Figure 48



Figure 48

# Double obverse: Two sub environments

(101110011101)x10 (223445)x10 (101110011101)x10

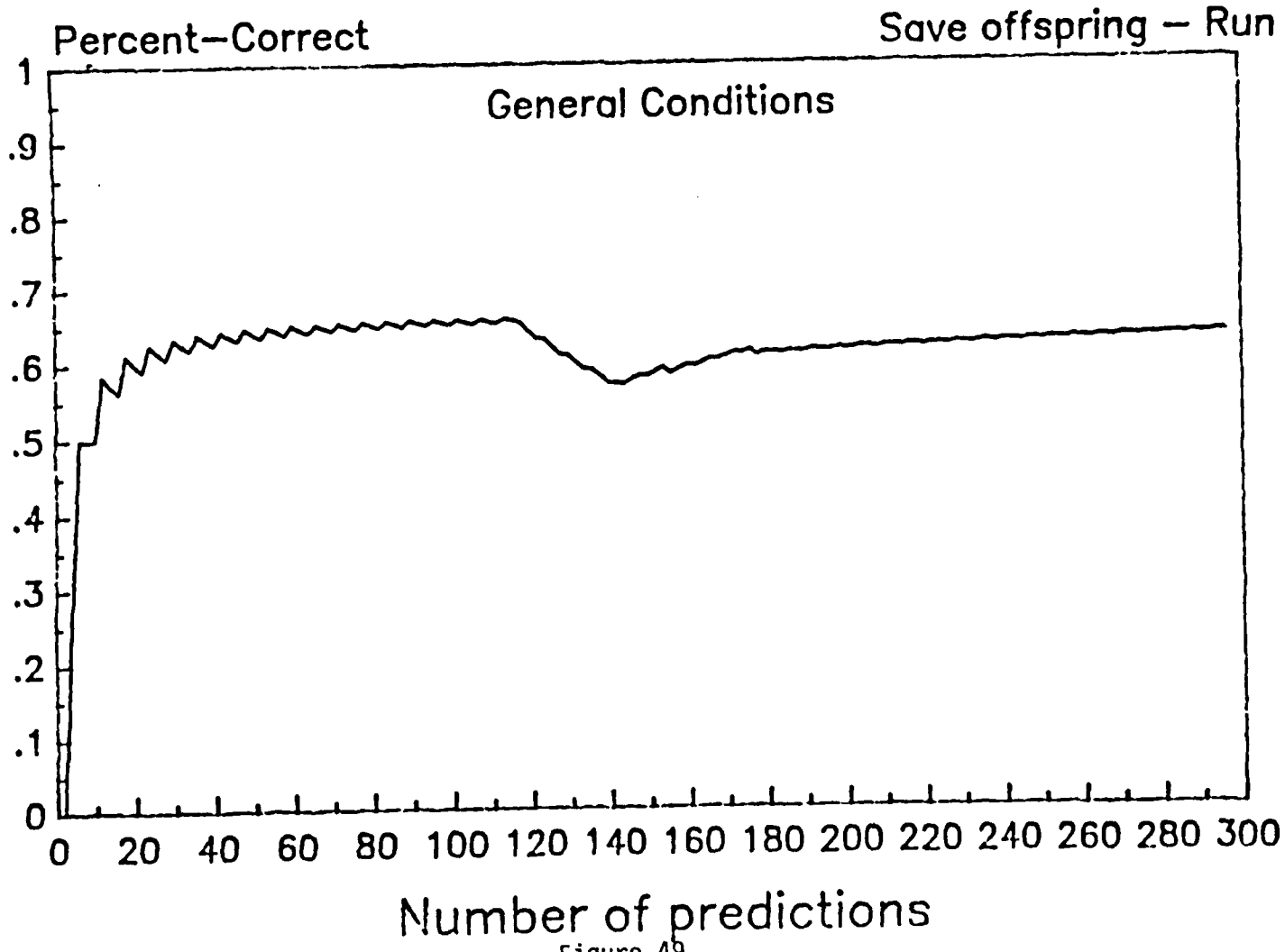


Figure 49

# Double obverse: Two sub environments

(101110011101)x10 (223445)x10 (101110011101)x10

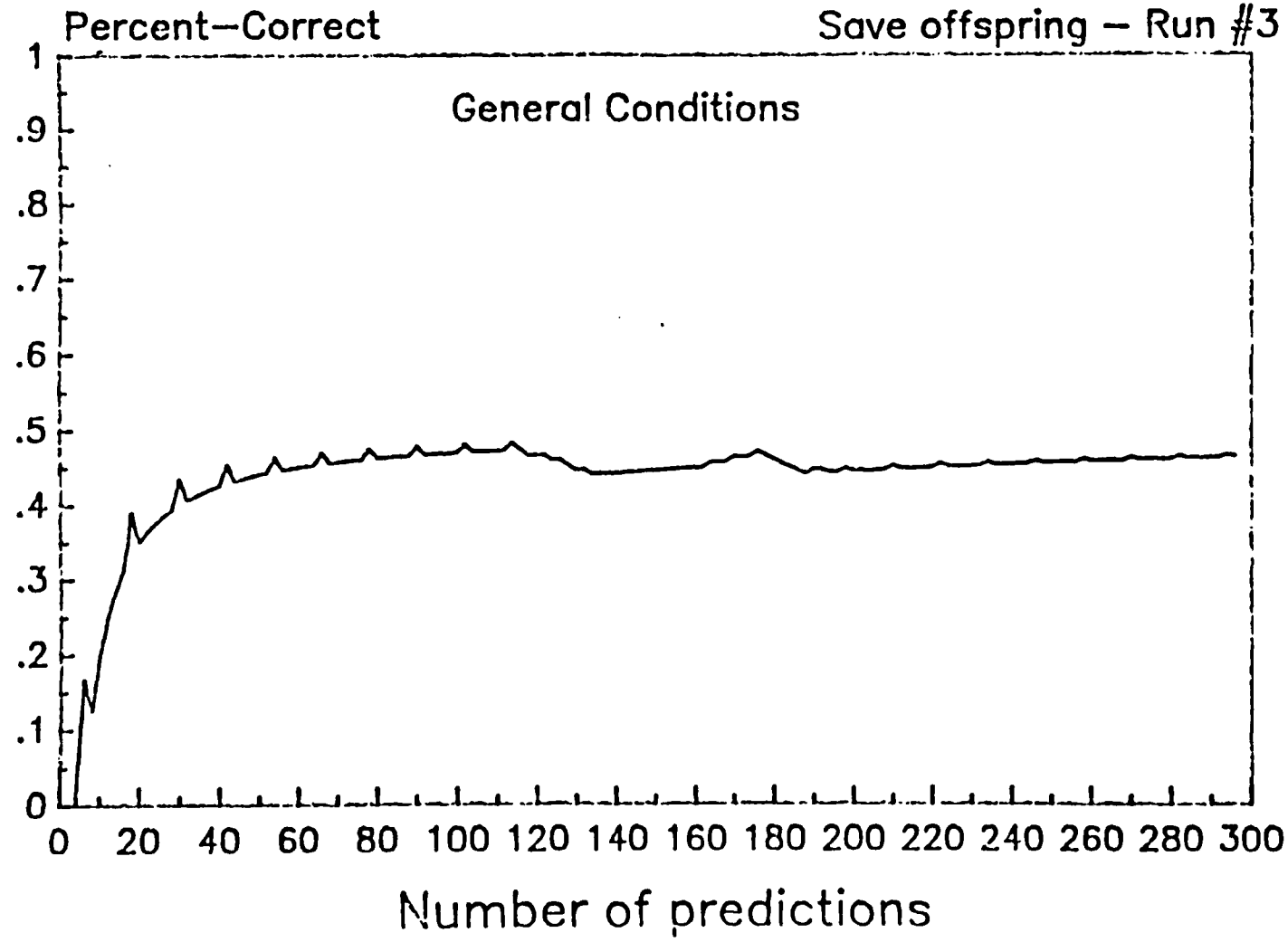


Figure 50

# Double obverse: Two sub environments

(101110011101)x10 (223445)x10 (101110011101)x10

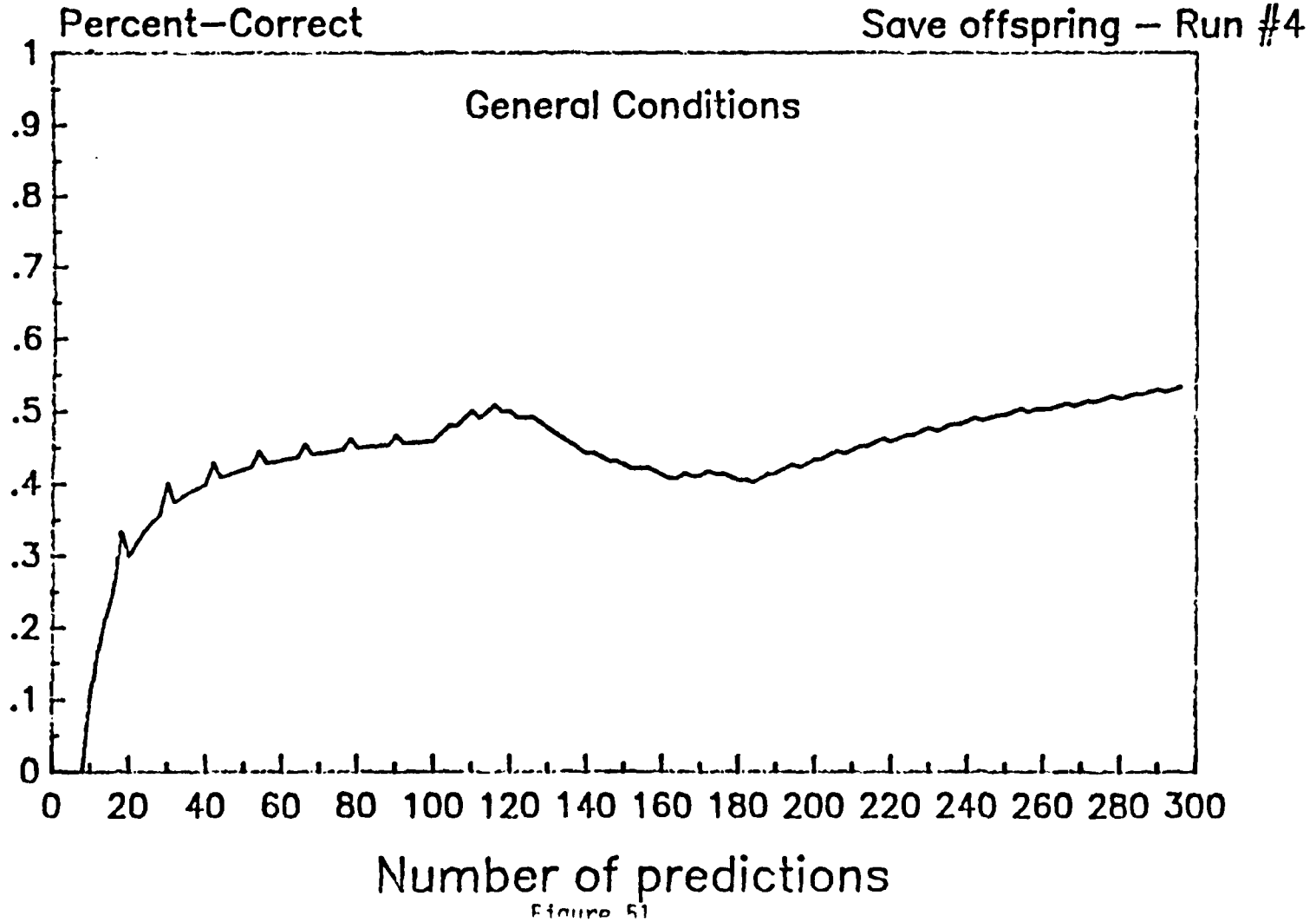


Figure 51



Figure 51

# Double obverse: Two sub environments

(101110011101)x10 (223445)x10 (101110011101)x10

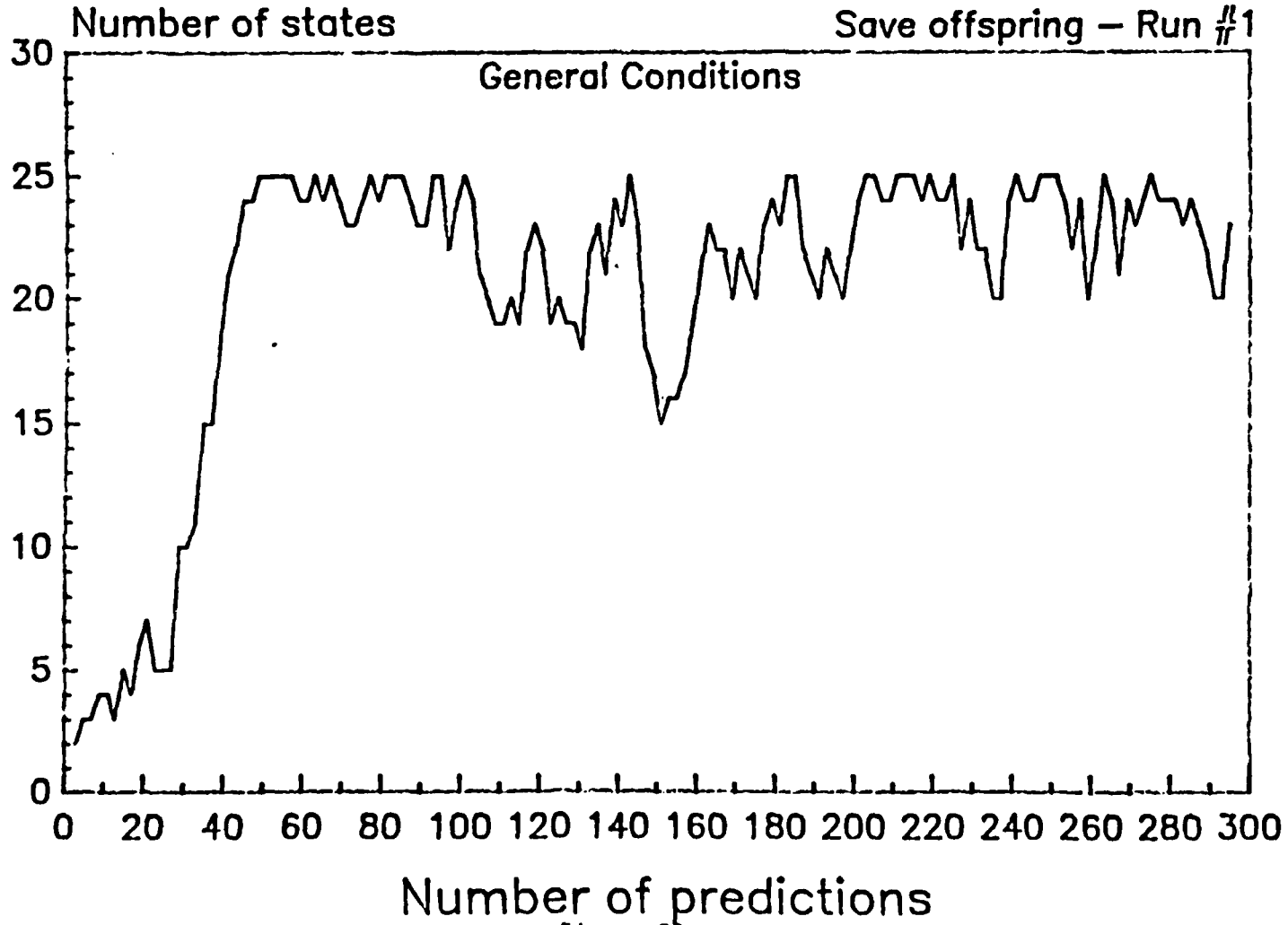


Figure 53



Figure 53

# Double obverse: Two sub environments

(101110011101)x10 (223445)x10 (101110011101)x10

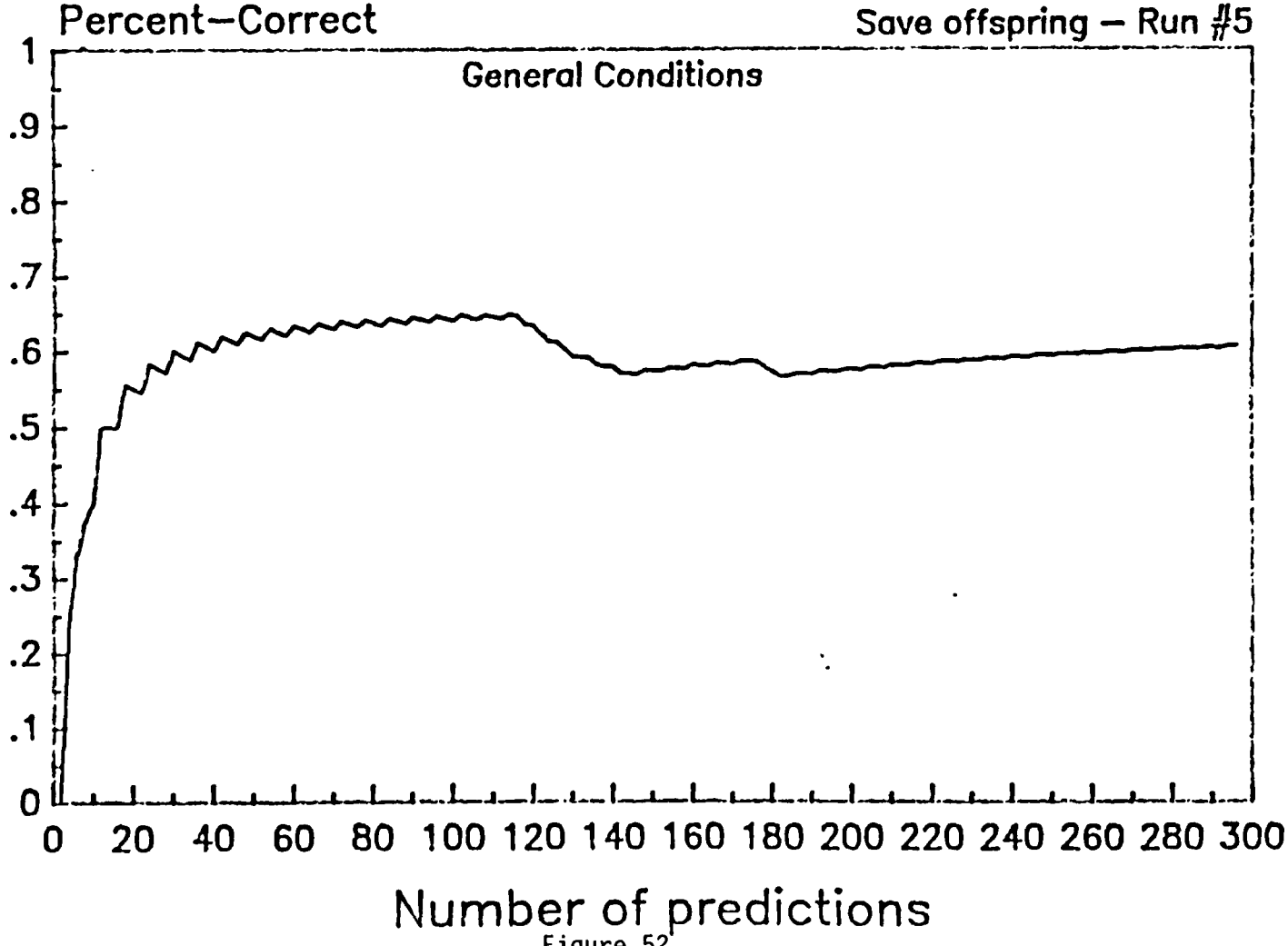


Figure 52

# Double obverse: Two sub environments

(101110011101)x10 (223445)x10 (101110011101)x10

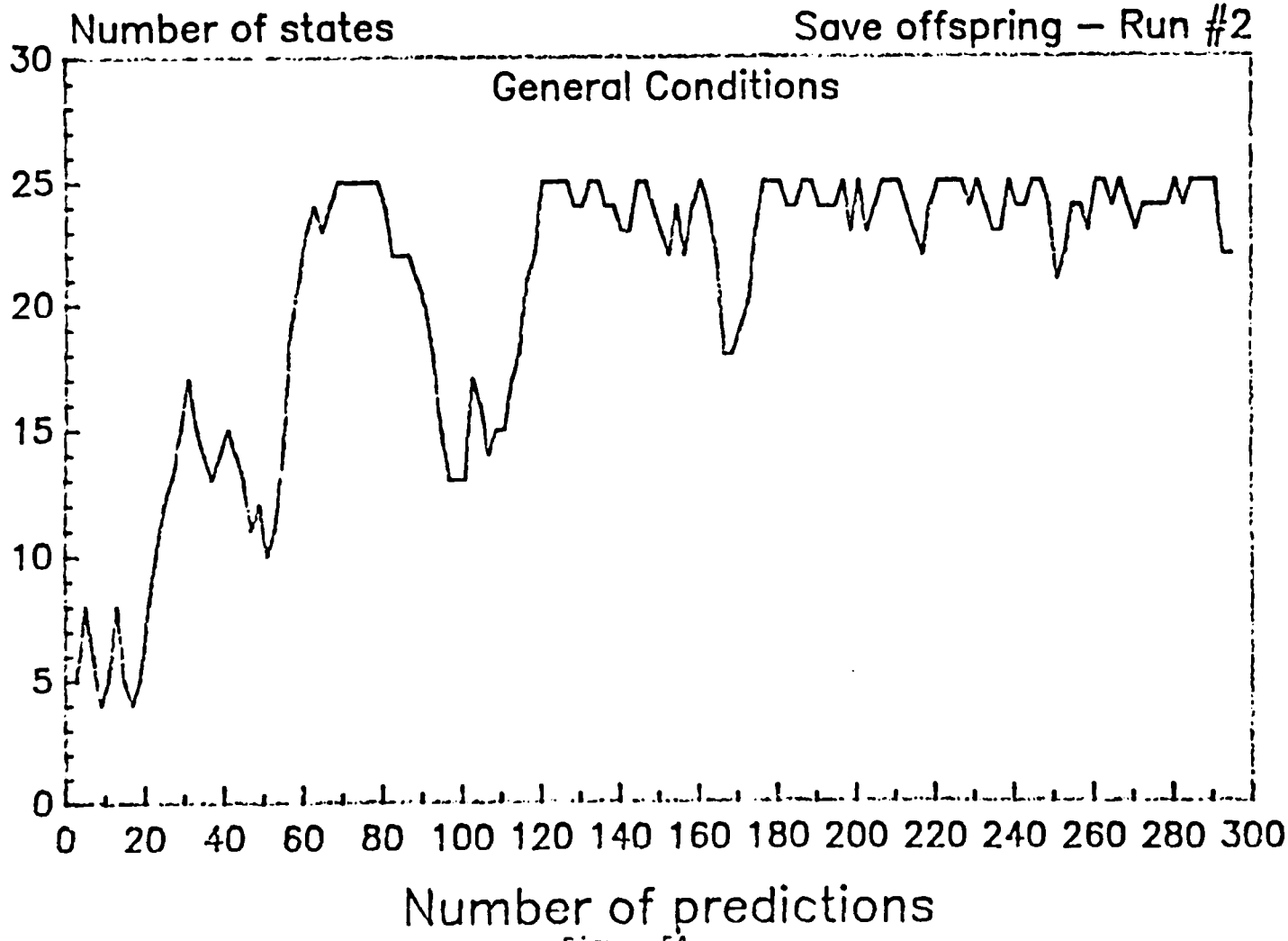


Figure 54



Figure 54

# Double obverse: Two sub environments

(101110011101)x10 (223445)x10 (101110011101)x10

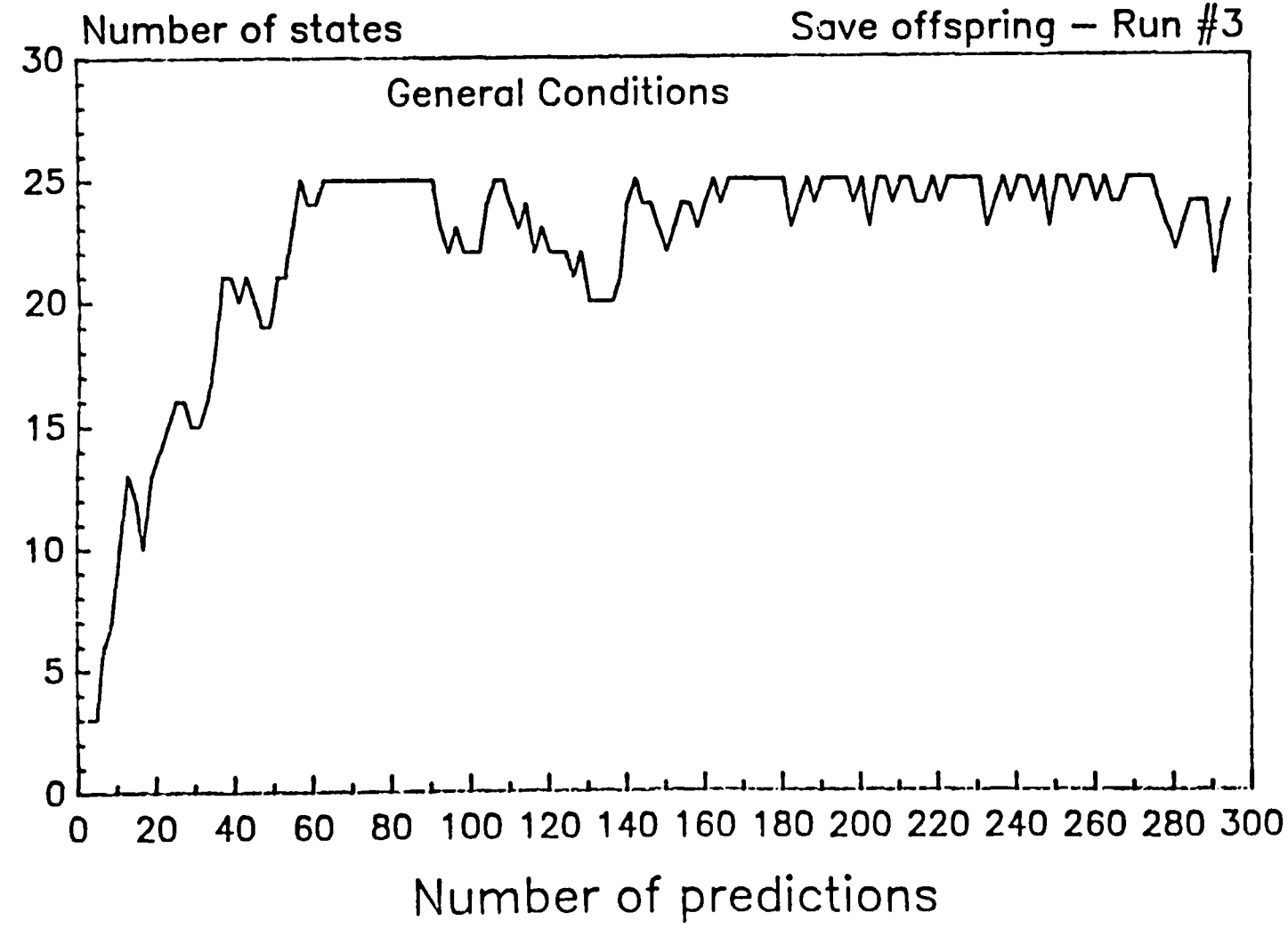


Figure 55

# Double obverse: Two sub environments

(101110011101)x10 (223445)x10 (101110011101)x10

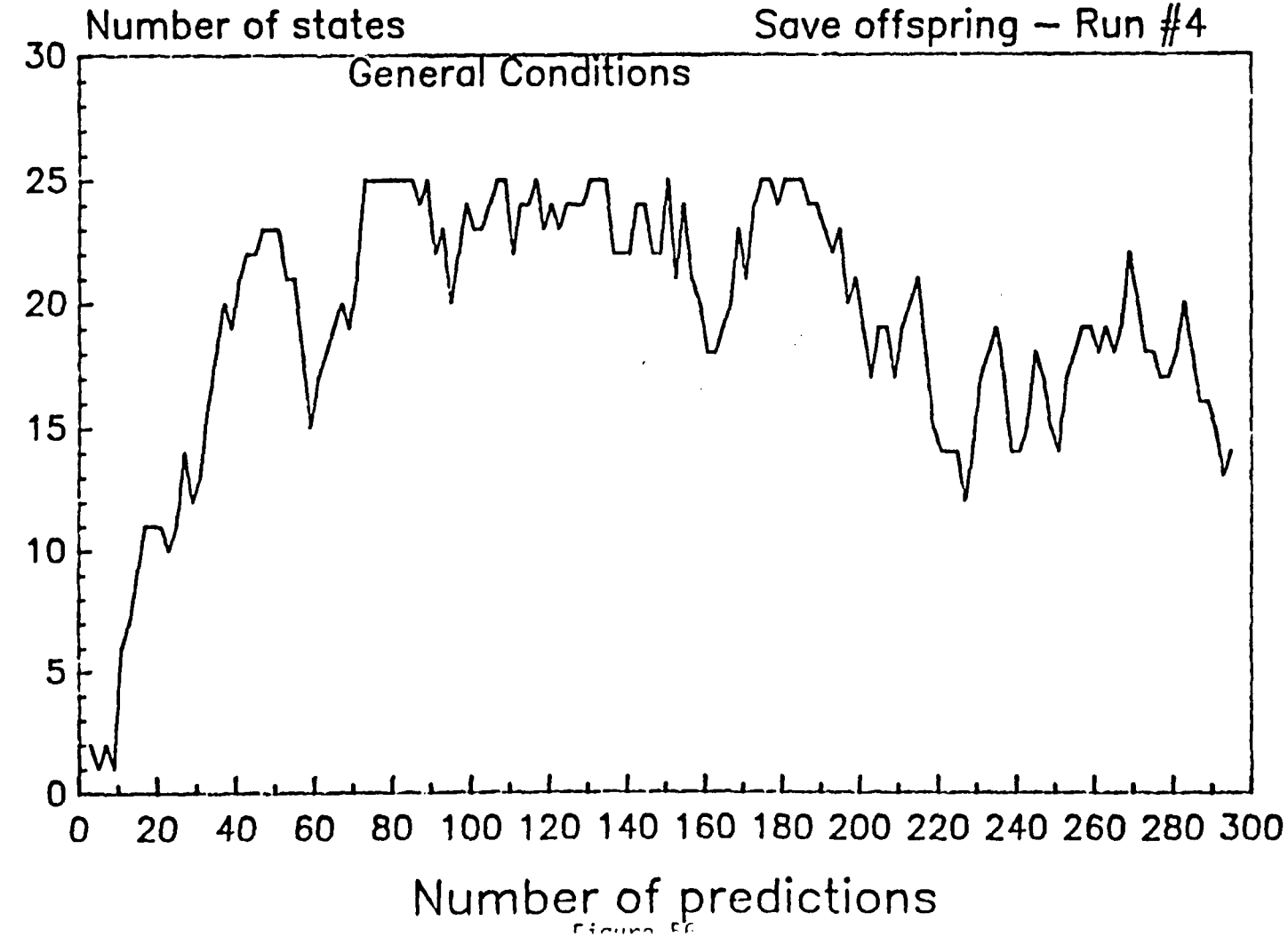


Figure 56

# Double obverse: Two sub environments

(101110011101)x10 (223445)x10 (101110011101)x10

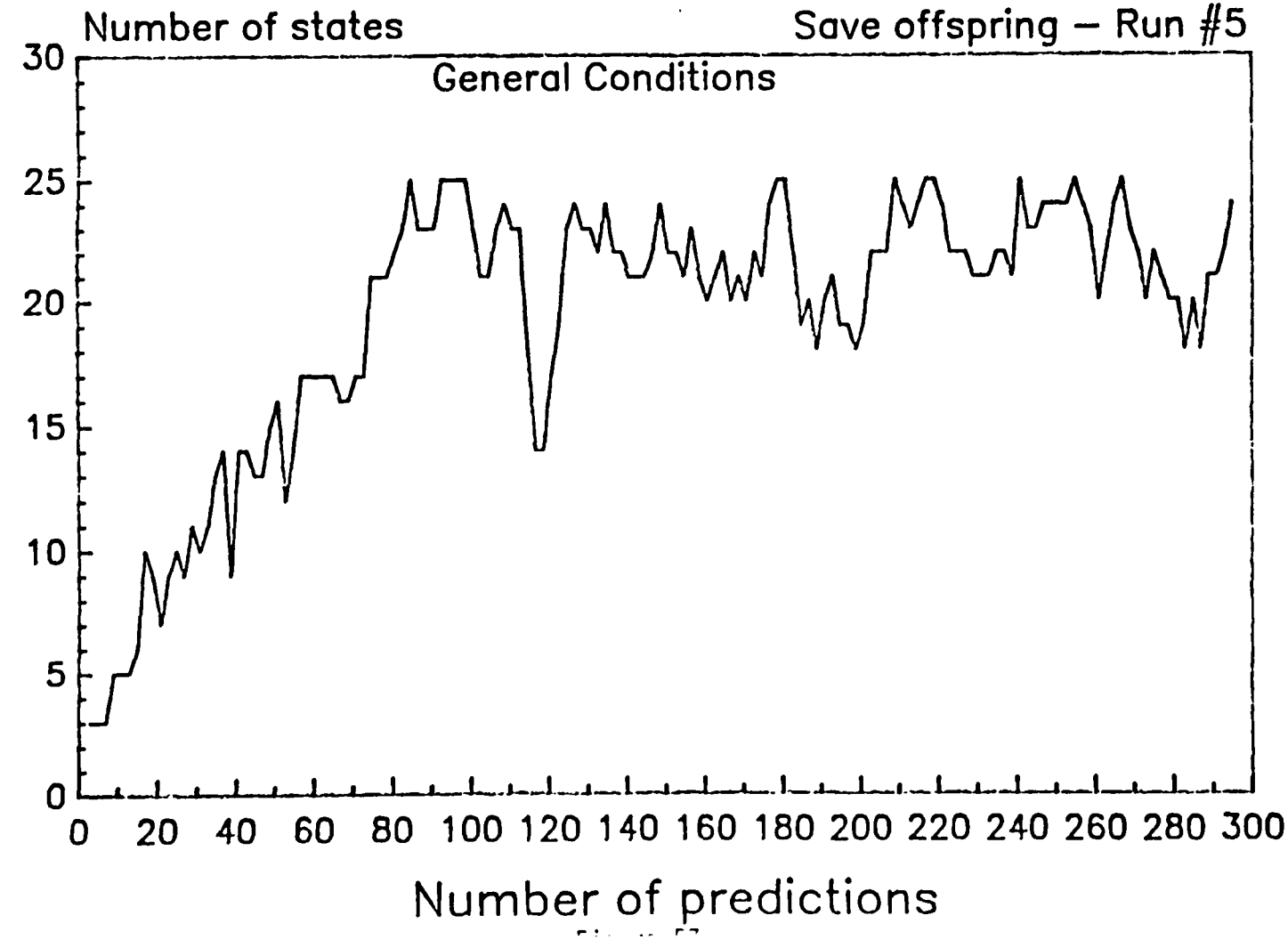


Figure 57

# Double obverse environment

Four symbols: (0132)x15 (331022)x15 (0132)x15

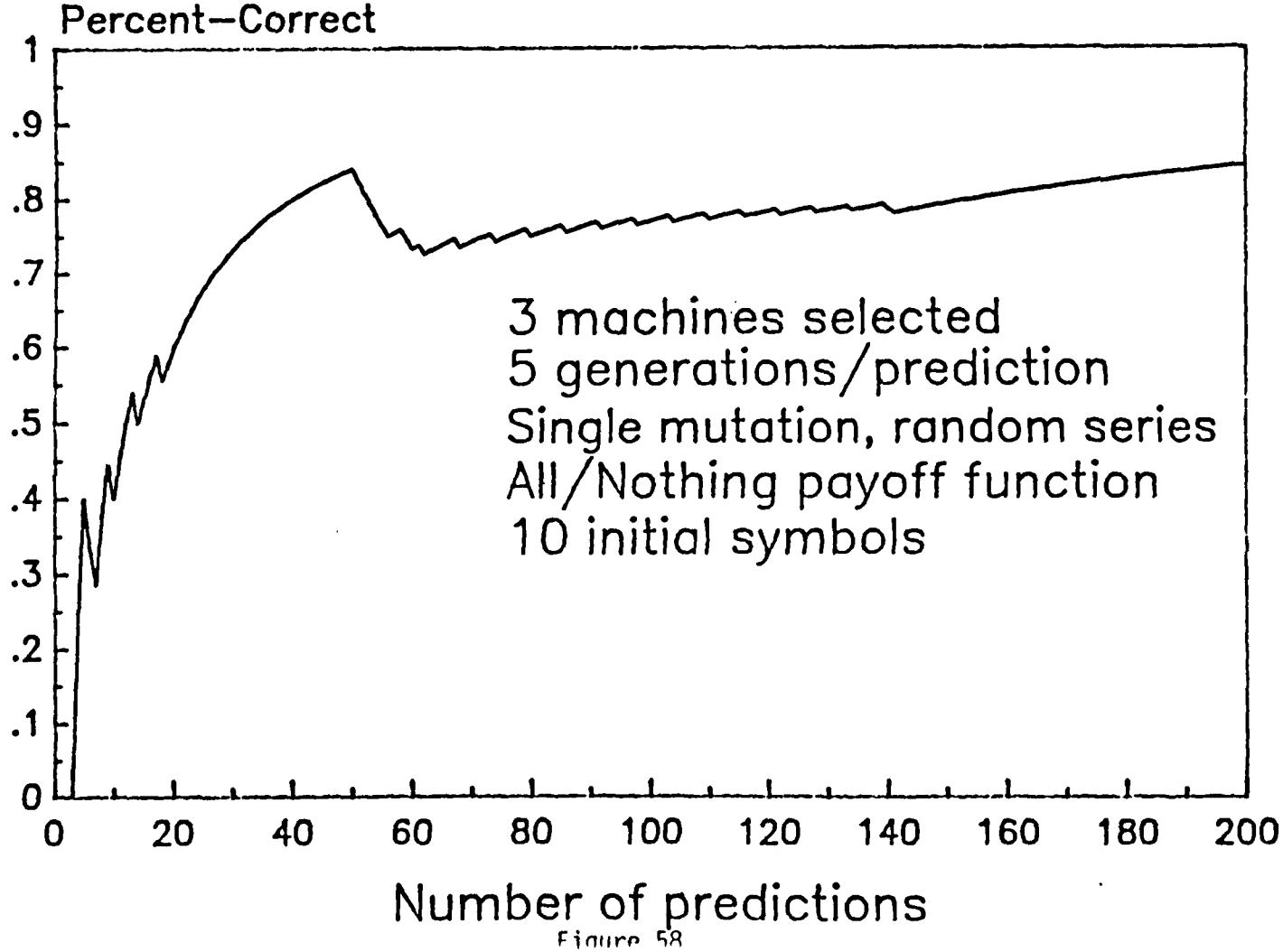


Figure 58



Figure 58

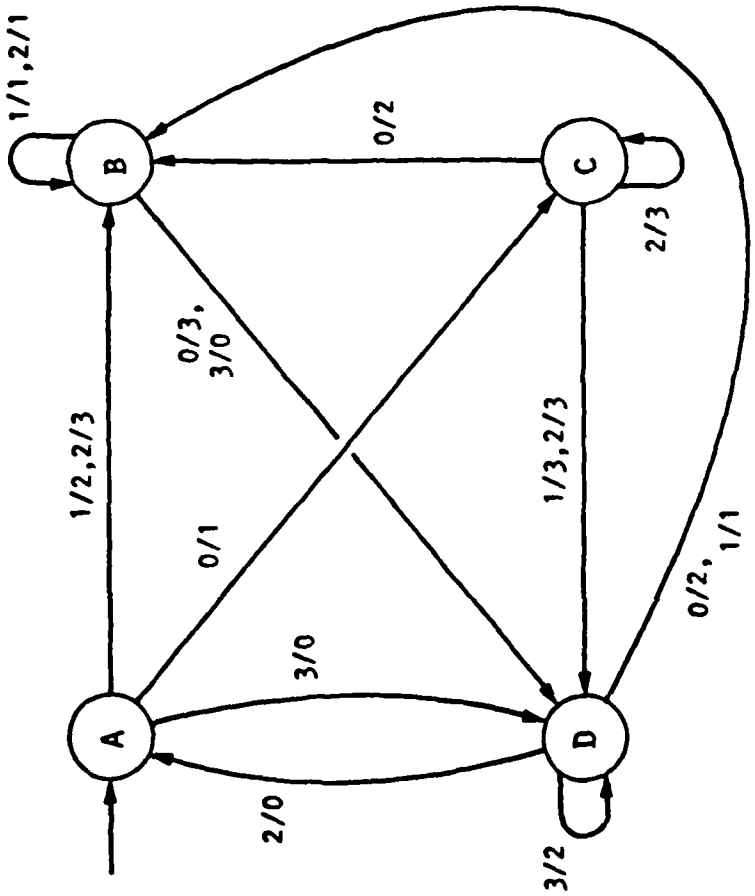


Figure 59

Figure 59

superior learning ability perfectly predicting the cyclic environment on only the eighth prediction (having seen only four cycles) see Figure 60. However, the evolutionary process had great difficulty in learning the second environment (due to the added complexity gained while learning the first environment). When the initial environment returned, a perfect predictor was still in evidence. The size of the finite state machines quickly grew to the upper limit of twenty-five states. This experiment tends to confirm the notion that large machines slow down the evolutionary process. It is useful to compare the above results to the classical zeroth order prediction of the same environment, reference Figure 61. Note its extremely poor performance.

Crossover Experiments

J.H. Holland, in his book, Adaptation in Natural and Artificial Systems, 1975, proposed the use of genetic operators. He believed that simulated evolution could be improved by drawing an anology to sexual reproduction. Specifically, he suggested that two machines be "crossed-over" through a substitution of states. A large number of experiments have been reported by Holland and others on the use of this technique... without their being any reference to finite state machines as being the embodiment of the evolving behavior.

Several experiments were therefore conducted to investigate the worth of this proposition. The first fifteen experiments involved the same binary cyclic environment and used a two-state crossover, that is, two arbitrary states of the best machine was substituted for two states in another machine of similar size. The probability of this crossover was chosen to be fifty percent with the remaining probabilities uniformly distributed over the five modes of mutation. Figures 62 through 67 show that the best experiment discovered a perfect predictor at the 63$^{rd}$ prediction. 2,976 offspring were evaluted, and the size of the evolving machines rose steadily. The worst experiment did not discover a perfect predictor in 196 predictions. 3,018 offspring were evaluated. The

# Double obverse environment

Four symbols: (0132)x15 (331022)x15 (0132)x15

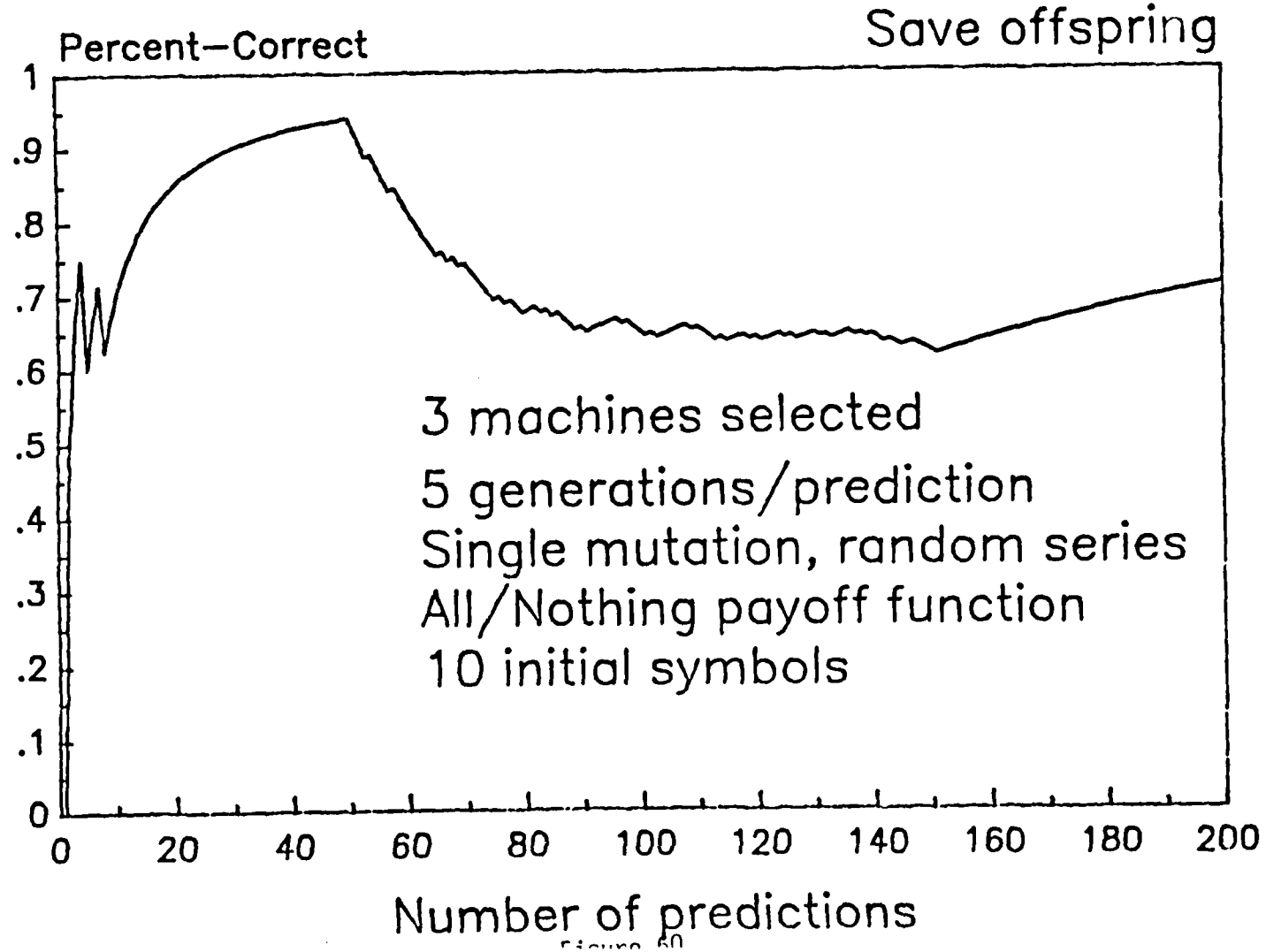


Figure 60

# Double obverse environment

Four symbols: (0132)x15 (331022)x15 (0132)x15

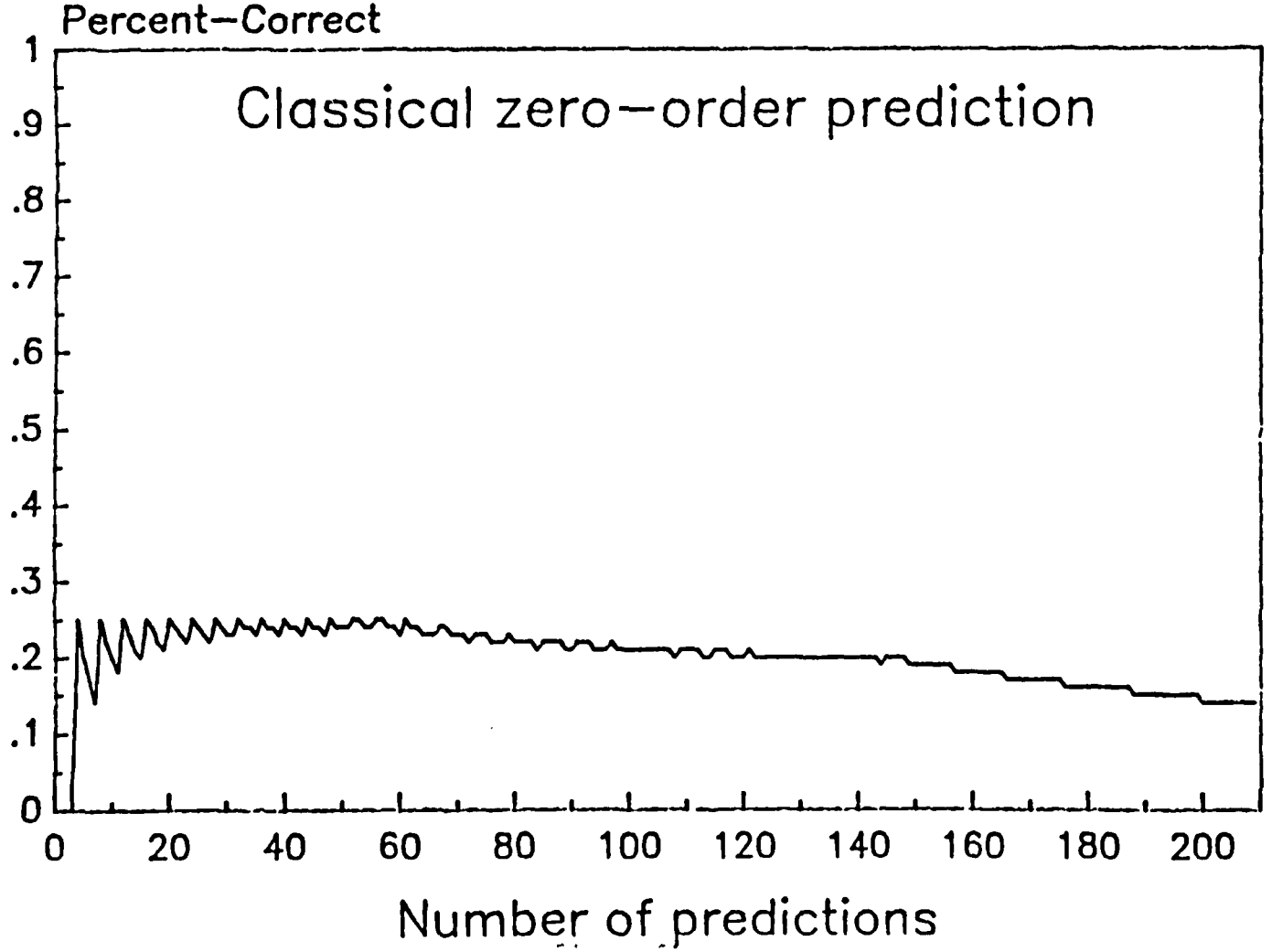


Figure 61

# Using Simple Crossover to Predict (101110011101) cyclic

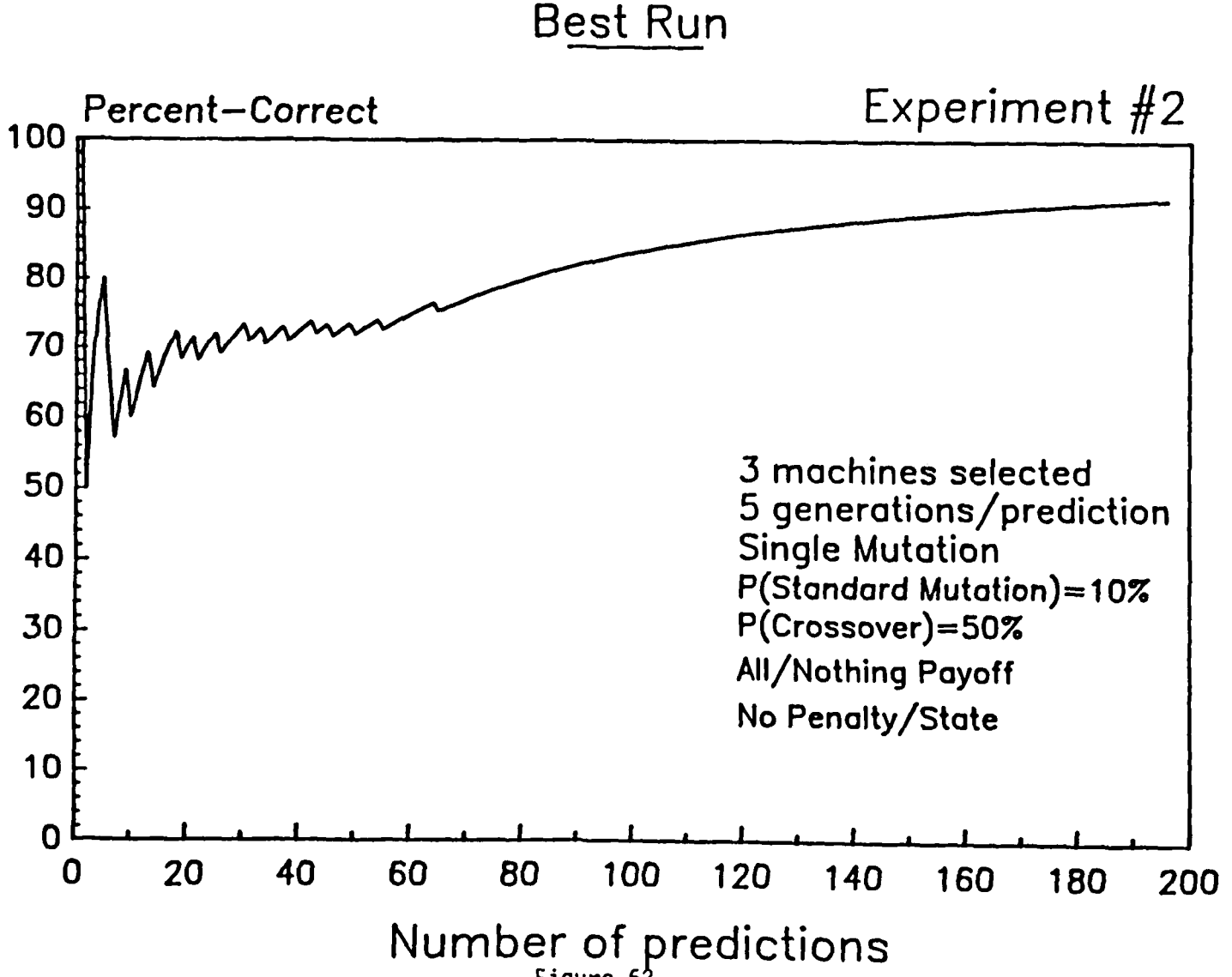


Figure 62

# Using Simple Crossover to Predict (101110011101) cylic

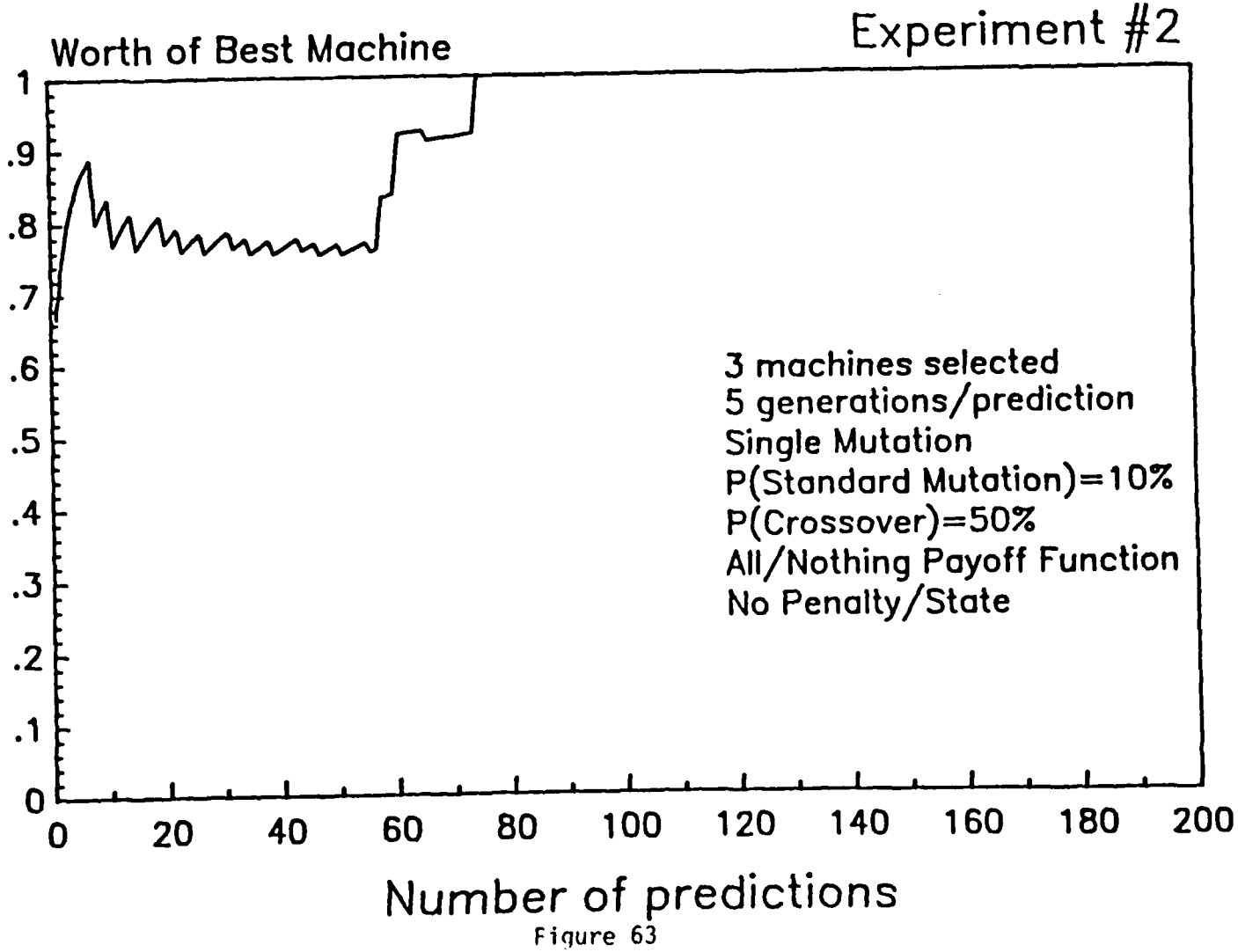


Figure 63



Figure 63

# Using Simple Crossover to Predict (101110011101) cyclic

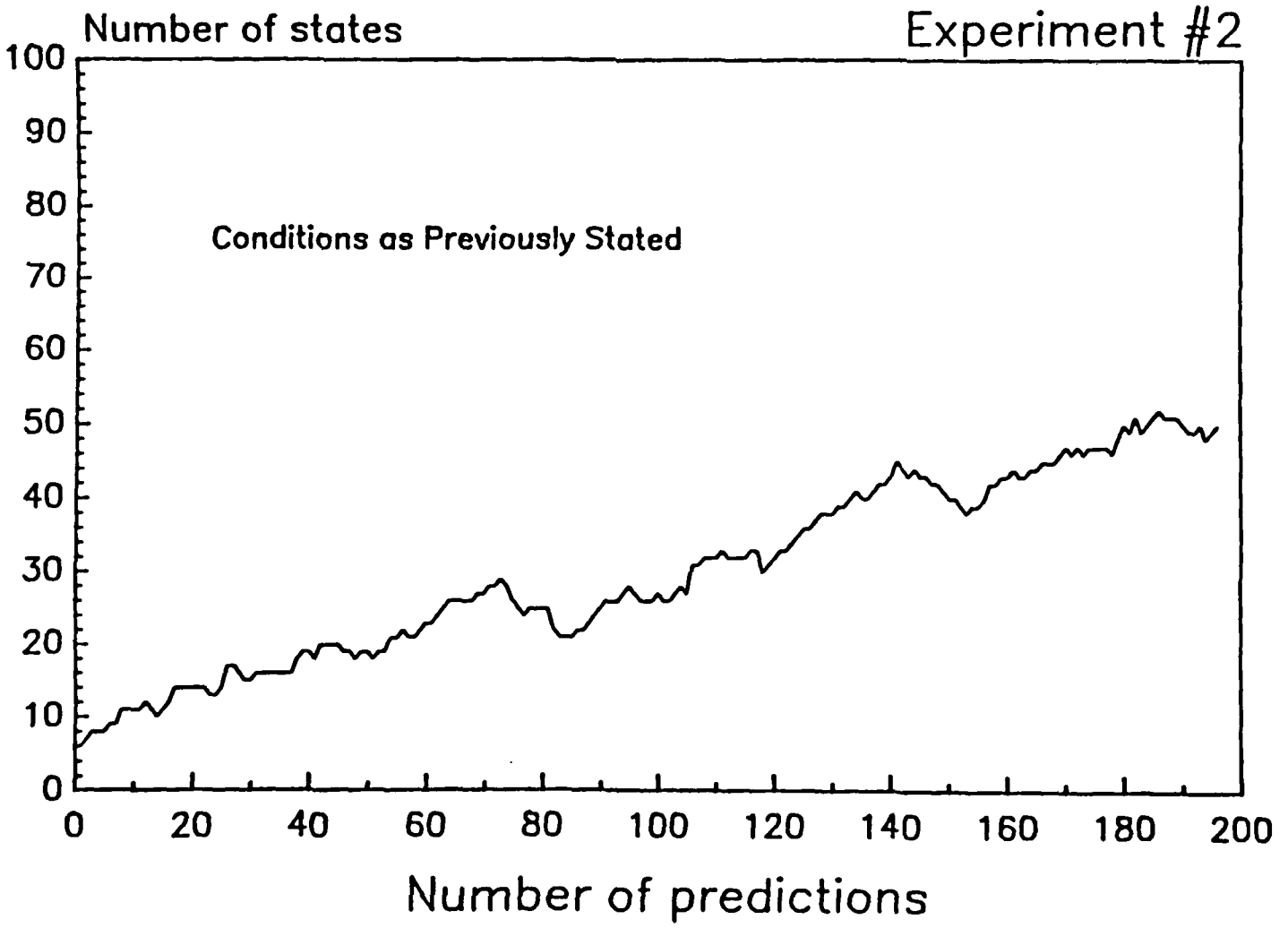


Figure 64



Figure 64

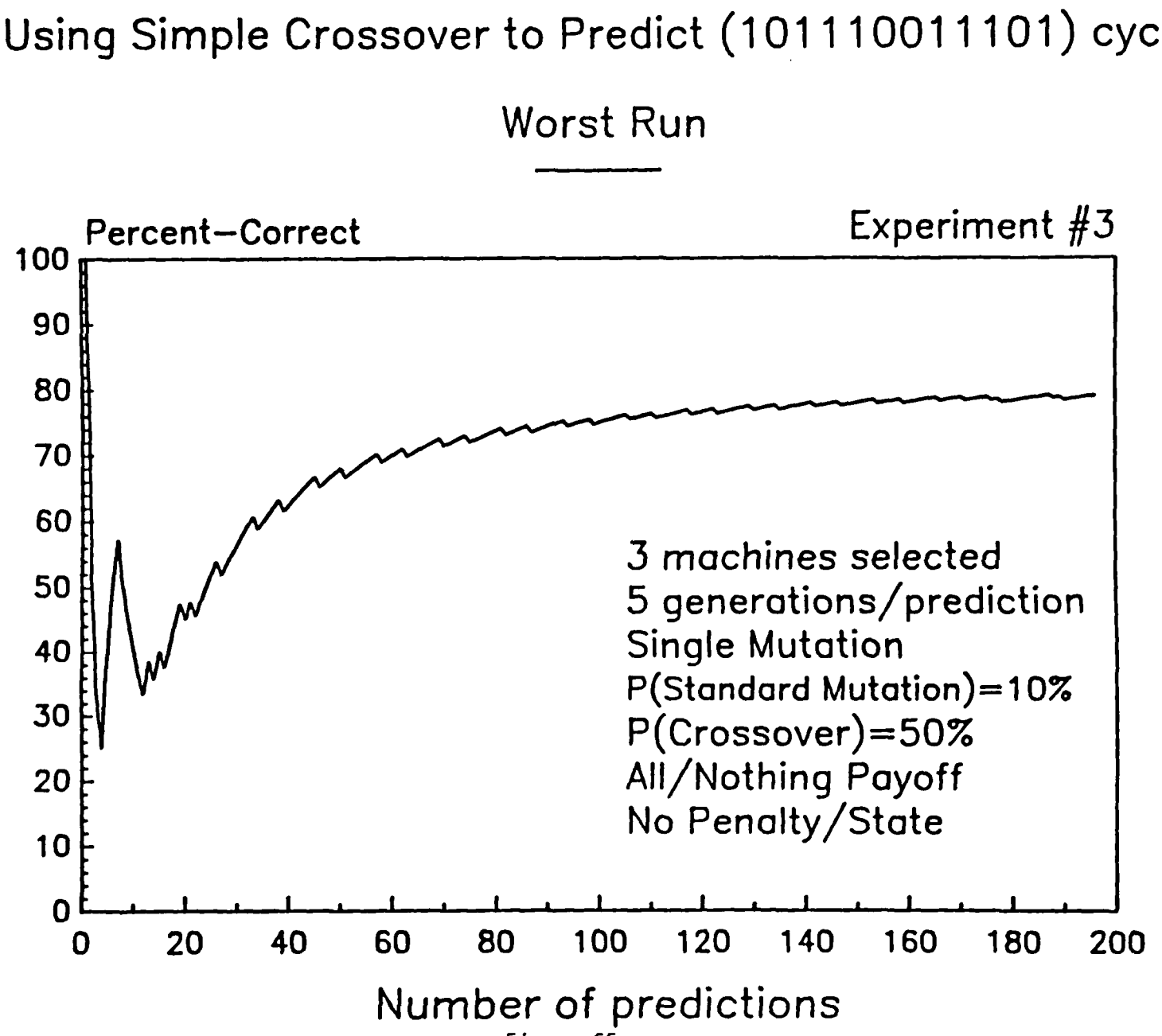
Using Simple Crossover to Predict (101110011101) cyclic
Worst Run
Percent-Correct
Experiment #3
100
90
80
70
60
50
40
30
20
10
0
0
20
40
60
80
100
120
140
160
180
200
Number of predictions
3 machines selected
5 generations/prediction
Single Mutation
P(Standard Mutation)=10%
P(Crossover)=50%
All/Nothing Payoff
No Penalty/State

Figure 65

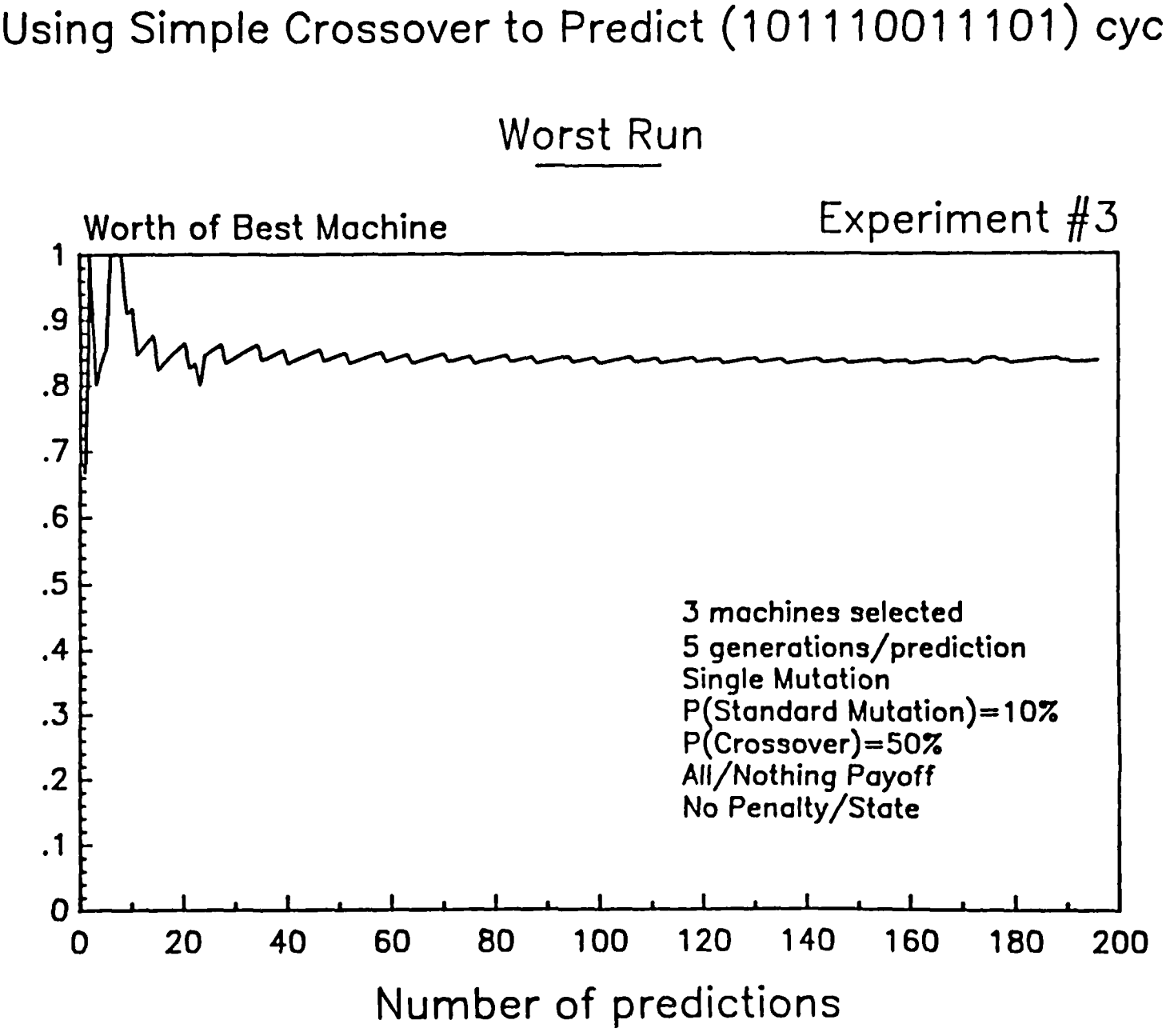
Using Simple Crossover to Predict (101110011101) cyclic
Worst Run
Worth of Best Machine
Experiment #3
1
.9
.8
.7
.6
.5
.4
.3
.2
.1
0
0
20
40
60
80
100
120
140
160
180
200
3 machines selected
5 generations/prediction
Single Mutation
P(Standard Mutation)=10%
P(Crossover)=50%
All/Nothing Payoff
No Penalty/State
Number of predictions

Figure 66

# Using Simple Crossover to Predict (101110011101) cyclic

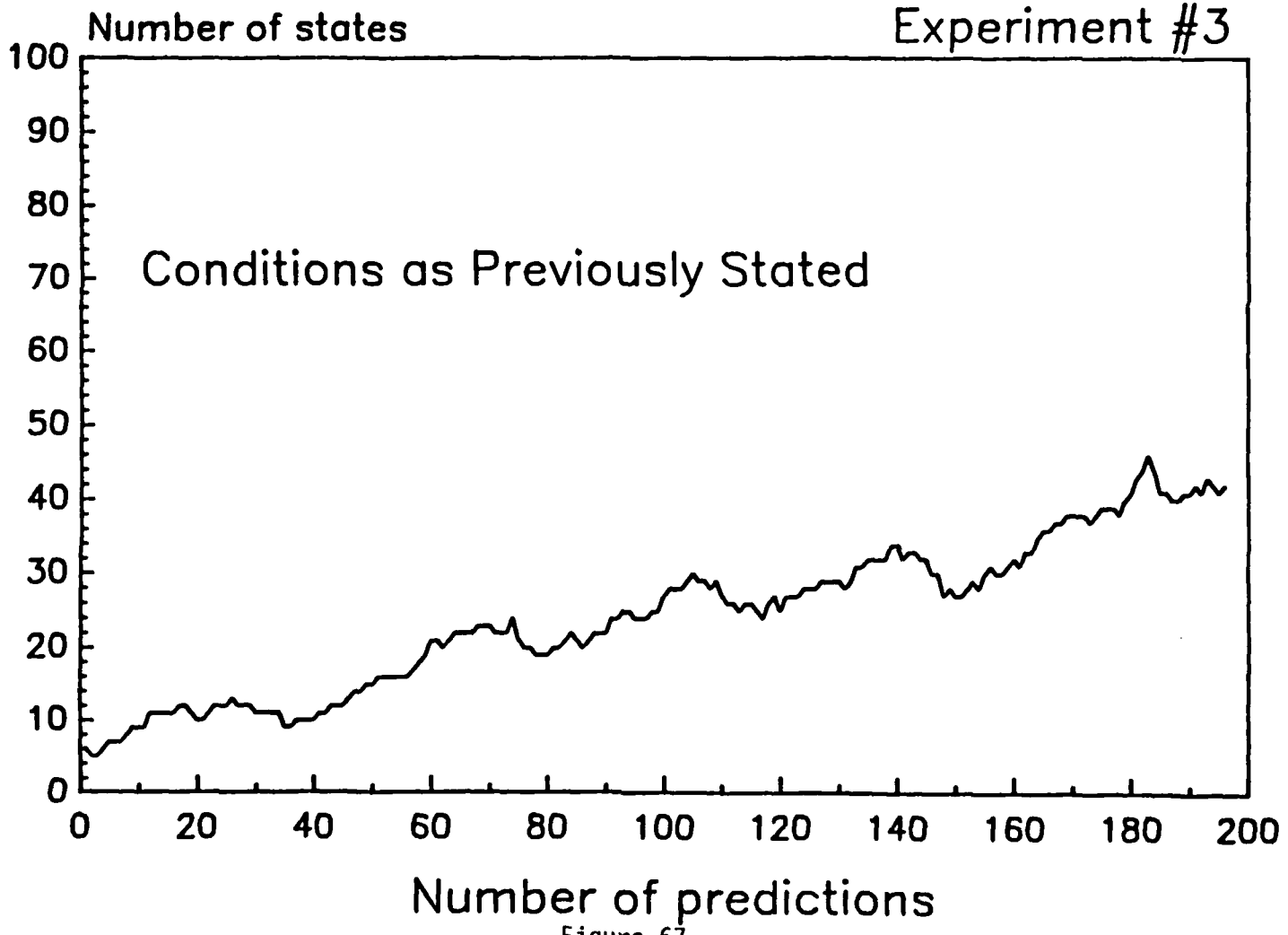


Figure 67

increase in machine size was similar to that previously demonstrated. While the mean worth (predictive fit) of the process consistently increased (Figures 68 and 69) the mean percent correct of future predictions was almost identical to that without crossover; see Figures 70 and 71. On the average, 3,001.2 offspring were evaluated in 196 predictions.

Crossover was also examined within the context of the I.Q. test environment. Twenty experiments were conducted, each performing 85 predictions with an all-or-none payoff function. Figures 72 and 73 indiate the mean cumulative percent correct and the two sigma limits. After the $40^{th}$ prediction, the process properly predicts all the zeros, although with a significant variance. However, Figures 74 and 75 show that the predictive fit variation around the mean is very narrow. All machines tend to fit the previous history in the same manner. There was little difference in the results with and without crossover.

The environment was then predicted using the asymmetric payoff function.

| | | predicted 0 | predicted 1 |
|---|---|---|---|
| actual | 0 | 3 | 0 |
| | 1 | 0 | 10 |

Figures 76 through 79 show the result of eighteen experiments. The mean worth of the abilities to fit previous history was slightly worse than that without any crossover. The poorest results asymptotically fell toward the expected value of the environment. The experiment with the best results showed some ability to predict the early one's.

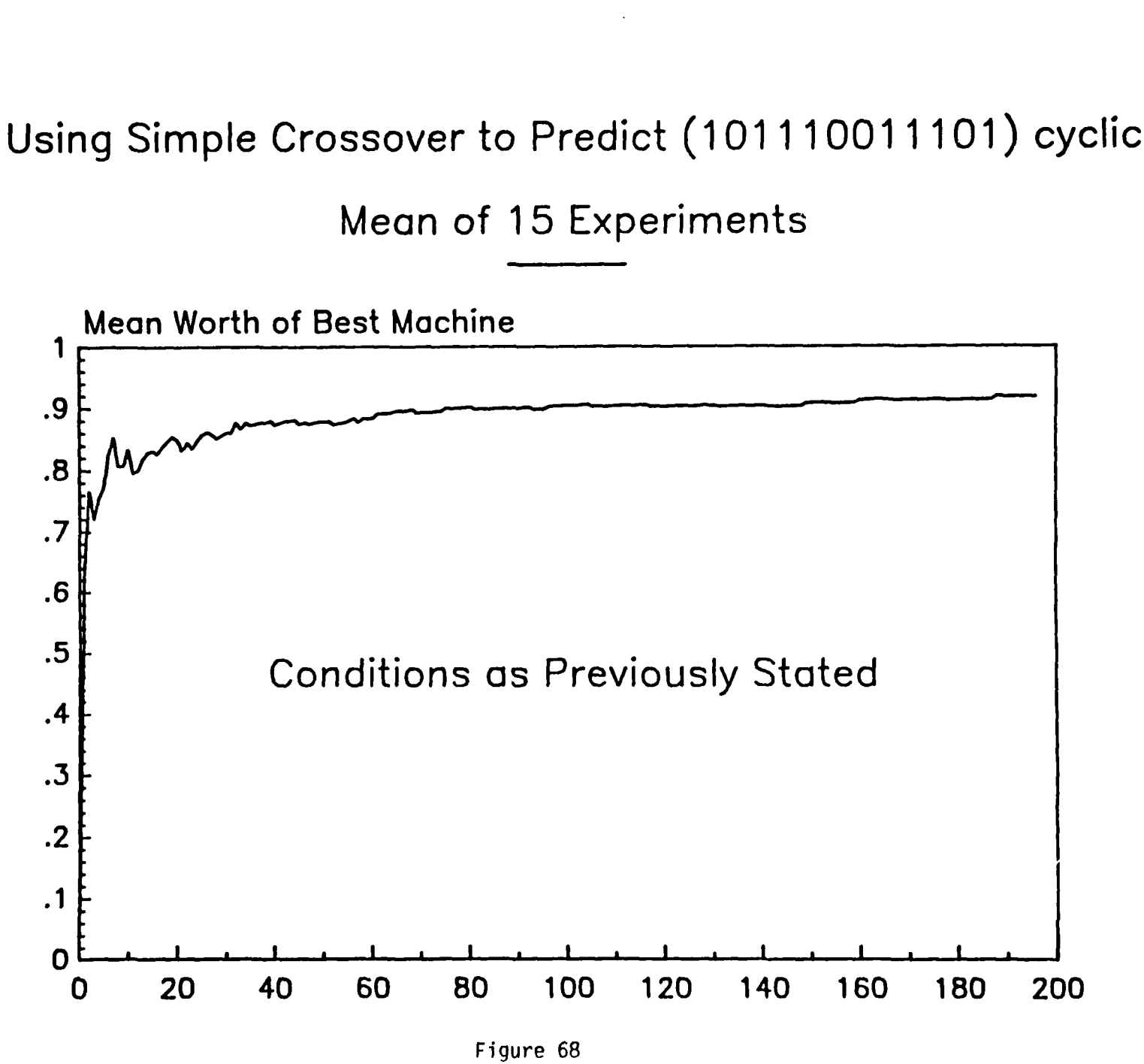
Using Simple Crossover to Predict (101110011101) cyclic
Mean of 15 Experiments
Mean Worth of Best Machine
1
.9
.8
.7
.6
.5
.4
.3
.2
.1
0
0
20
40
60
80
100
120
140
160
180
200
Conditions as Previously Stated

Figure 68

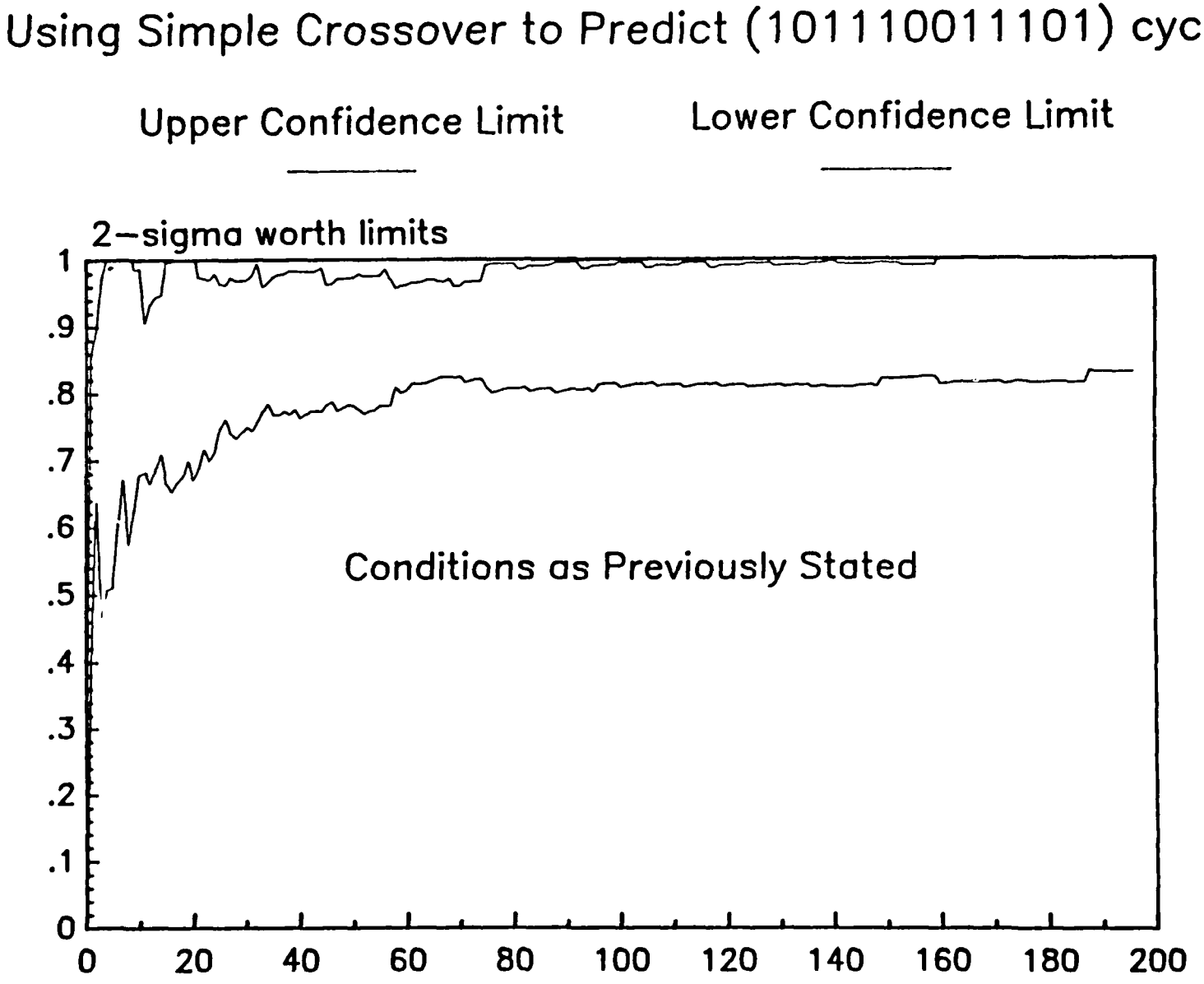
Using Simple Crossover to Predict (101110011101) cyclic
Upper Confidence Limit
Lower Confidence Limit
2−sigma worth limits
1
.9
.8
.7
.6
.5
.4
.3
.2
.1
0
0
20
40
60
80
100
120
140
160
180
200
Conditions as Previously Stated

Figure 69

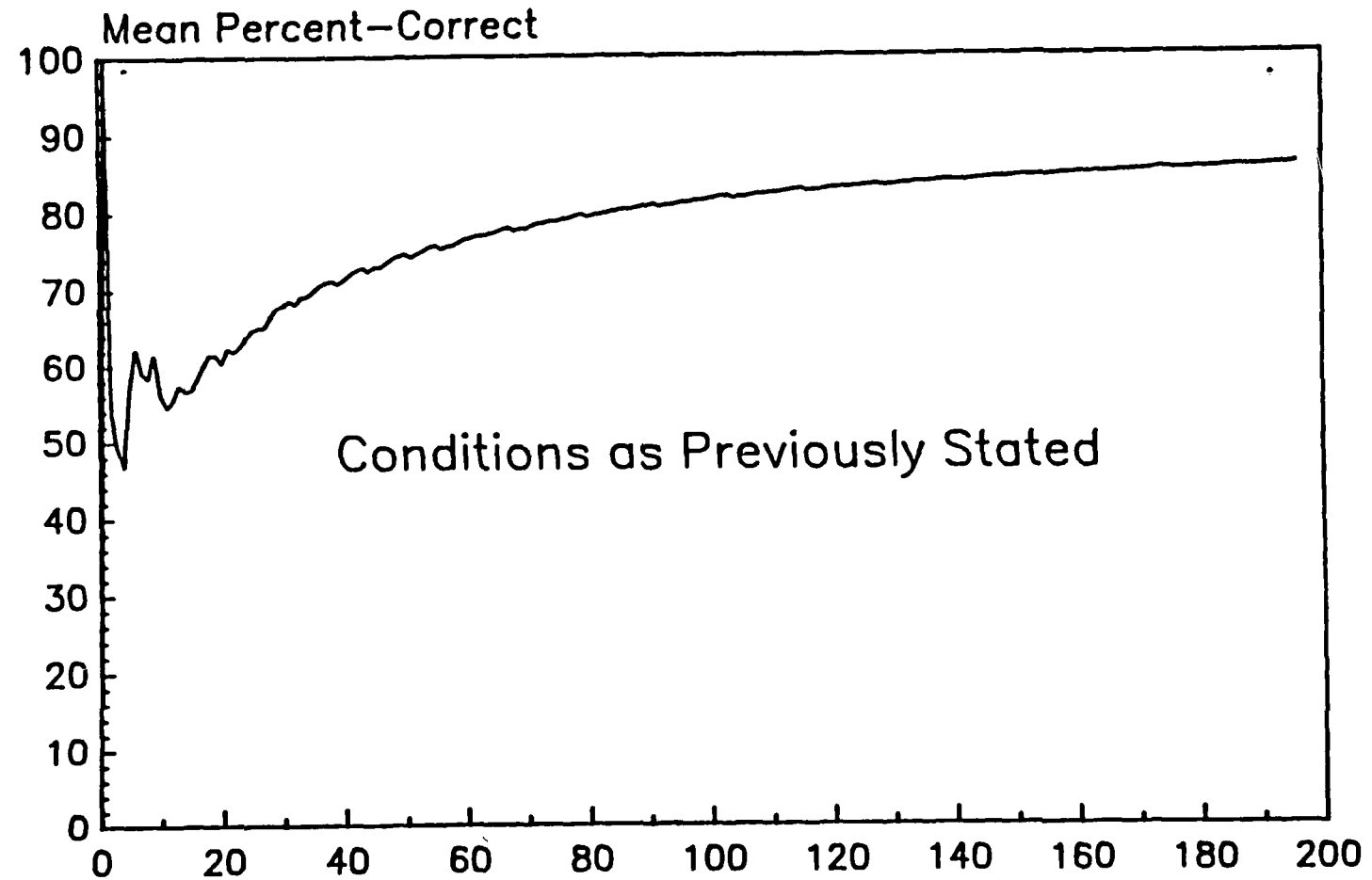
Using Simple Crossover to Predict (101110011101) cyclic
Mean of 15 Experiments
Mean Percent-Correct
100
90
80
70
60
50
40
30
20
10
0
0
20
40
60
80
100
120
140
160
180
200
Conditions as Previously Stated

Figure 70

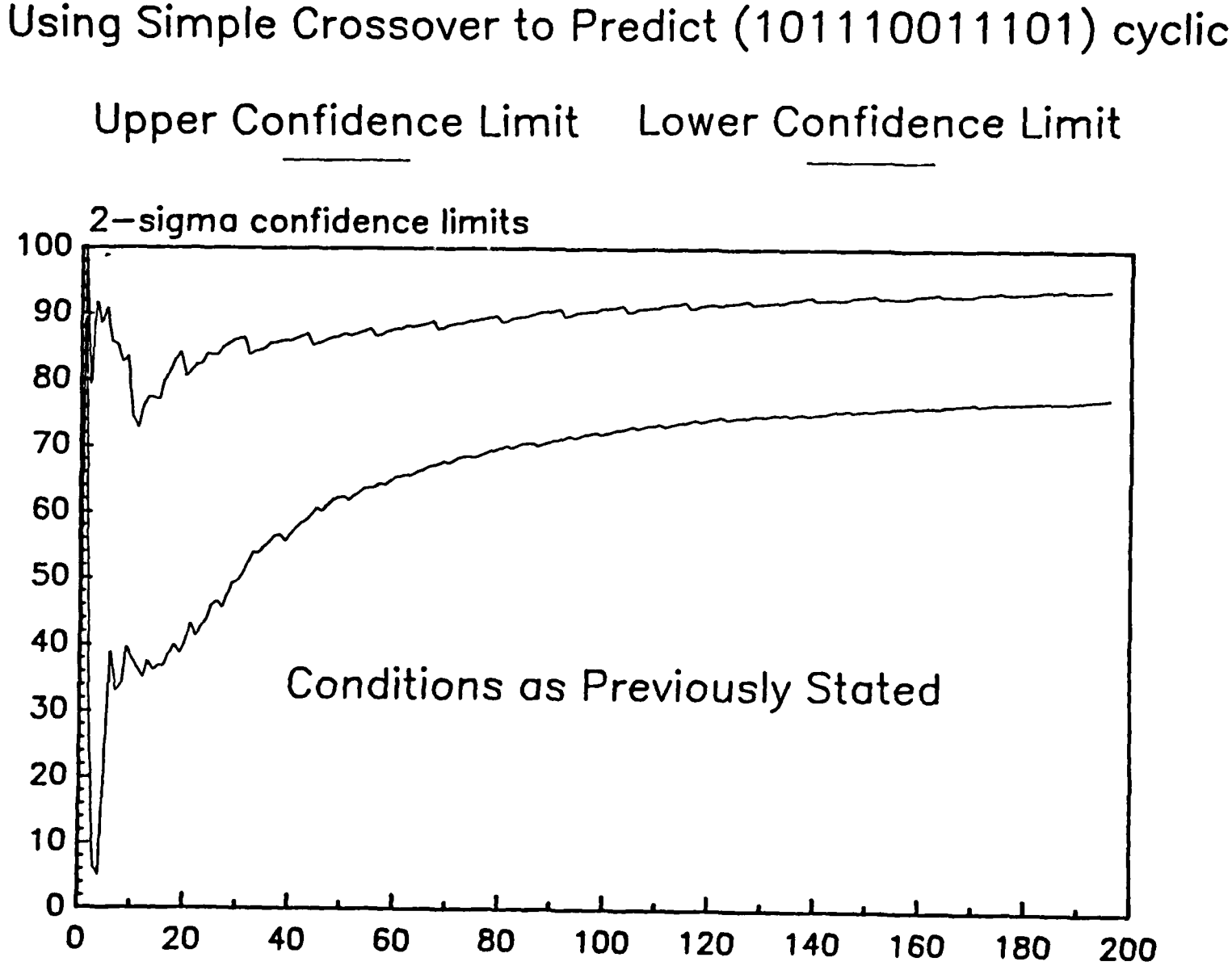
Using Simple Crossover to Predict (101110011101) cyclic
Upper Confidence Limit
Lower Confidence Limit
2−sigma confidence limits
100
90
80
70
60
50
40
30
20
10
0
0
20
40
60
80
100
120
140
160
180
200
Conditions as Previously Stated

Figure 71

# Using Simple Crossover to Predict (101001000...)

## Mean of 20 Experiments

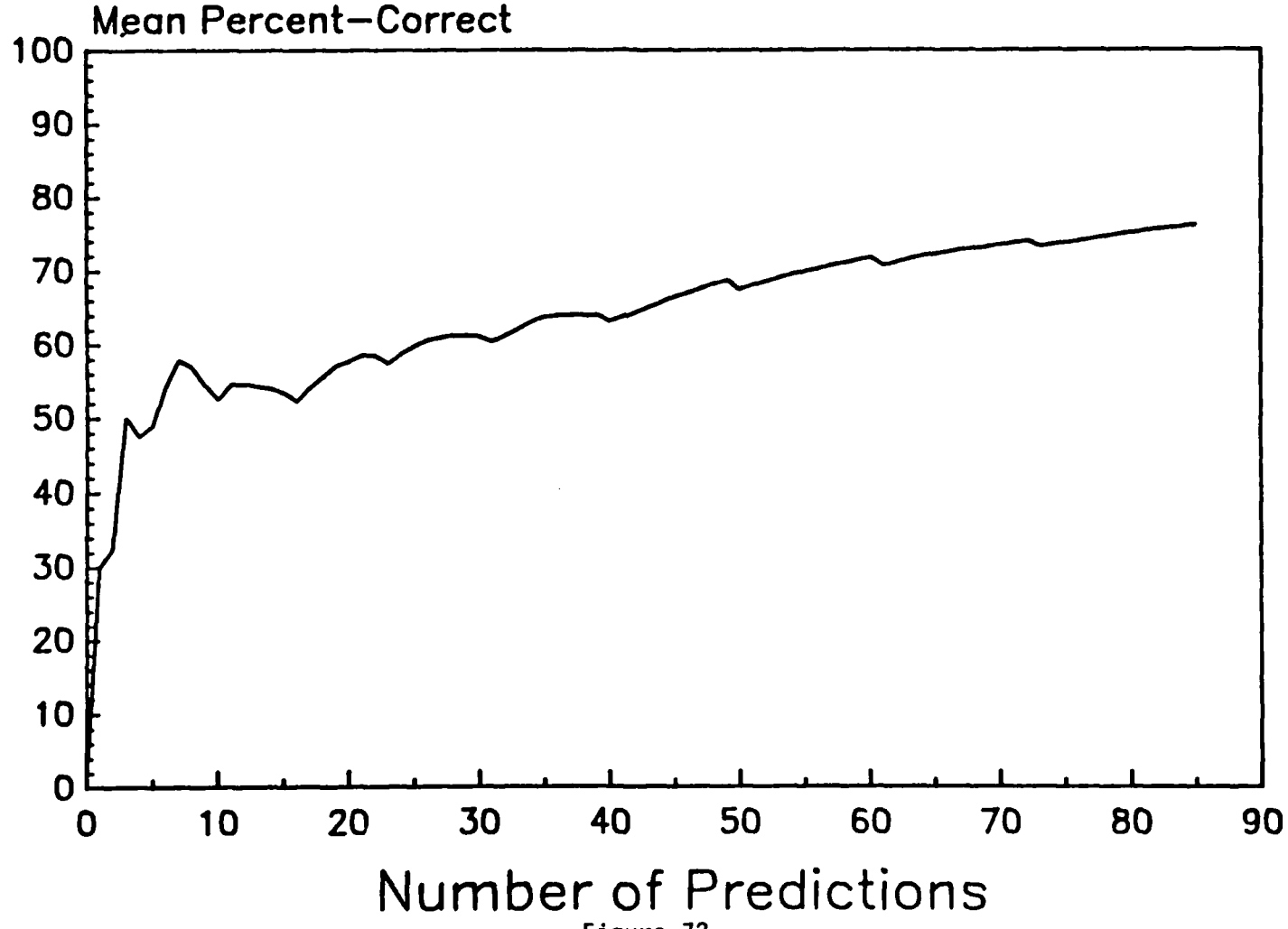


Figure 72

# Using Simple Crossover to Predict (101001000...)

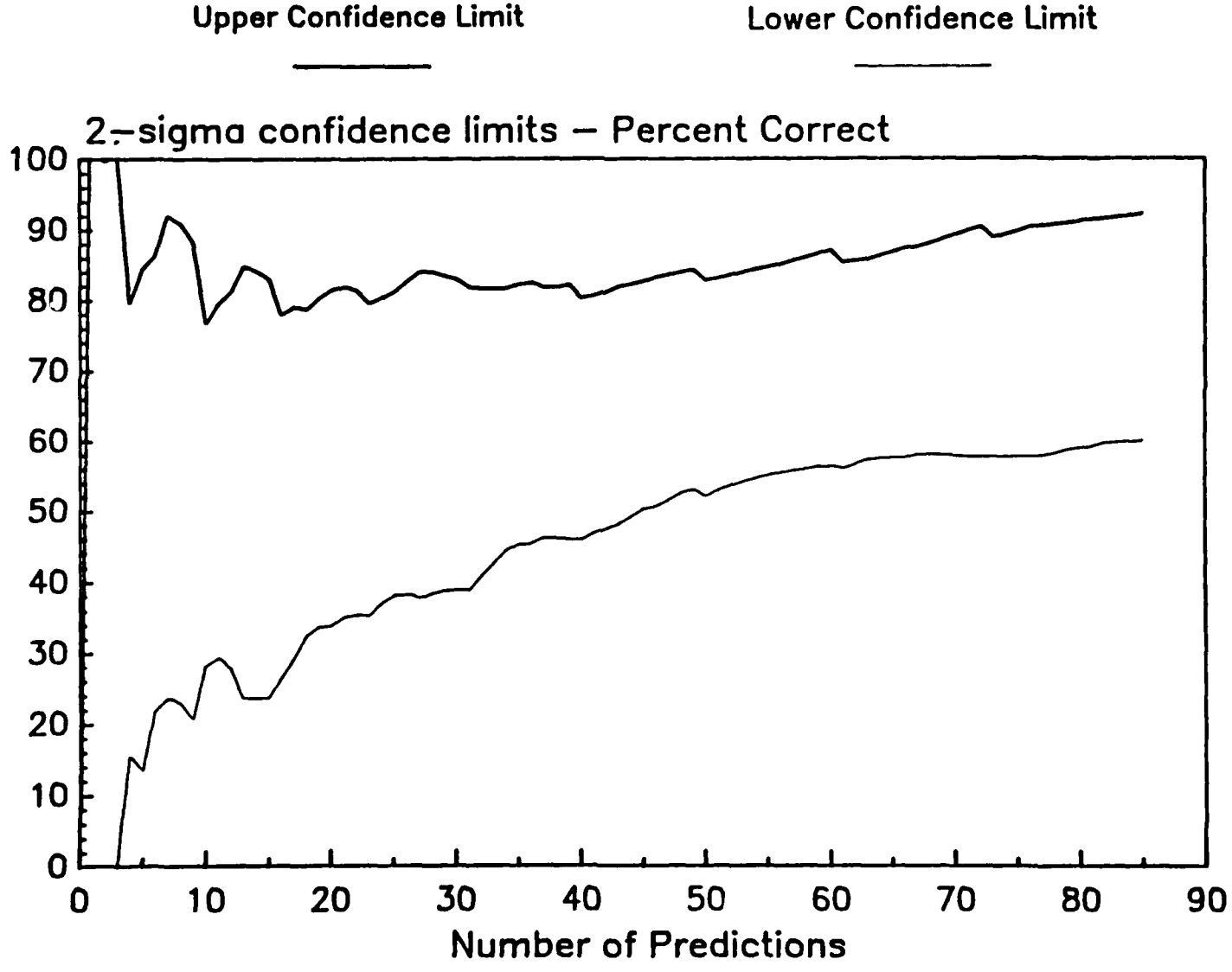


Figure 73

# Using Simple Crossover to Predict (101001000...)

## Mean Worth of 20 Experiments

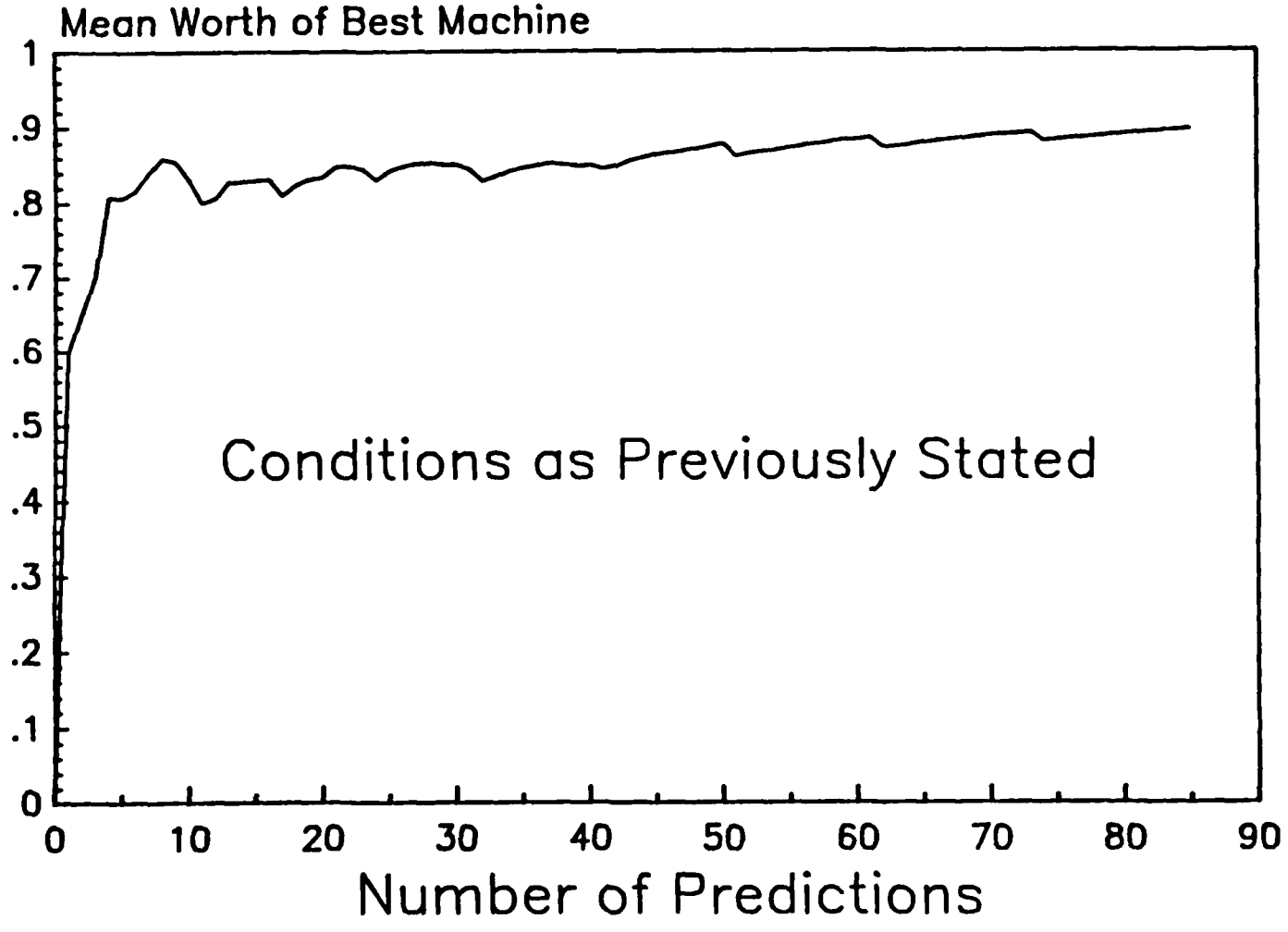


Figure 74

# Using Simple Crossover to Predict (101001000...)

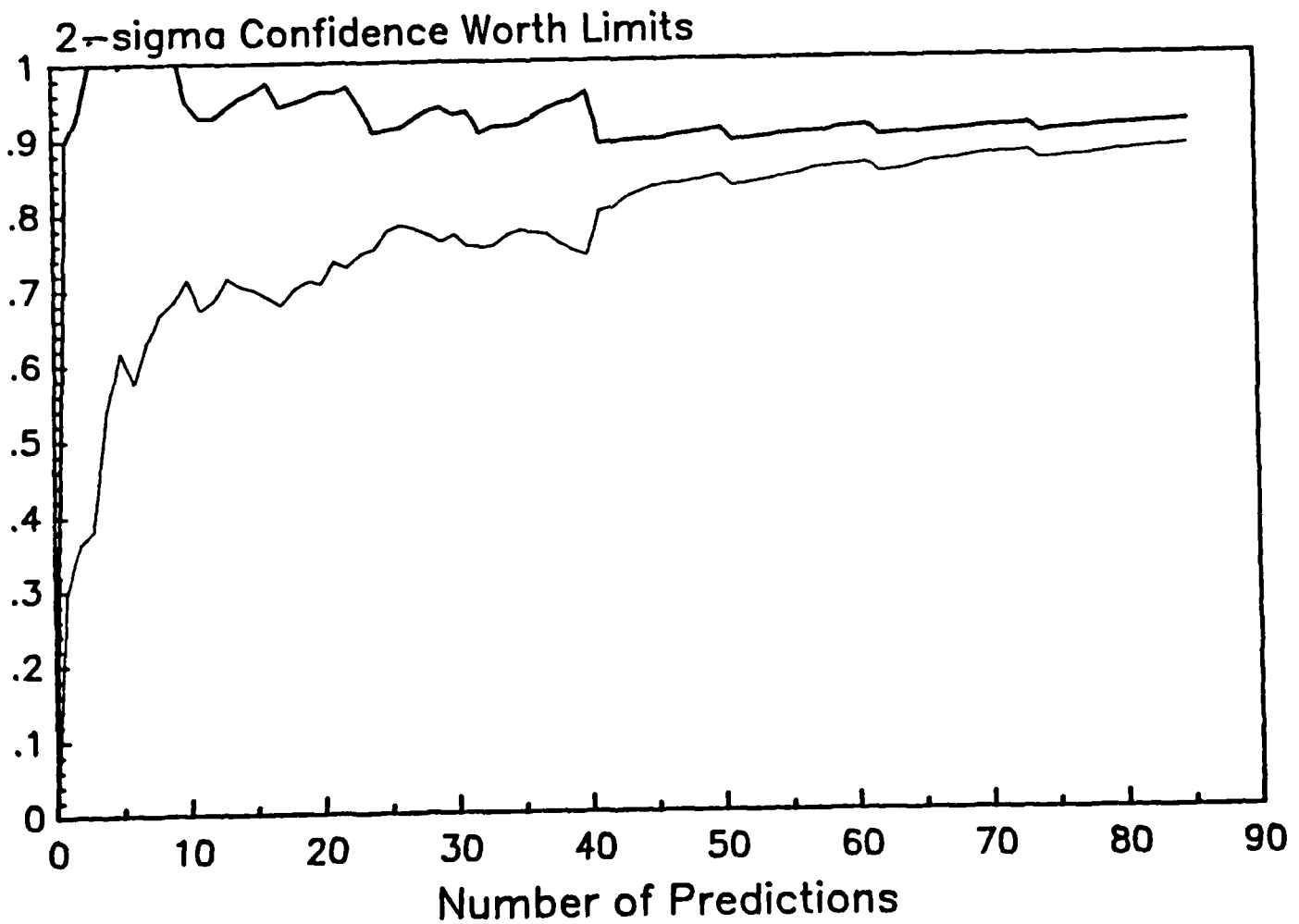


Figure 75

# Using Simple Crossover to Predict (101001000...)

## Mean of 18 Experiments

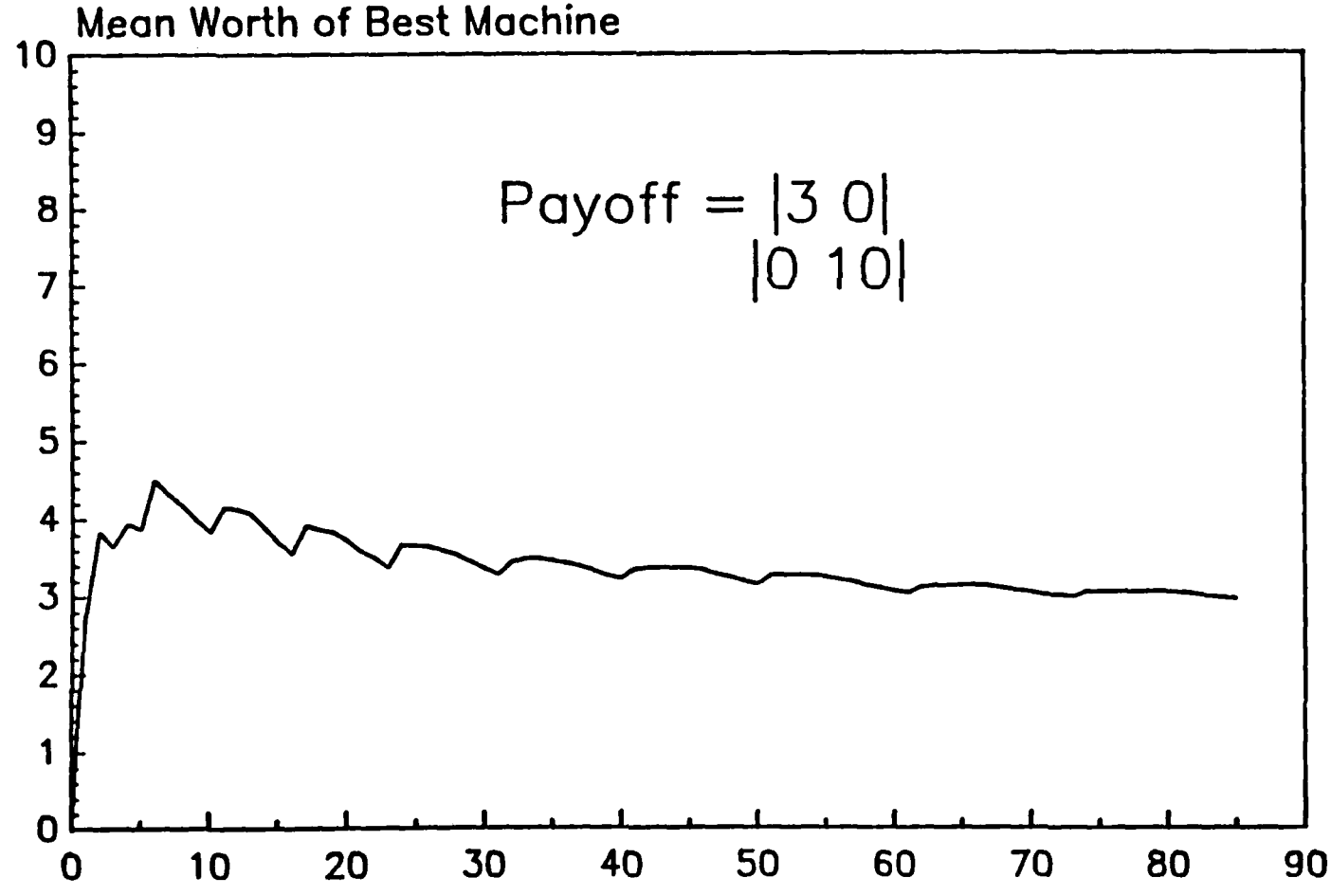


Figure 76

# Using Simple Crossover to Predict (101001000...)

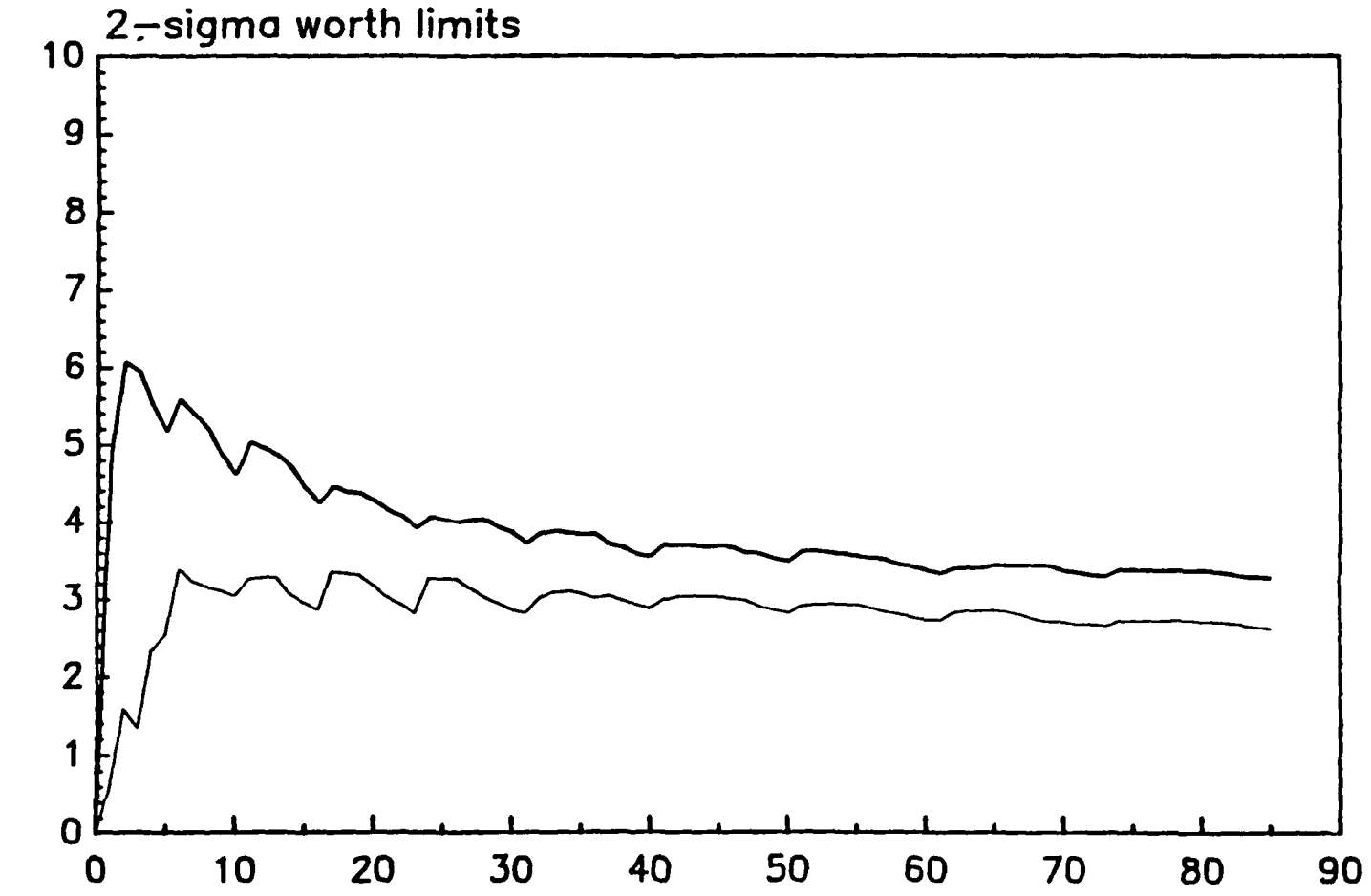


Figure 77

# Predicting (101001000...) without Crossover
## Mean of 16 Experiments

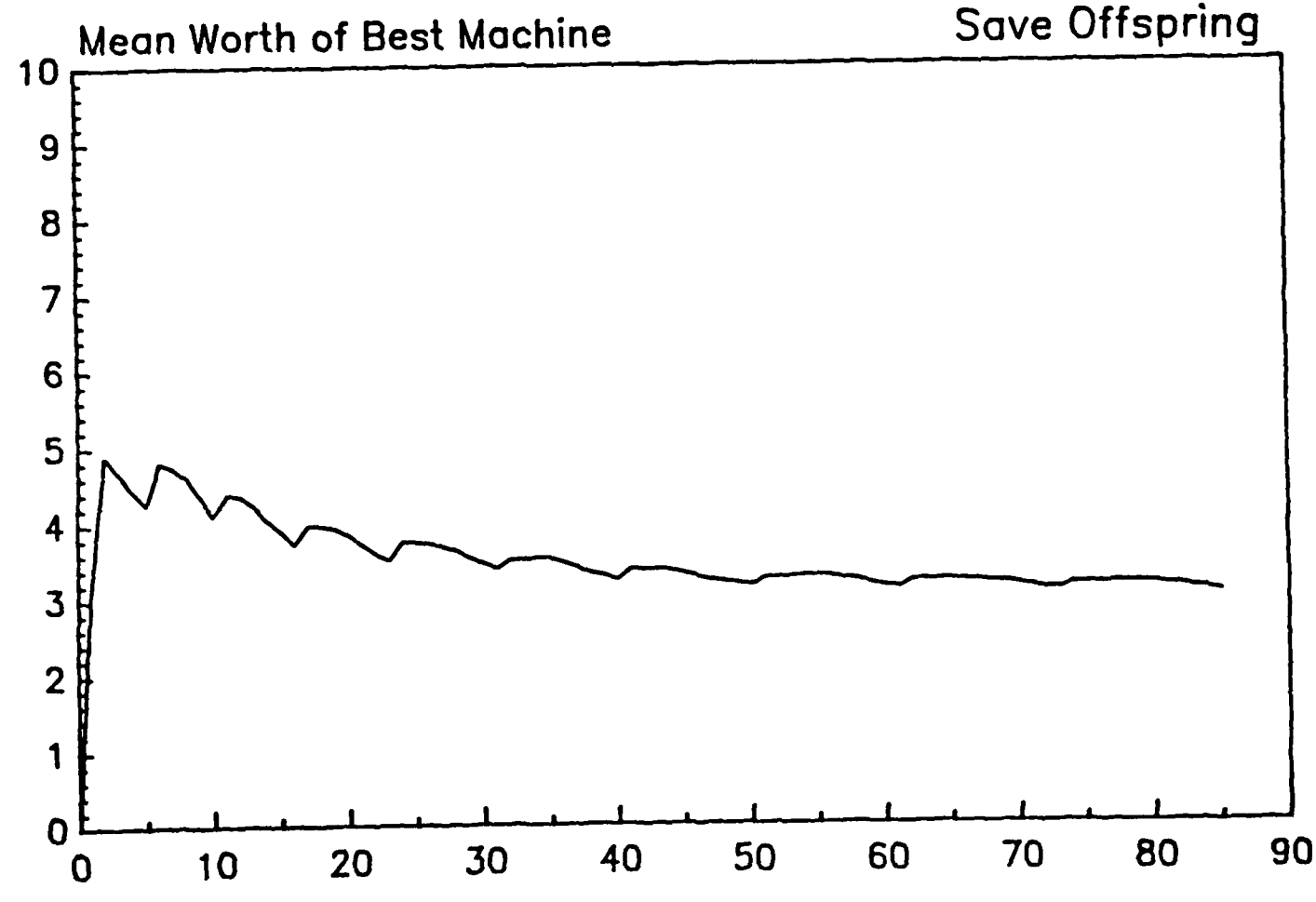


Figure 78



Figure 78

# Predicting (101001000...) without Crossover

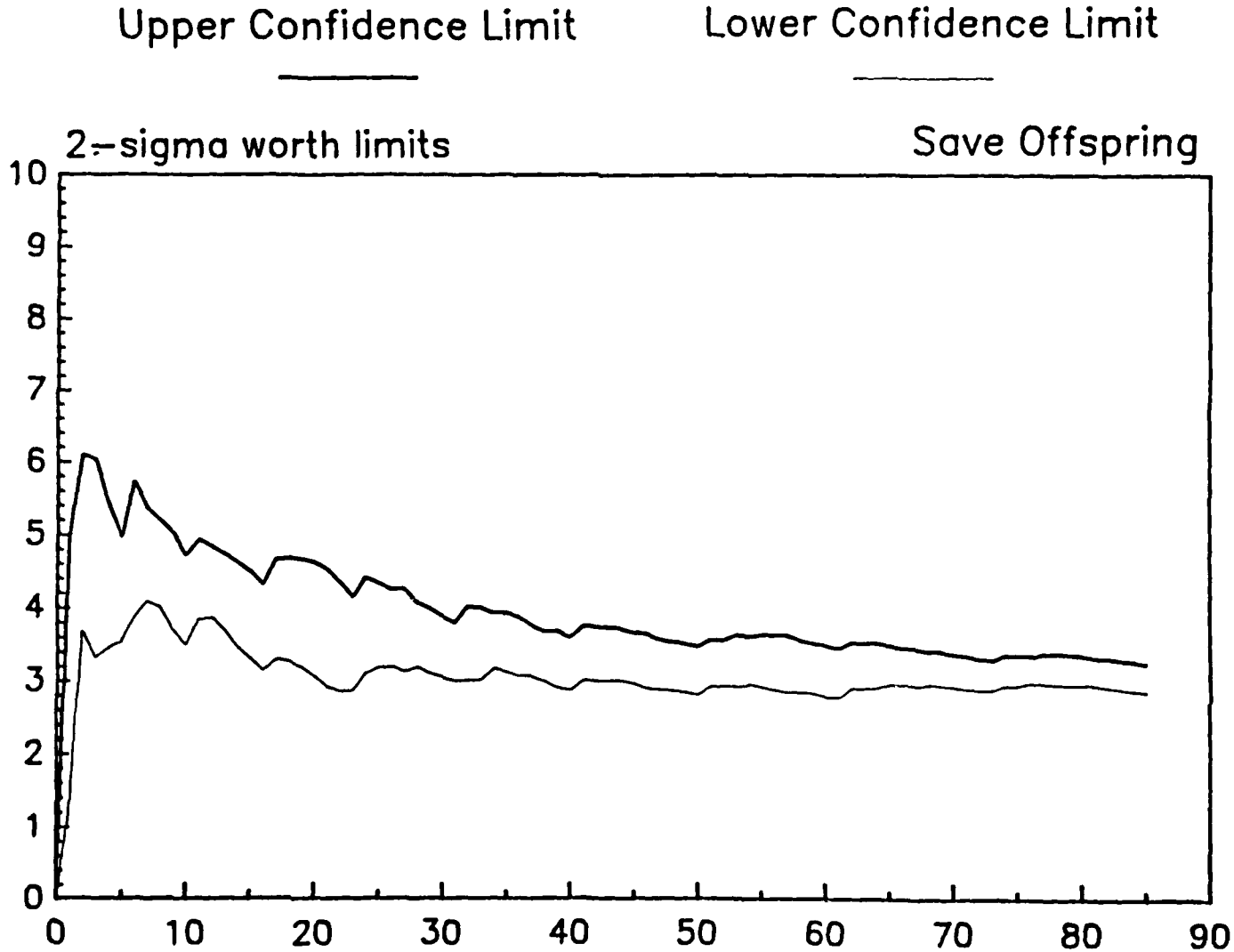


Figure 79

As pointed out by Dr. Wirt Atmar, Holland's crossover technique does not resemble the manner in which recombination occurs in nature. Sexuality in natural organisms results in the recombination of alleles. Alleles are defined as one of two or more alternative genes that control the same characteristic and occupy the same place on similar chromosomes. For Holland's technique to be effective, it must be assumed that a finite state machine is analogous to a chromosome and that each state output and state transition is analogous to a gene. But, this is not correct. Genes contain the instructions for generating specific physical functions. They are much like subroutines. Here a finite state machine could be thought of as a gene. However, the structure of a finite state machine could never be thought of as a gene.

As pointed out in Elements of Biology by Charles K. Levy, Addison Wesley Publisher, Reading, Massachusetts, 1982, changes by crossover are not true mutations "since neither the amount nor the function of genetic material is altered." Under Holland's crossover, while the amount of coding is not altered, the function of the coding is greatly altered. The result of this is a near-random search throughout the entire space of possible solutions. As the size of the machines grow larger, the number of states used in corssover increases. As shown in Figures 80 through 85, when predicting the cyclic environment 012345676532210, without loss of generality, the evolutionary processes which do not use crossover perform significantly better than those using two-state and ten-state crossover. Holland's technique effectively destroys the link between parent and offspring that is necessary for an evolving process to succeed. In the words of Dr. Atmar, "While his original thought was in the right direction, the technique he promotes is at too severe a scale. True sexuality does not destroy (nor create) information; it only shuffles contending subroutines, eventually trying almost every probable combination of those in existence, retaining only those combinations found to be most beneficial.

Even at this far milder level of informational reorganization, evaluating the "costs" of sexual recombination in producing inappropriately behaving

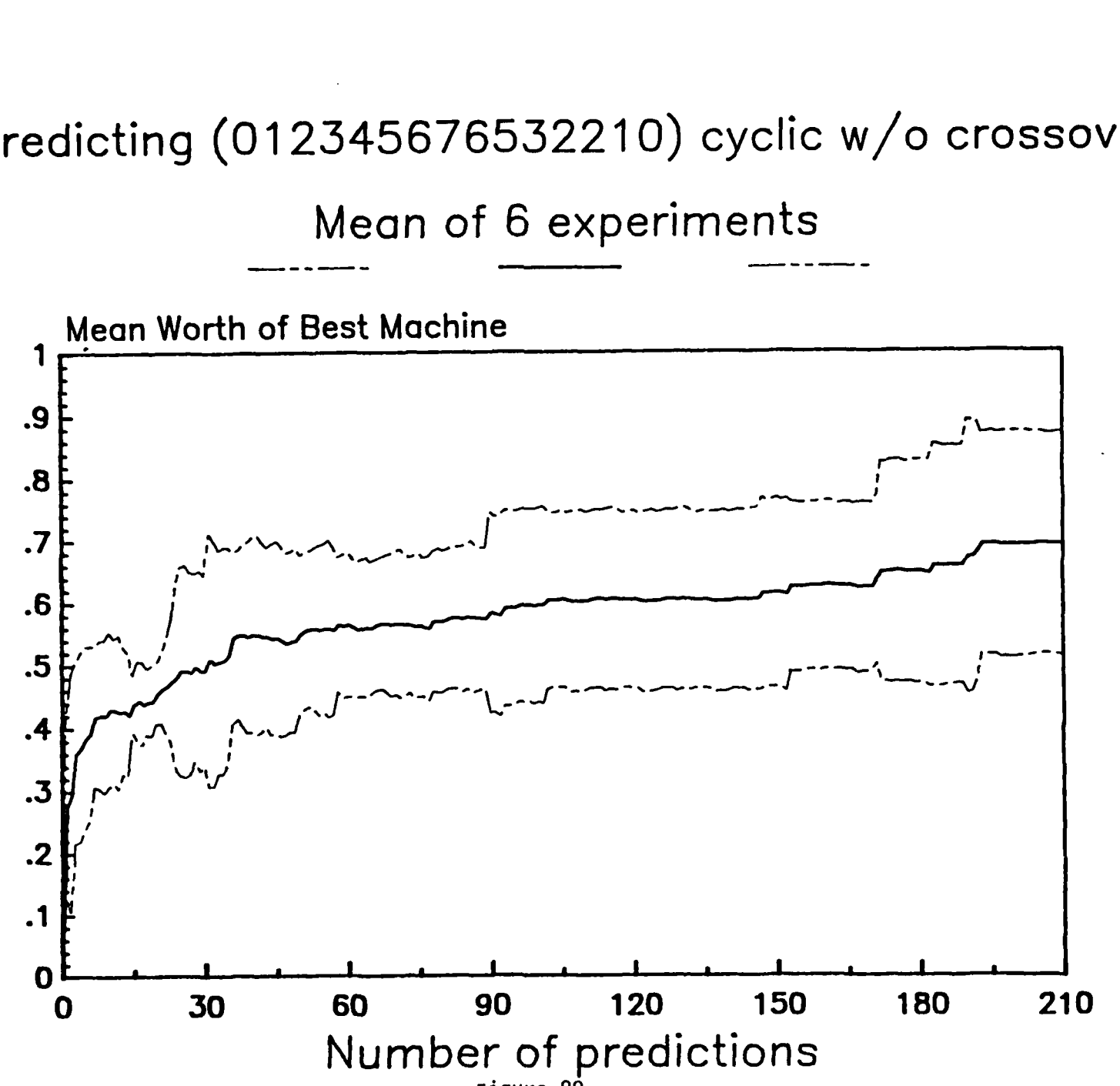
Predicting (012345676532210) cyclic w/o crossover
Mean of 6 experiments
Mean Worth of Best Machine
1
.9
.8
.7
.6
.5
.4
.3
.2
.1
0
0
30
60
90
120
150
180
210
Number of predictions

Figure 80

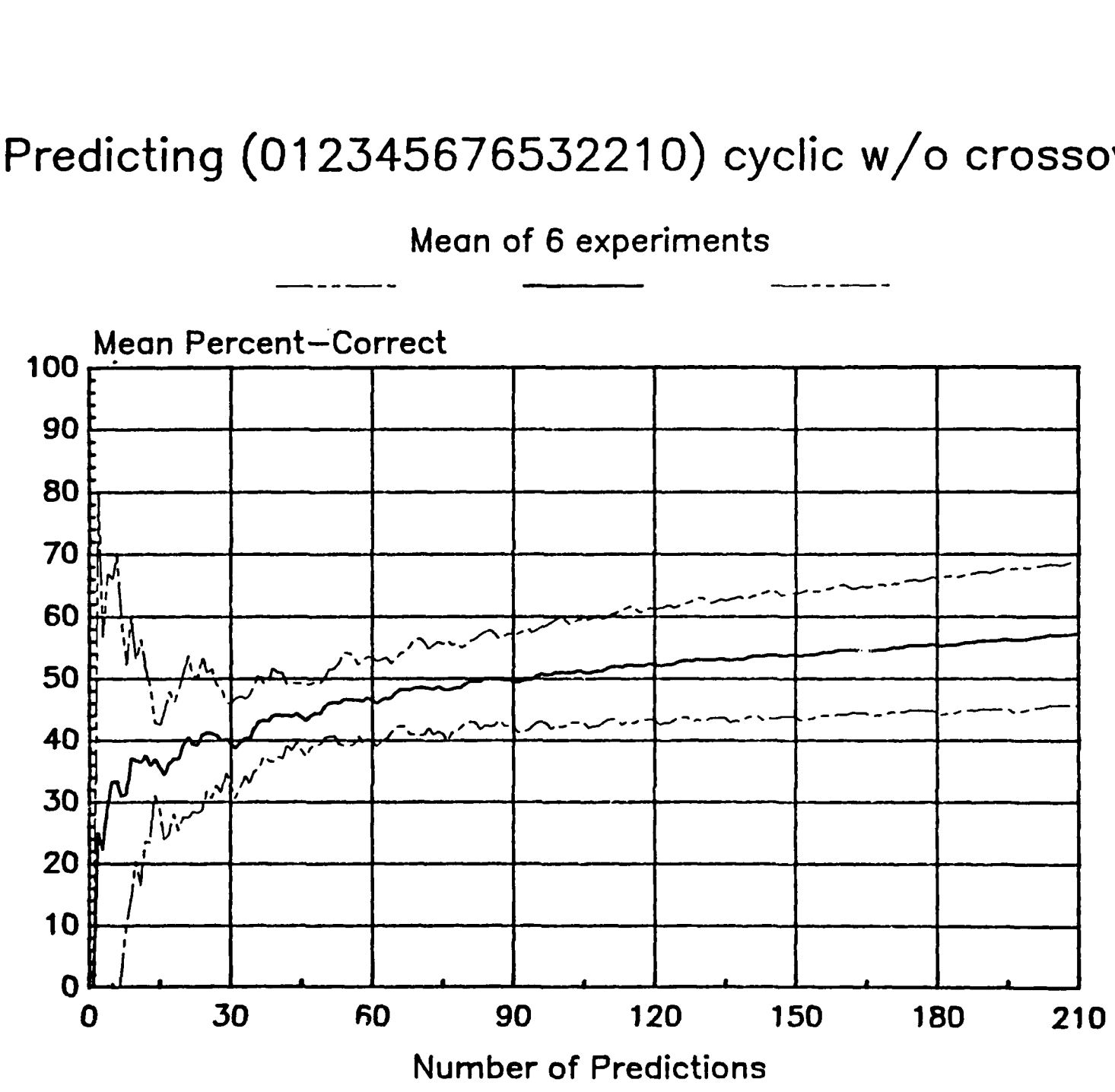
Predicting (012345676532210) cyclic w/o crossover
Mean of 6 experiments
Mean Percent—Correct
100
90
80
70
60
50
40
30
20
10
0
0
30
60
90
120
150
180
210
Number of Predictions

Figure 81

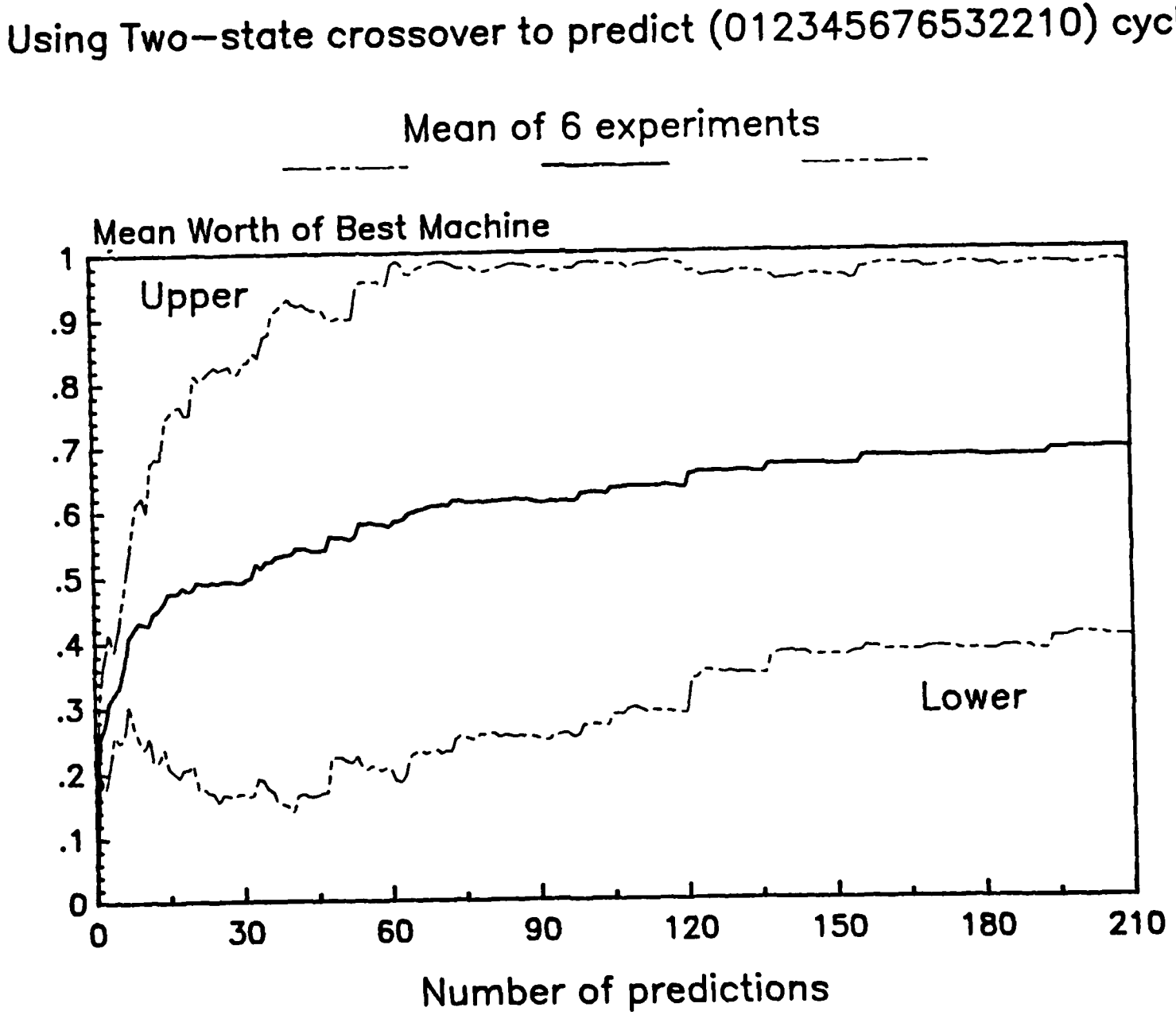
Using Two-state crossover to predict (012345676532210) cyclic
Mean of 6 experiments
Mean Worth of Best Machine
Upper
Lower
1
.9
.8
.7
.6
.5
.4
.3
.2
.1
0
0
30
60
90
120
150
180
210
Number of predictions

Figure 82

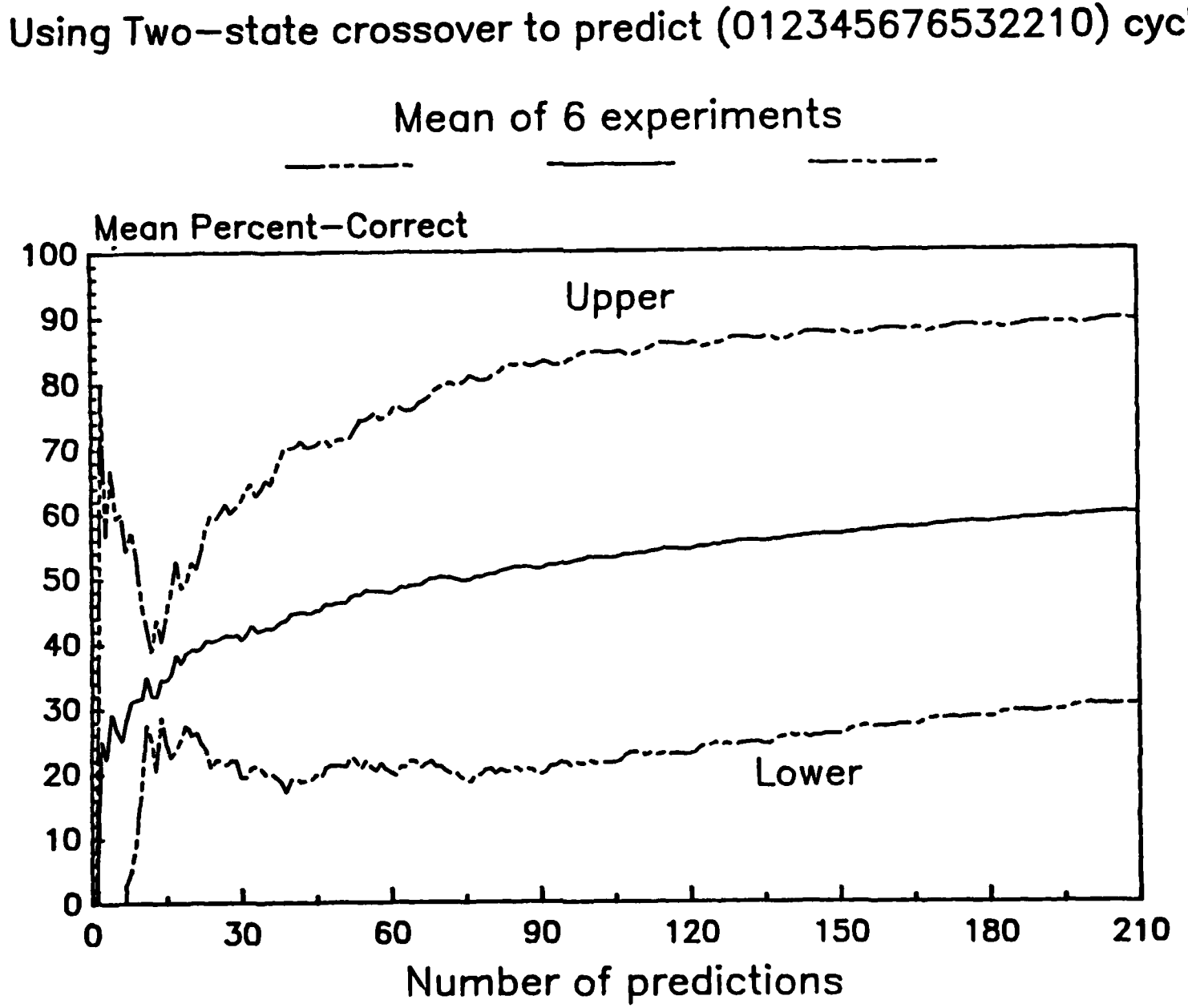
Using Two–state crossover to predict (012345676532210) cyclic
Mean of 6 experiments
Mean Percent–Correct
100
90
80
70
60
50
40
30
20
10
0
Upper
Lower
0
30
60
90
120
150
180
210
Number of predictions

Figure 83

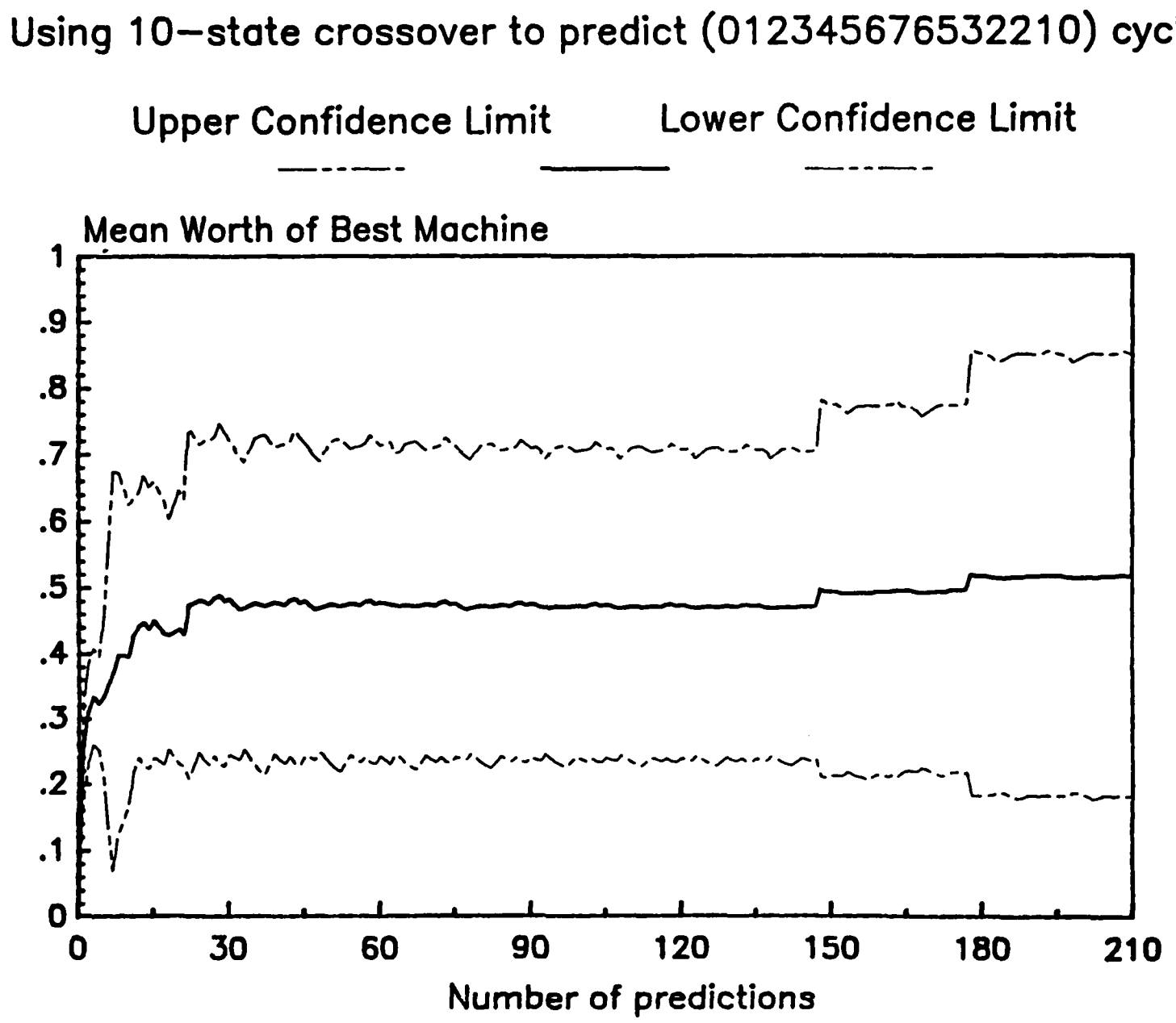
Using 10–state crossover to predict (012345676532210) cyclic
Upper Confidence Limit
Lower Confidence Limit
Mean Worth of Best Machine
1
.9
.8
.7
.6
.5
.4
.3
.2
.1
0
0
30
60
90
120
150
180
210
Number of predictions

Figure 84

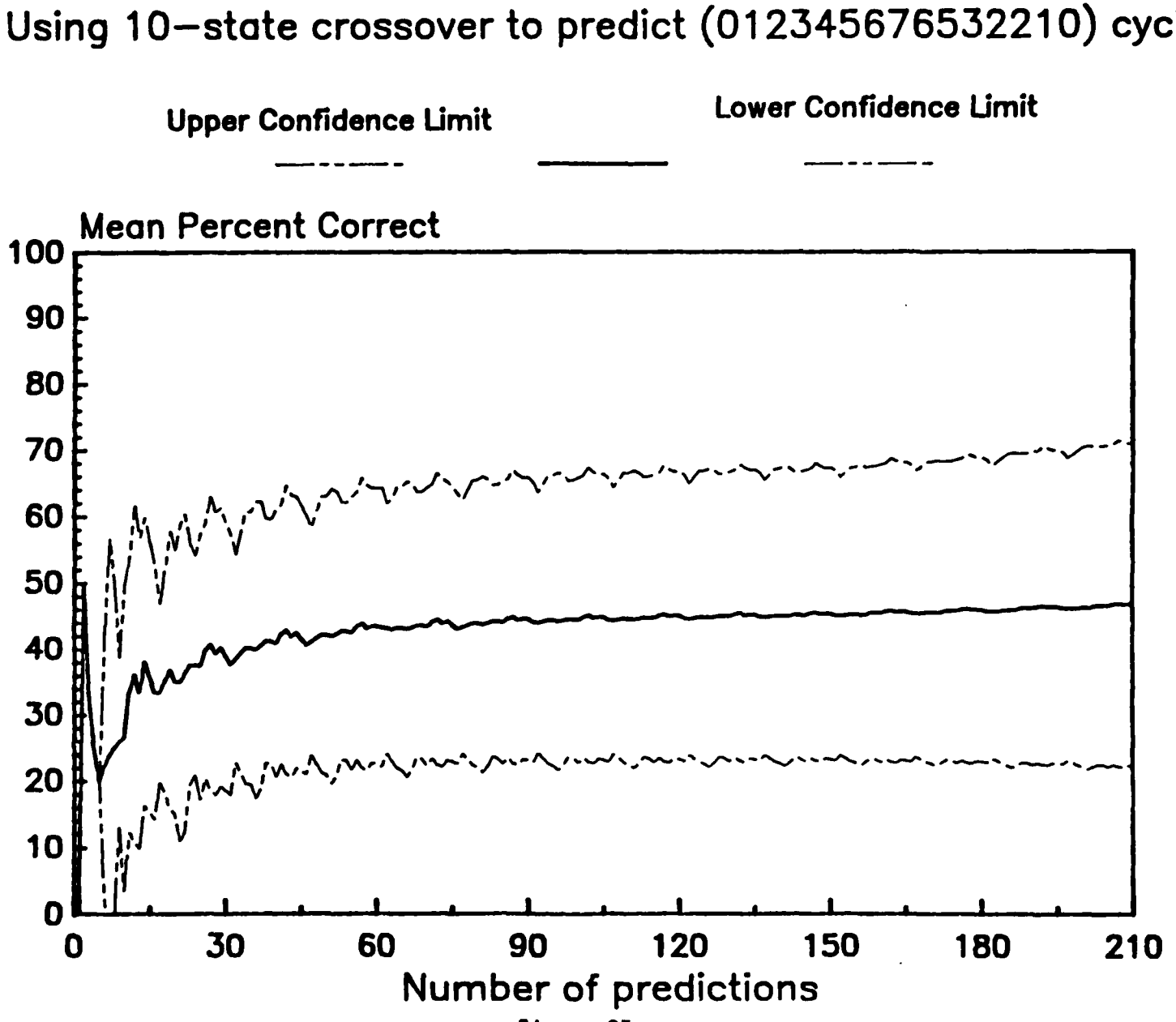
Using 10–state crossover to predict (012345676532210) cyclic
Upper Confidence Limit
Lower Confidence Limit
Mean Percent Correct
100
90
80
70
60
50
40
30
20
10
0
0
30
60
90
120
150
180
210
Number of predictions

Figure 85

organisms has been a topic of substantial debate which has continued on in biology for some time now. The debate is currently being resolved in the following manner: the costs of sexual recombination are no longer considered severe because (1) the contending alleles (subroutines) have generally been found to be quite similar, and (2) there are not nearly as manny alleles contending for each site as was originally thought or predicted.

Sexuality is not a mutational phenomenon. It is, rather, a method of rapidly sorting through the effects of mutated subroutines. The advantages of sexuality in an evolutionary optimization program are not to be underestimated. However, a very specific structure must be in place before these advantages are obtainable."

## Identification Experiments

As previously indicated, prediction is key to the process of identification. Suppose an unknown entity is responding to known stimuli in a clearly observable manner. The task is to develop an explicit representation for that transduction, this on the basis of the stimulus/response experienced to date. The predictor observes the sequence of stimulus/response pairs, the most recent stimulus, then predicts the next response... this process being repeated as each new stimulus is observed. The predictor must develop a most suitable logic in terms of the specific payoff function and span of prediction. If this span is the single time unit, the discovered predictive regularity should correspond with the underlying logic of the transducer. Evolutionary programming provides a basis for such prediction and identification.

To demonstrate this hypothesis, a series of pilot experiments were conducted. Success in this regard might justify more formidable experiments on the Cray computer. As a first step, the transducer or plant is

deterministic and fully controllable, that is, each of the possible response symbols is generated by each of the states. Plants of increasing complexity in terms of alphabet size and number of states were used. Once a searching prediction score has been obtained, the predictive logic is compared to the actual transduction as a test of the identification. In these experiments, an all-or-none payoff function was used in that all symbols are equally important, and no credit was to be given for nearness. To ensure adequate exploration, random input sequences were used to drive the plant.

The first experiment required the identification of a two-symbol, two-state finite state machine, see Figure 86. The evolutionary process properly identified this finite state machine at the second prediction, this after evaluating only thirty offspring.

The next experiment required identification of a two-symbol, five-state machine, see Figure 87. Here, in 254 predictions, the evolutionary process correctly predicted 80 percent of the responses and had a predictive fit score of 90 percent, see Figures 88 and 89. In all, 32,376 offspring were evaluated. Figure 90 indicates the final evolved machine consisting of six states. It is tempting to compare this machine with the one shown in Figure 87, but this is dangerous. The addition of even a single state changes the meaning of all other states. There can be no simplistic state-to-state comparison. Note that a fully controllable finite state machine driven by random input in a binary alphabet has a fifty percent chance of responding with either of the alphabet symbols. The predictive score should be fifty percent. Clearly, the evolutionary process does significantly better.

In the next experiment, a two-symbol, ten-state machine was the unknown transducer, see Figure 91. Here, the evolutionary process was able to achieve a high predictive fit score, see Figure 92, and predicted extremely well, see Figure 93. The final machine was of fifteen states,

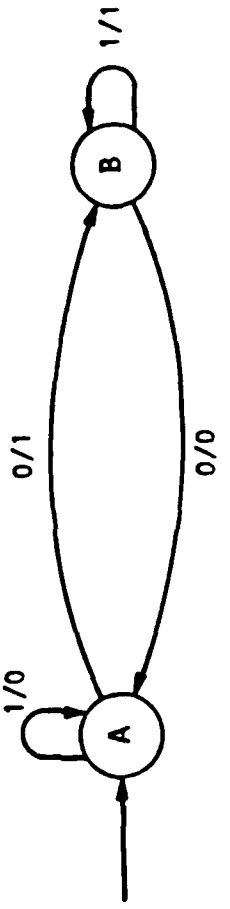


Figure 86

Figure 86

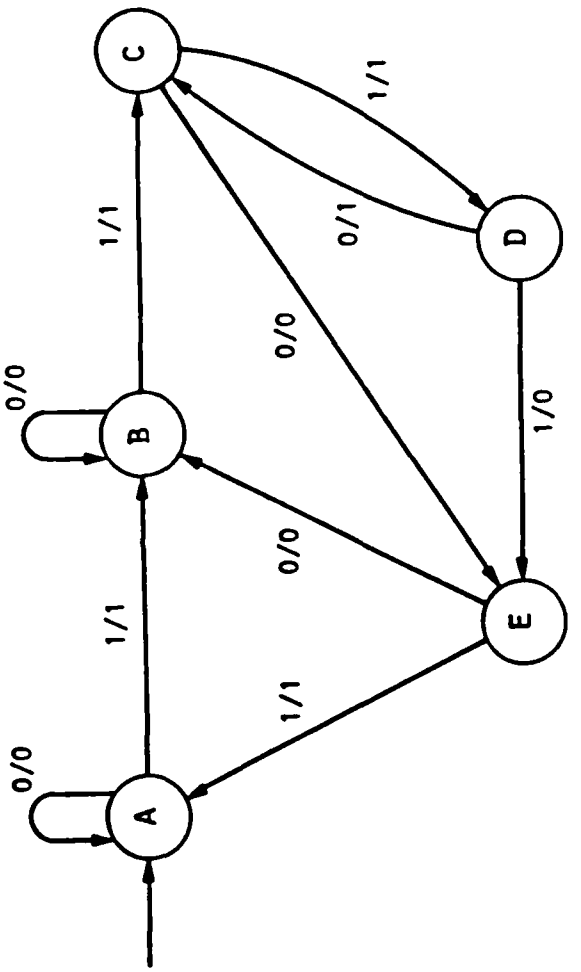


Figure 87

Figure 87

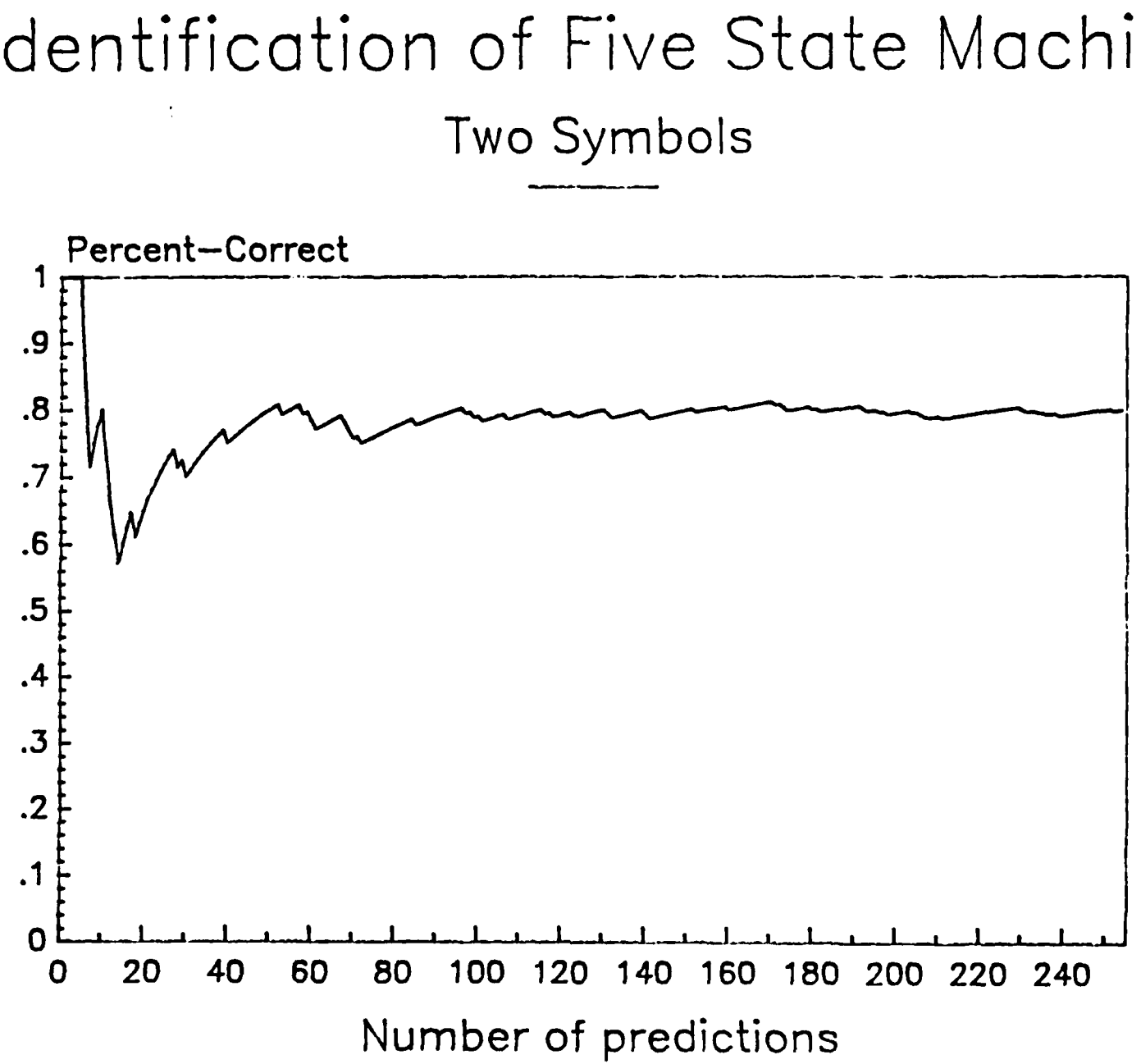


Figure 88

Figure 88

# Identification of a Five State Machine

## Two Symbols

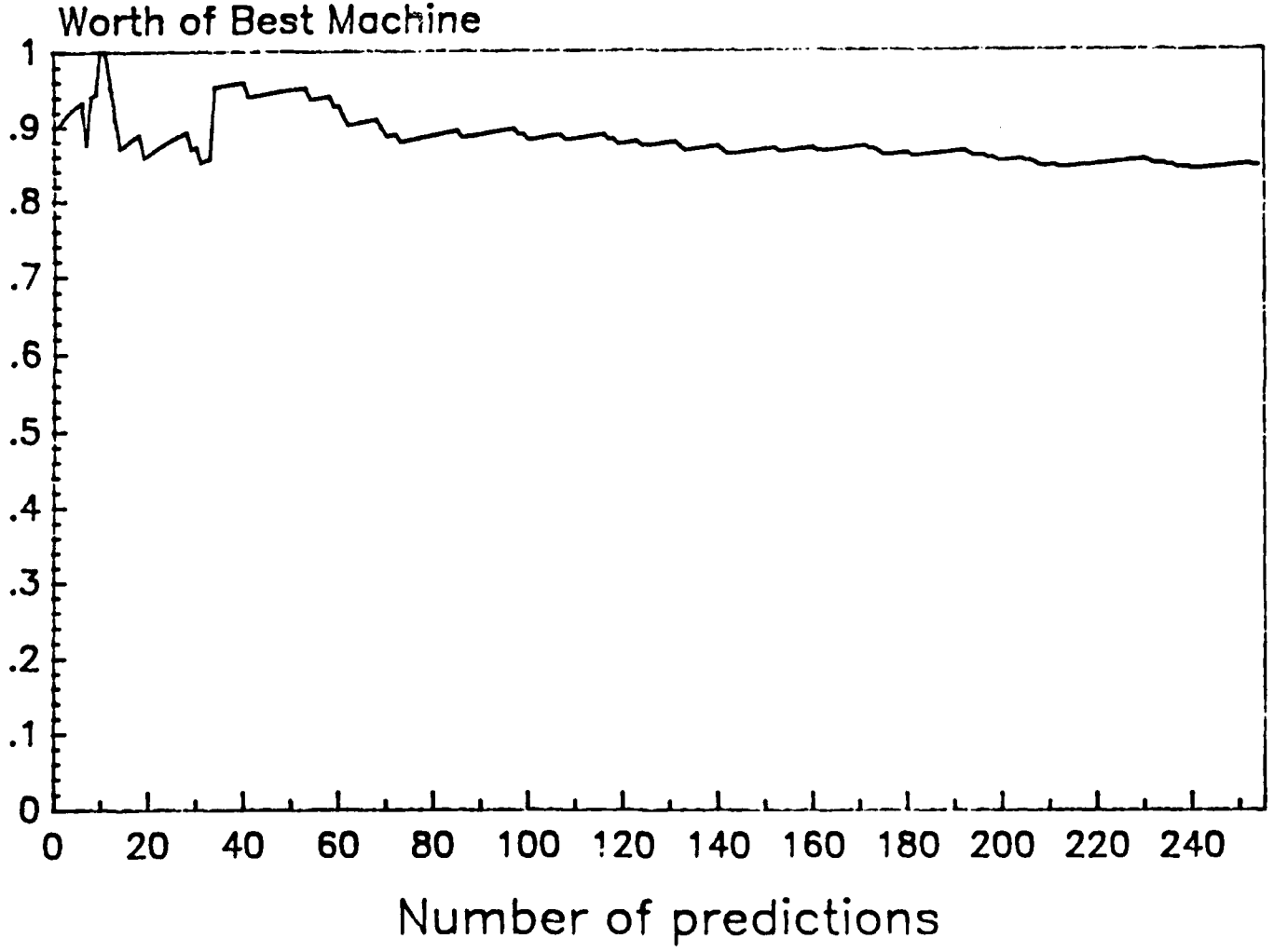


Figure 89



Figure 89

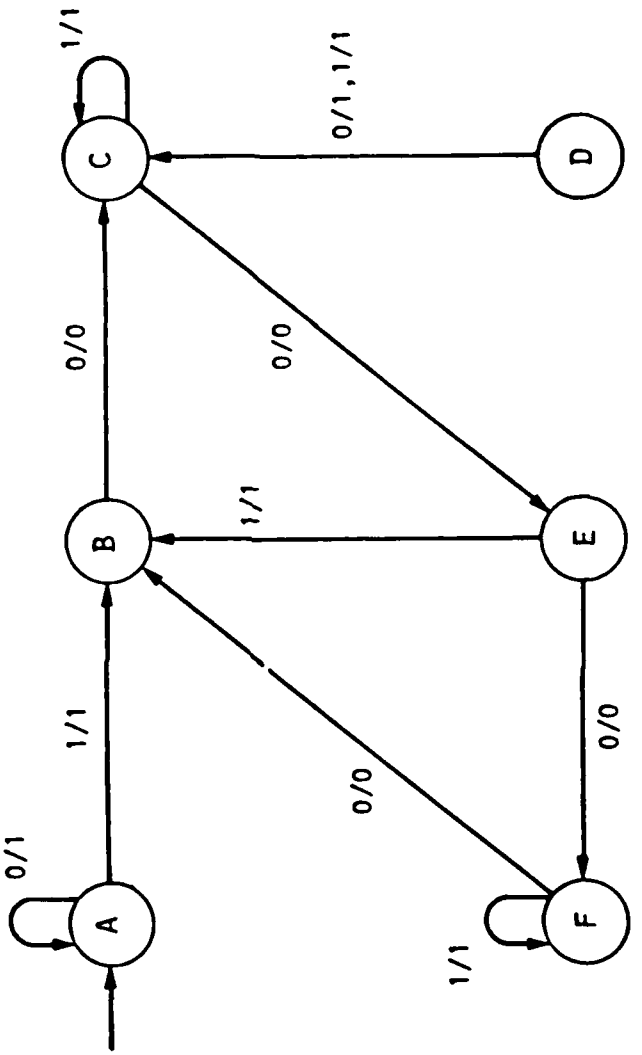


Figure 90

Figure 90

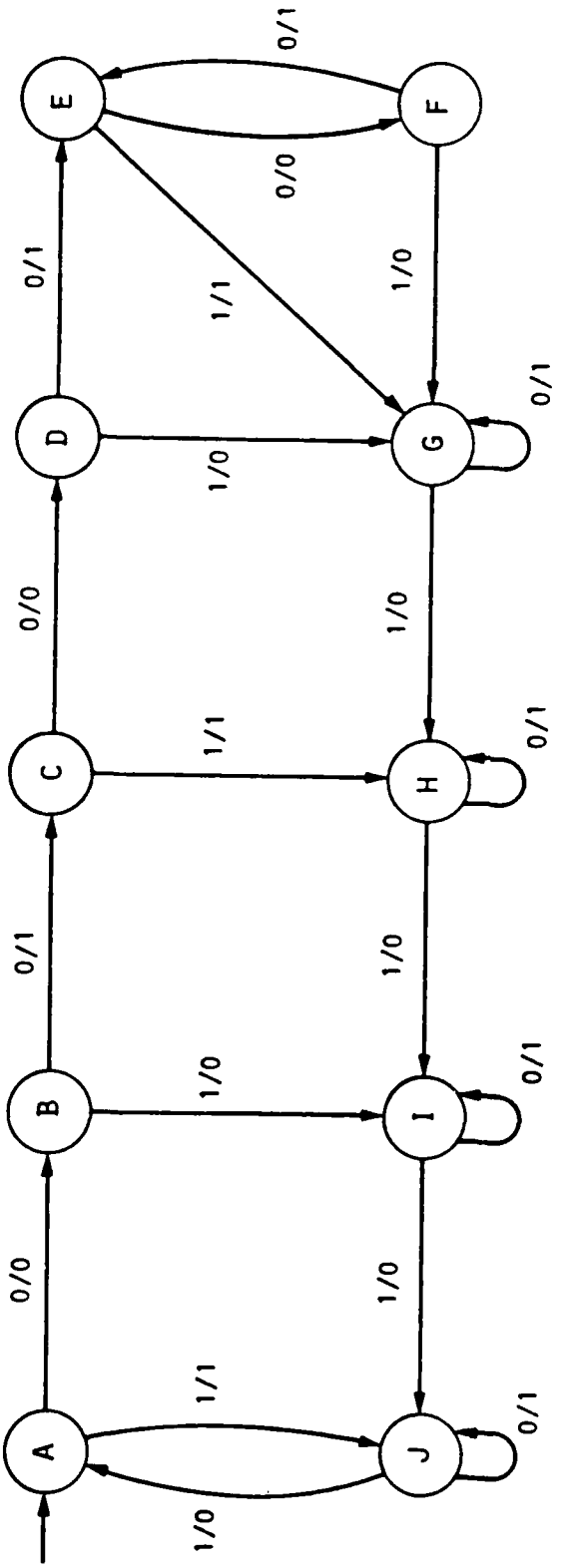


Figure 91

Figure 91

# Identification of a FSM of 10 states

## Binary Alphabet

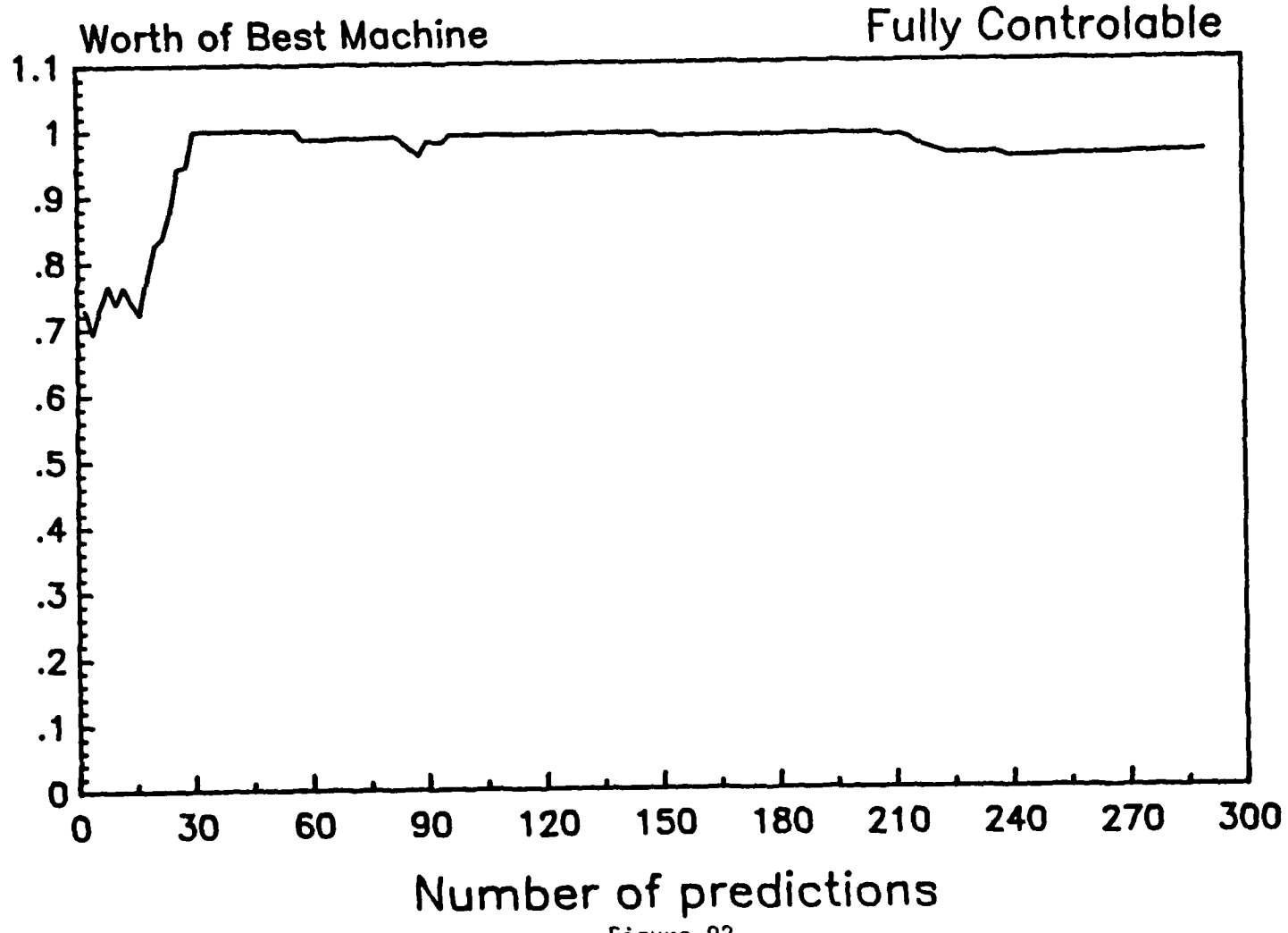


Figure 92



Figure 92

# Identification of a FSM of 10 states

## Binary Alphabet

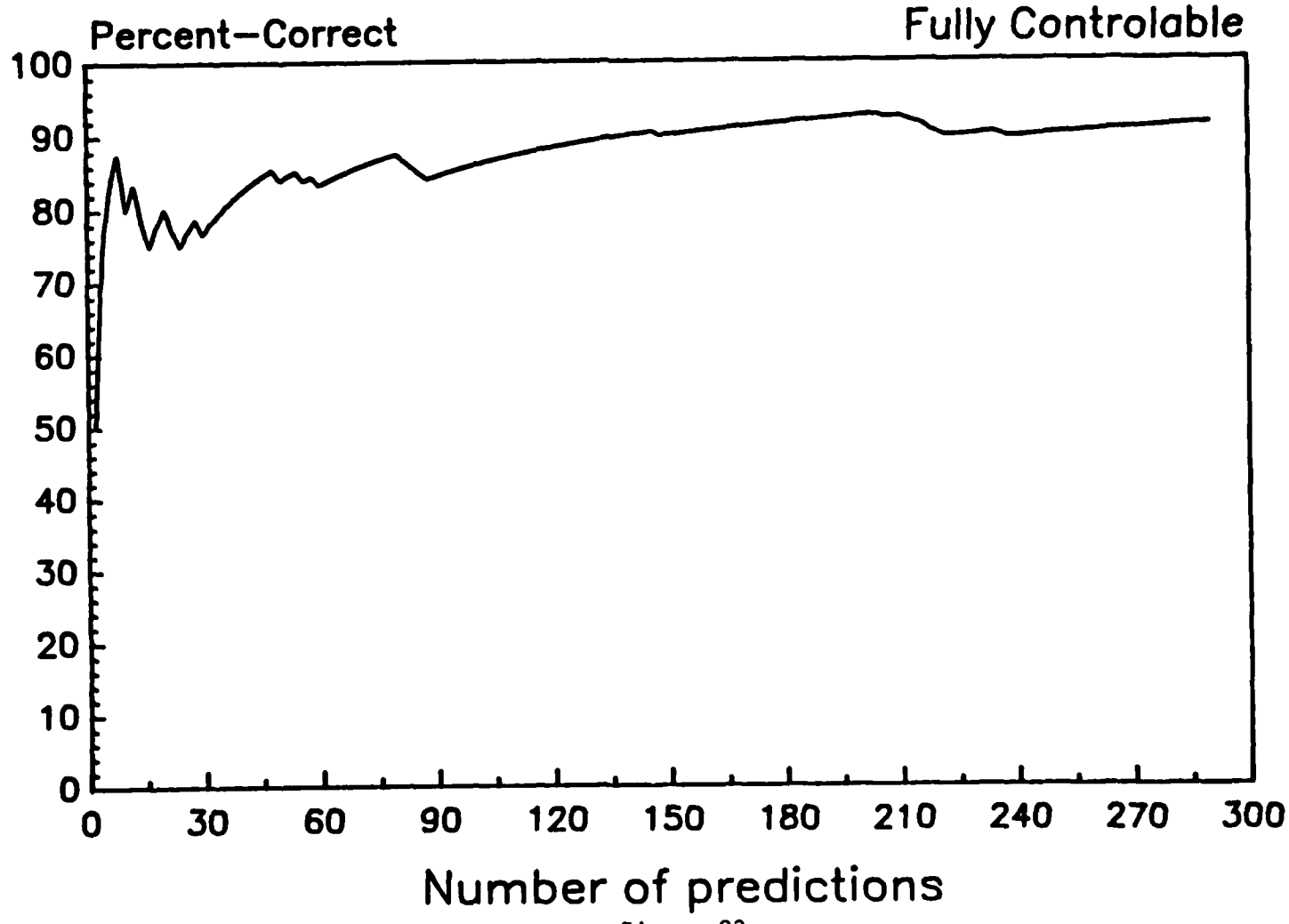


Figure 93

while evaluating 8,040 offspring in the 290 predictions. Note that increased complexity of behavior may not be simply determined by the number of states. Here the plant has a relatively simple structure which was evidently discovered by the evolutionary programming. Presumably, further evolution would finally discover the computer logic of this machine.

Next, a four-symbol alphabet was used driving a five-state machine that represented the unknown transducer, see Figure 94. Four experiments were conducted; the percentage correct prediction ranges from 36 to 50, but in every case, this is better than the expected 25 percent prediction by zeroth order statistics. Figures 95 to 101 show the predictive fit score and percent correct for each of these experiments.

An eight state machine was then used as the plant, see Figure 102. Four experiments were conducted. Even though the size of the machine had been increased, the evolutionary process was still able to predict 30 to 40 percent correct, see Figures 103 to 106. It is also of interest to examine the predictive fit score for these experiments, see Figures 107 to 110. The 60 to 70 percent worth indicates that if the evolutionary process had been given more time between predictions, superior response prediction could have been achieved. In each of these experiments, there was a rapid increase in the size of the machines, see Figures 111 to 114, even though equally valued offspring were not being saved. An average of 11,422.5 offspring were evaluated in making the 290 predictions. The final predictive machine of the first experiment consisted of ten states, see Figure 115.

Finally, two experiments were conducted using an eight-symbol, five-state machine as the plant, see Figure 116. Figures 117 and 118 indicate the precent correct predictions to be 20 to 25 percent, this being twice the level that would be expected on the basis of a zeroth order prediction (12.5%). Figures 119 and 120 indicate the predictive fit score for these two experiments. The results range from 45 to 50 percent worth. Evidently, five generations per prediction was insufficient. Greater exploration before predictions would have increased the predictive fit

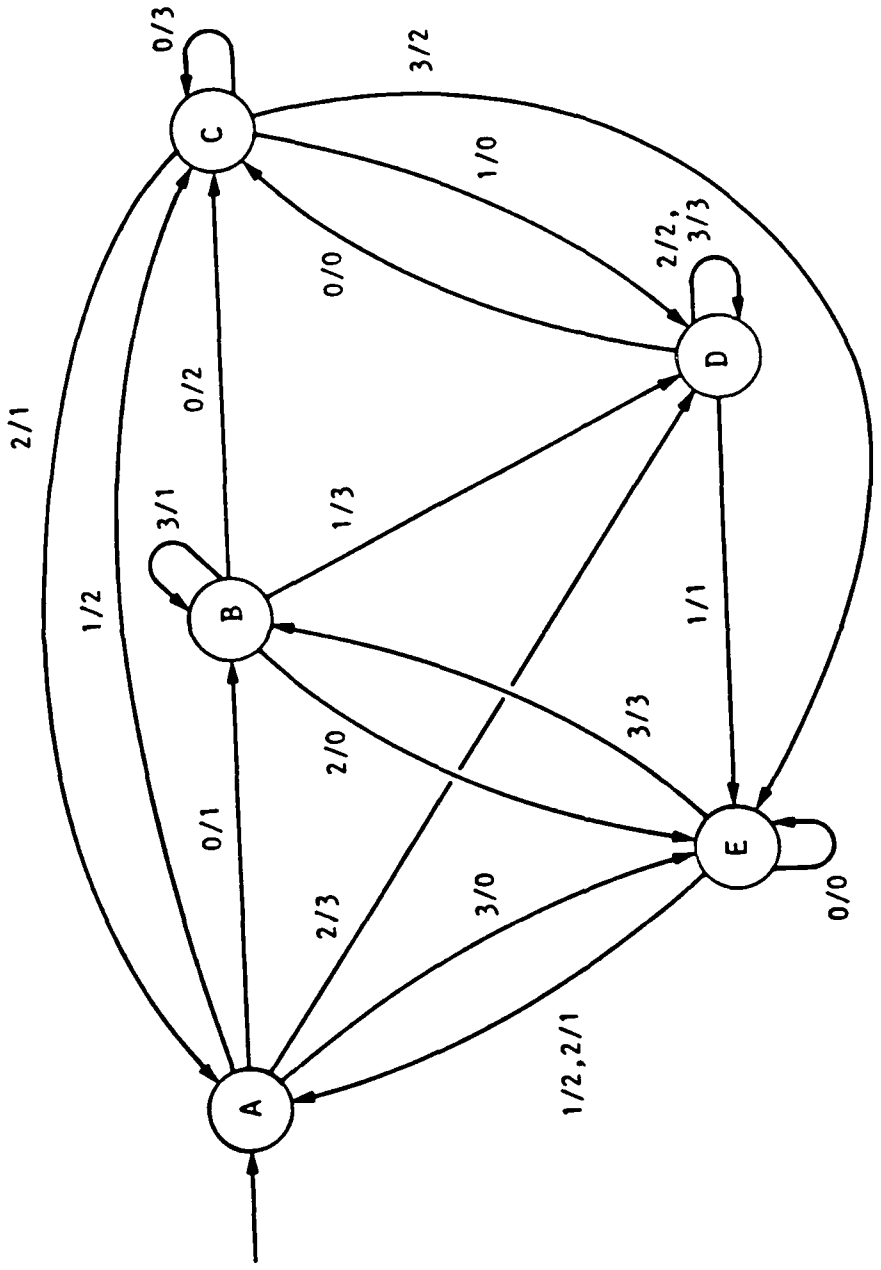


Figure 94

Figure 94

# Identification of a FSM of 5 states

## Four symbol alphabet

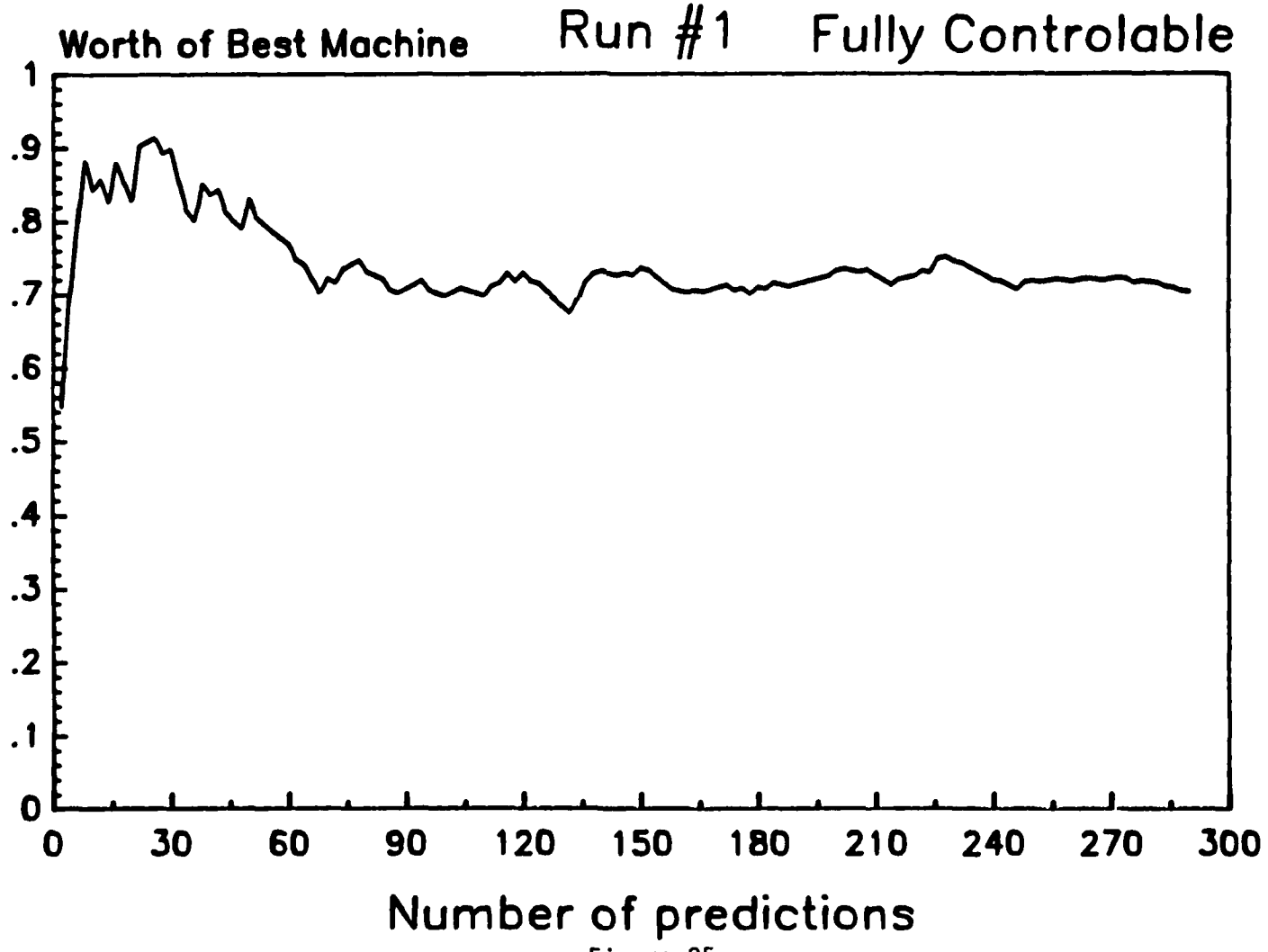


Figure 95

# Identification of a FSM of 5 states

## Four symbol alphabet

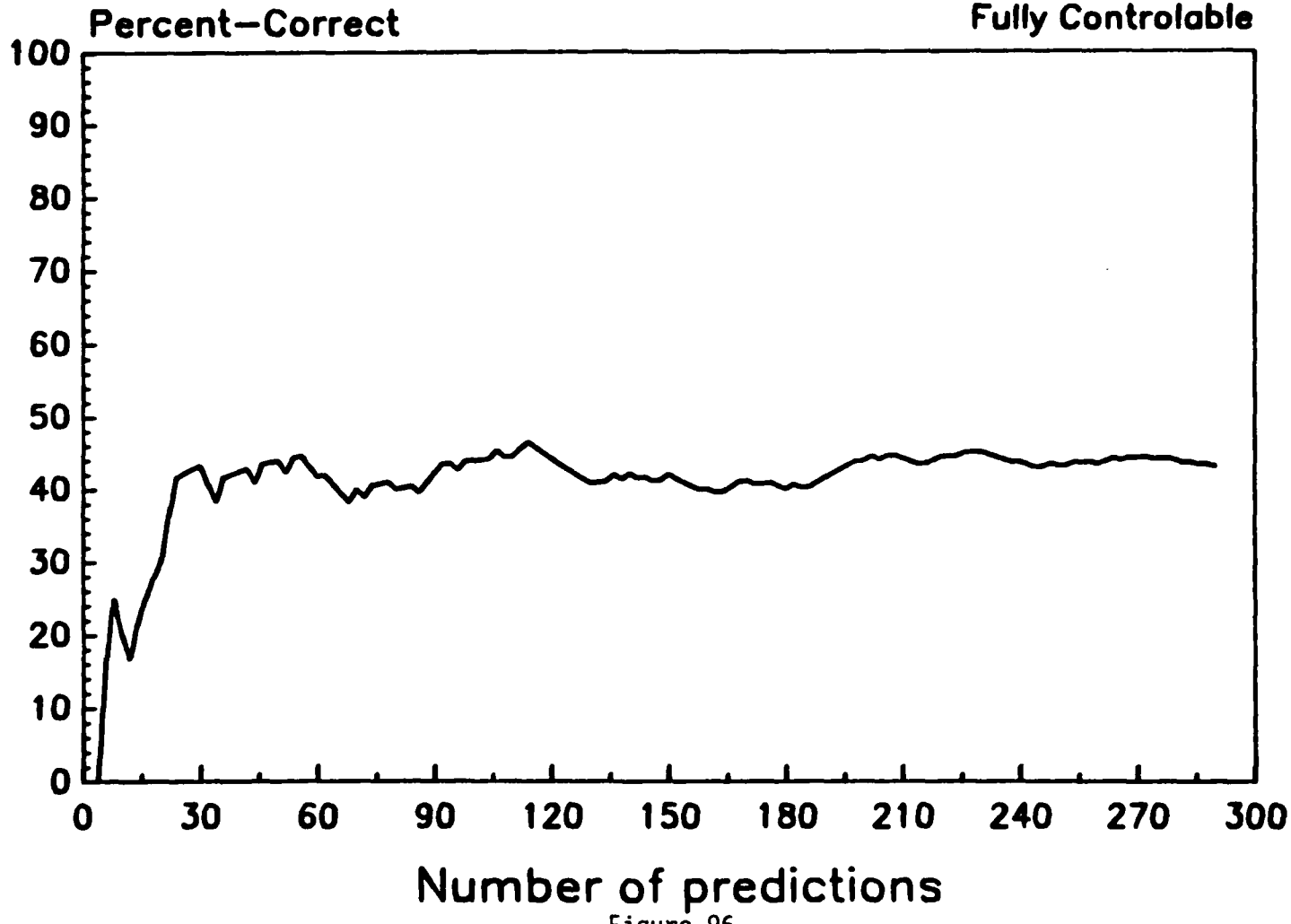


Figure 96

# Identification of a FSM of 5 states

## Four symbol alphabet

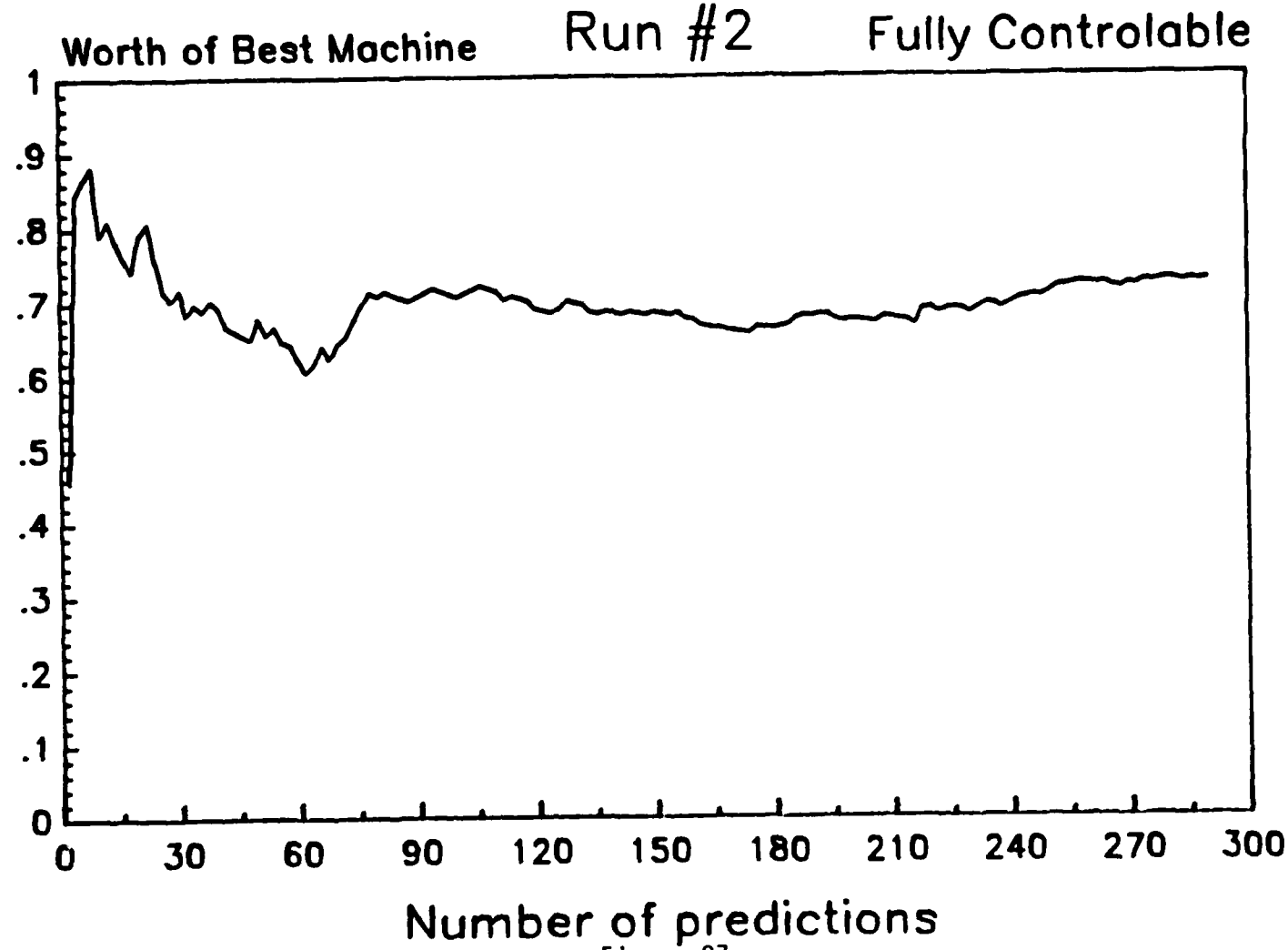


Figure 97

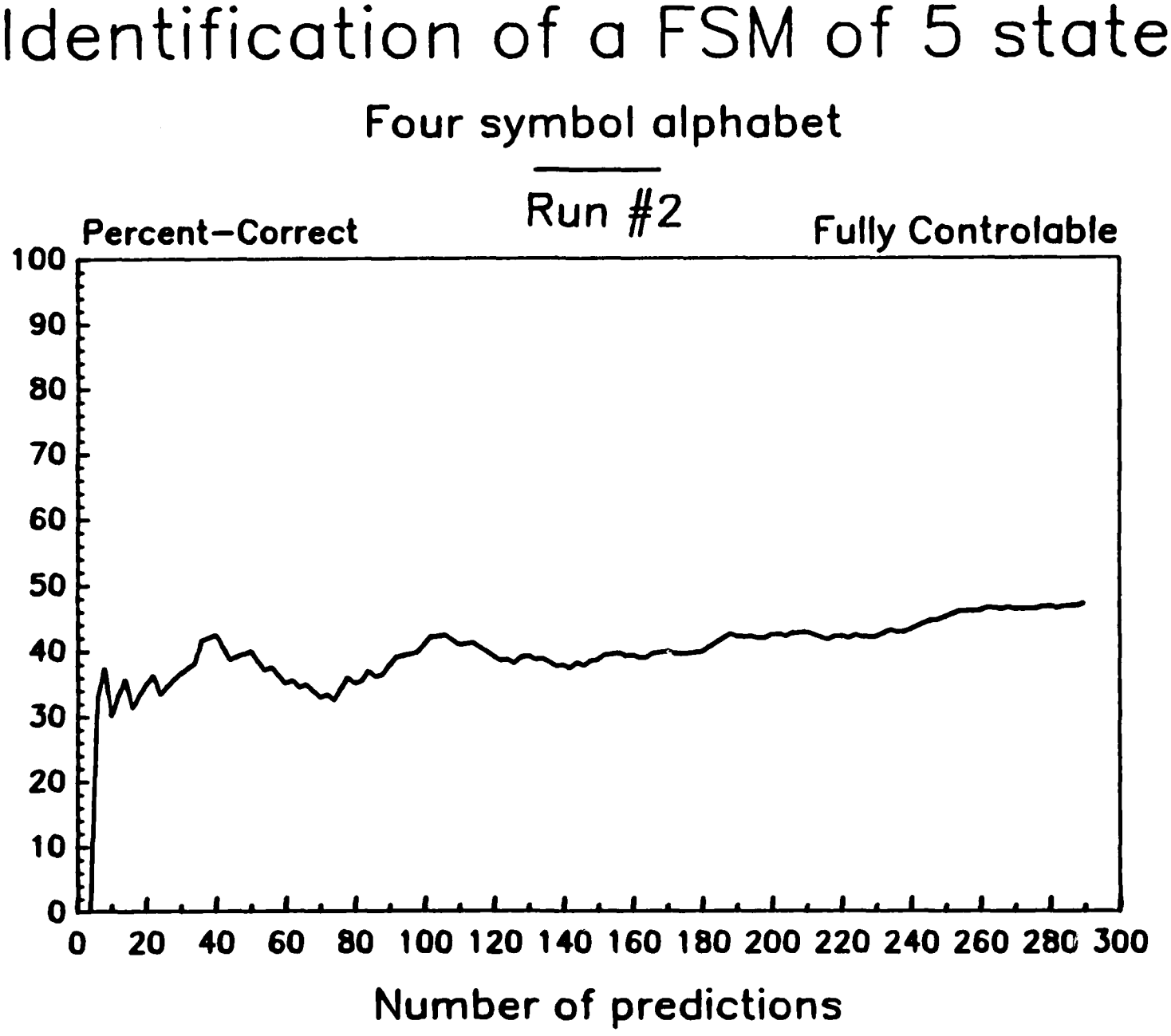
Identification of a FSM of 5 states
Four symbol alphabet
Run #2
Percent–Correct
Fully Controlable
100
90
80
70
60
50
40
30
20
10
0
0 20 40 60 80 100 120 140 160 180 200 220 240 260 280 300
Number of predictions

Figure 98

# Identification of a FSM of 5 states

## Four symbol alphabet

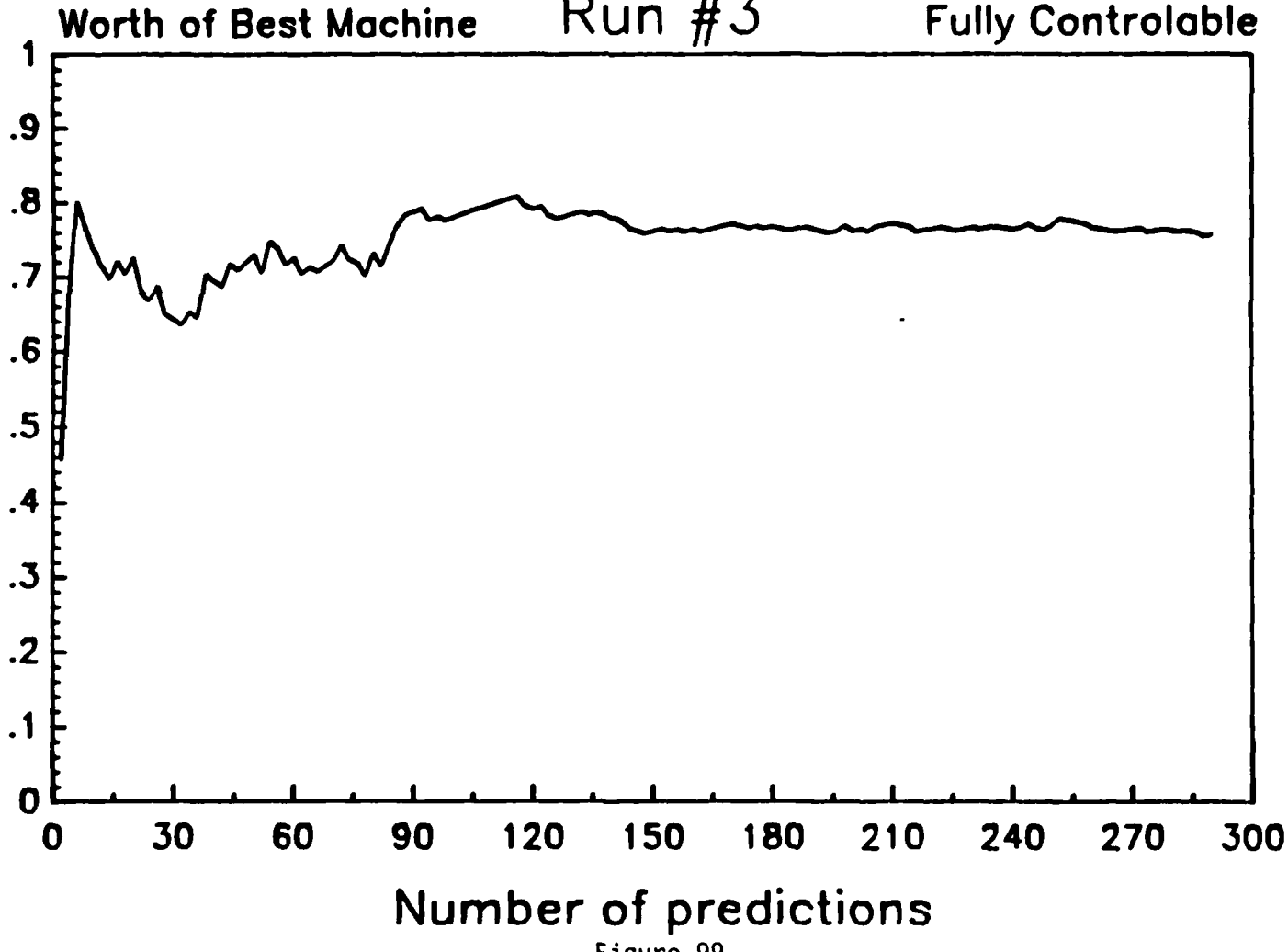


Figure 99



Figure 99

# Indentification of a FSM of 5 states

## Four symbol alphabet

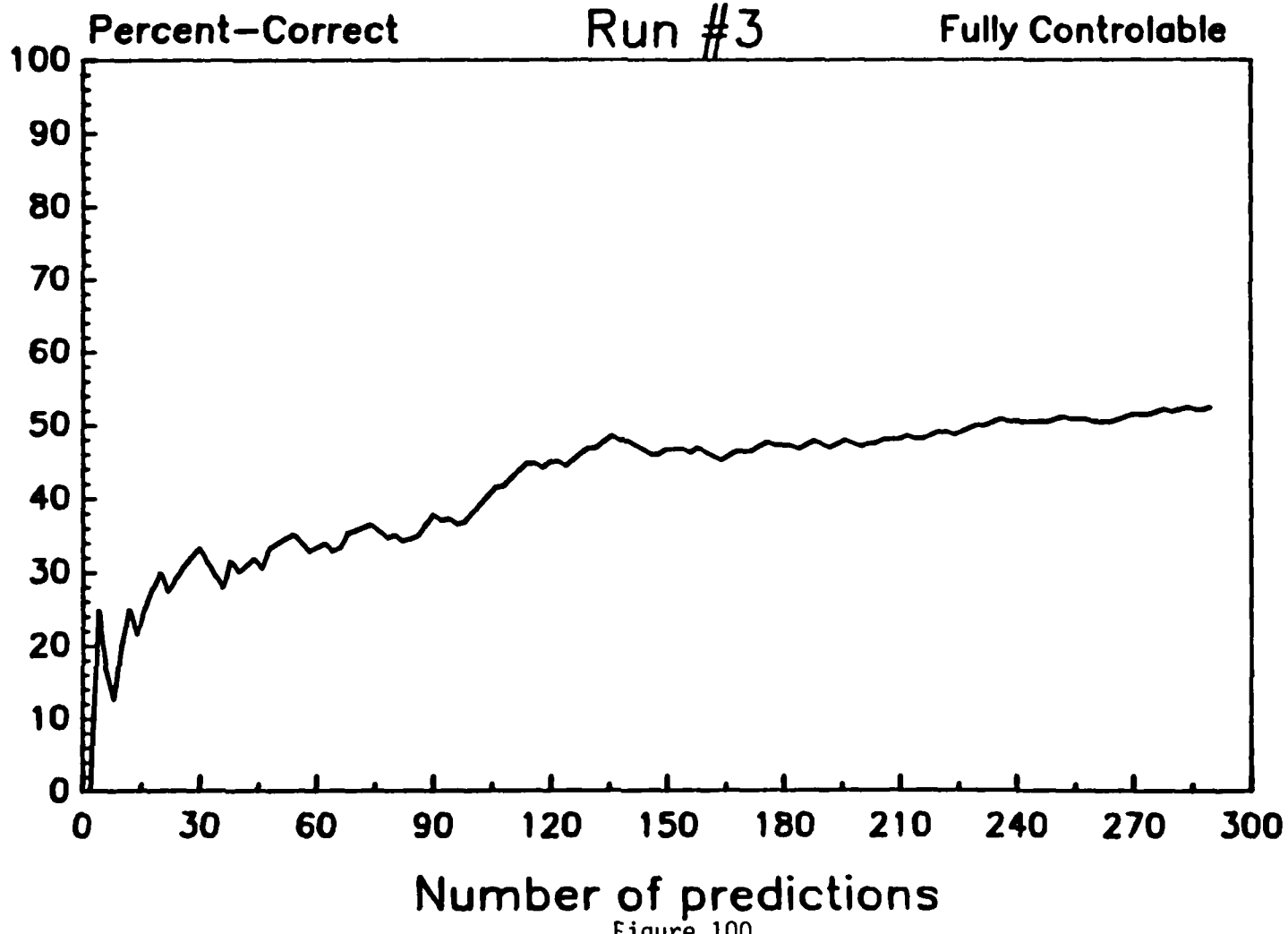


Figure 100



Figure 100

# Identification of a FSM of 5 states

## Four symbol alphabet

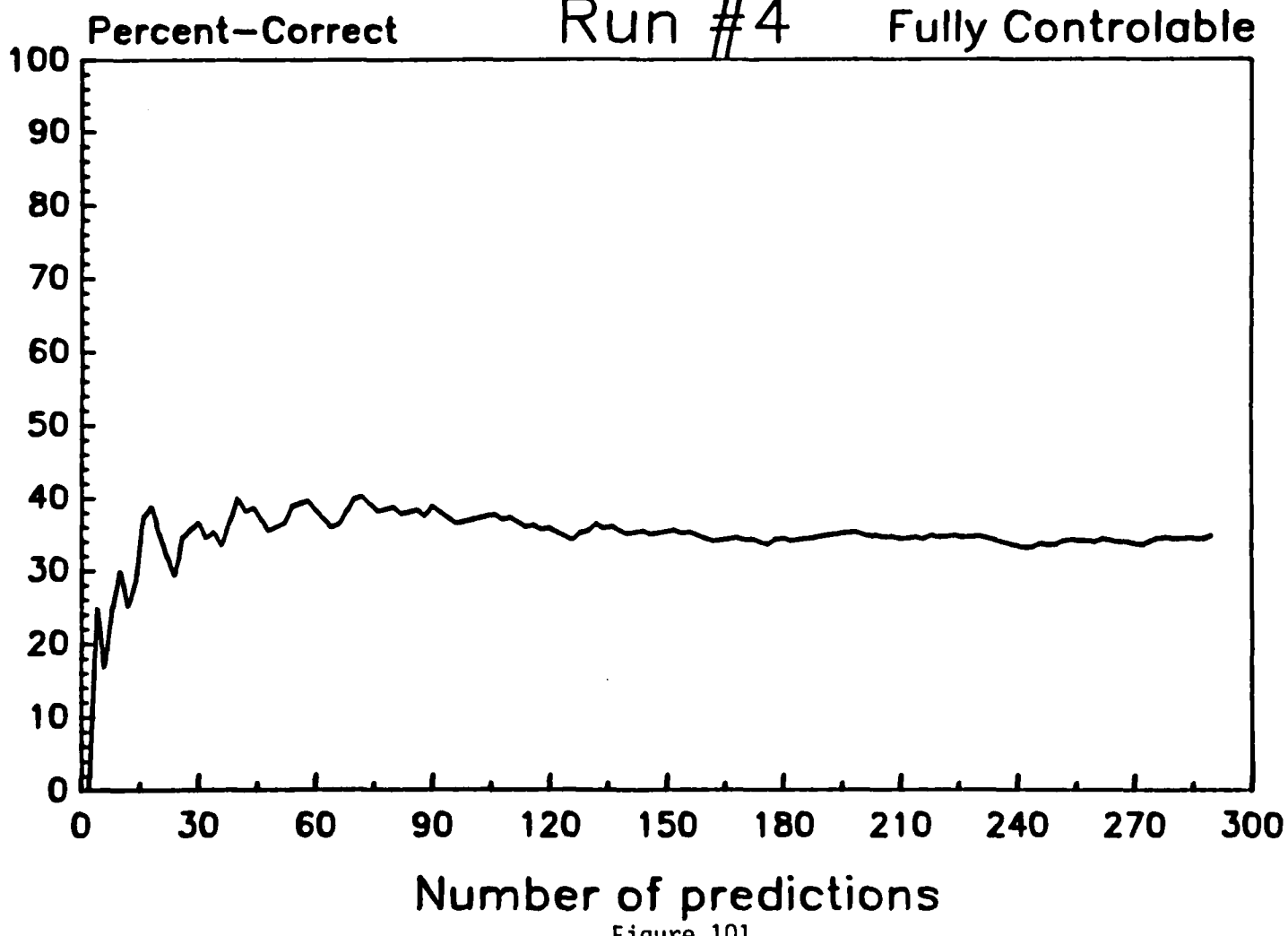


Figure 101

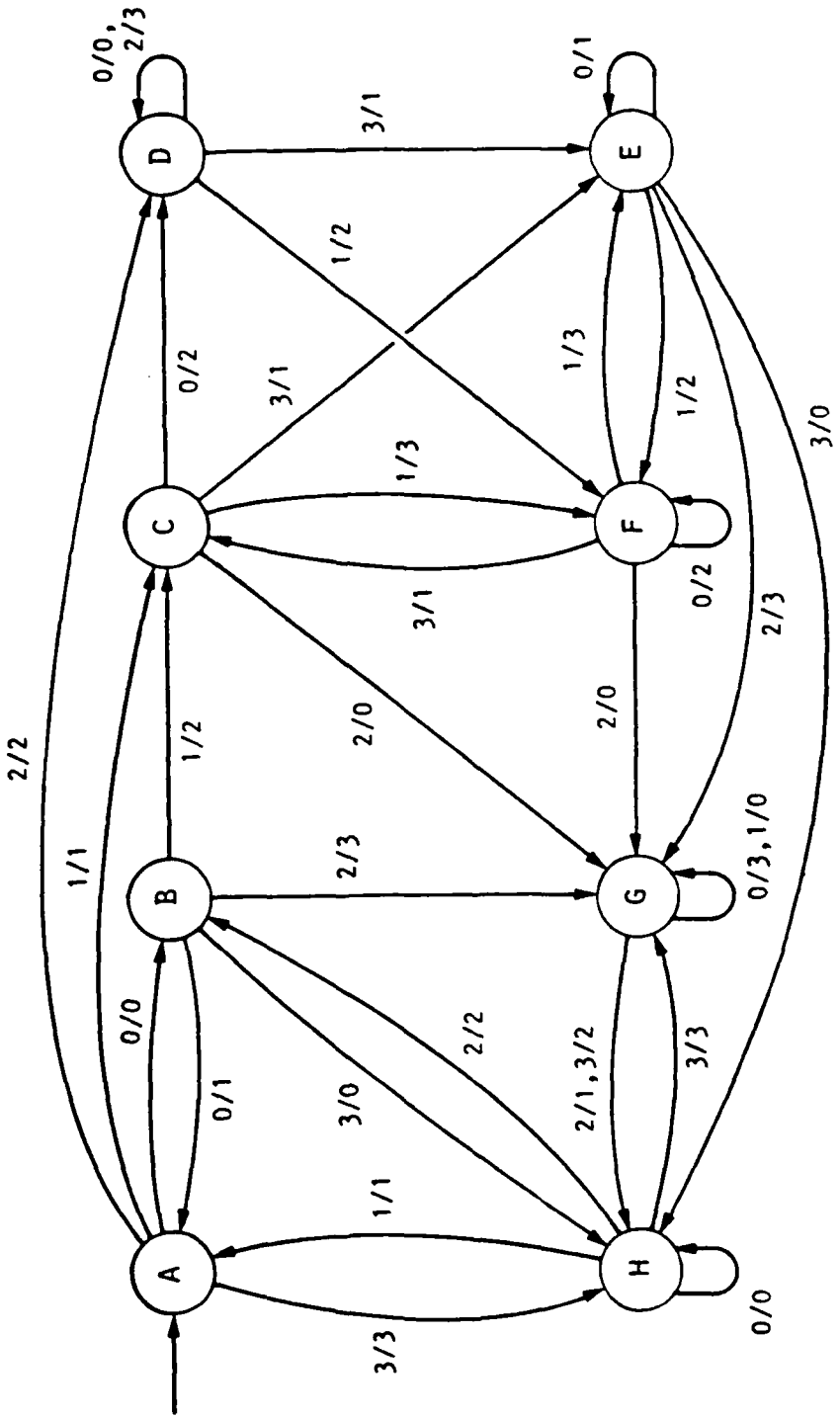


Figure 102

Figure 102

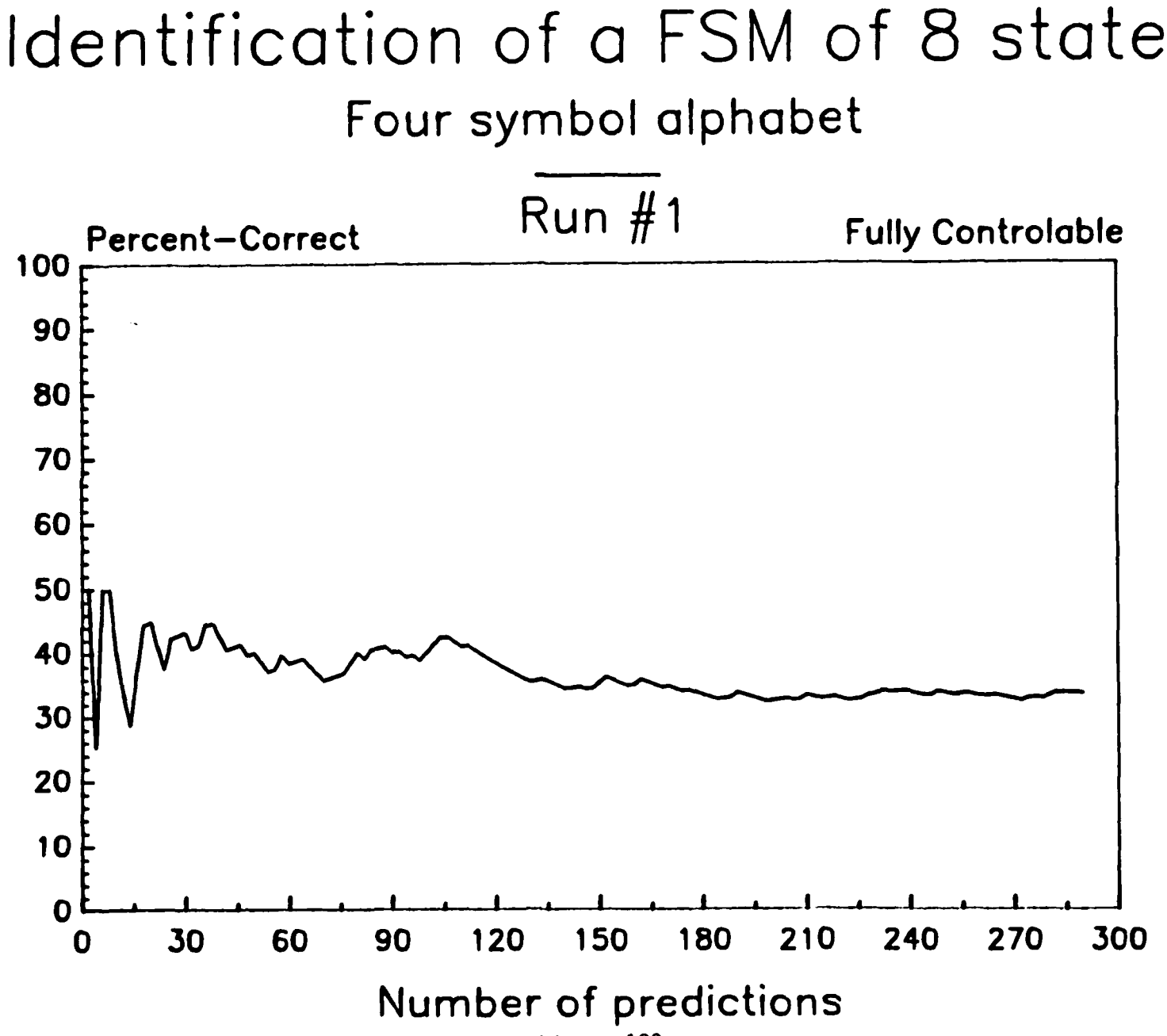
Identification of a FSM of 8 states
Four symbol alphabet
Run #1
Percent–Correct
Fully Controlable
100
90
80
70
60
50
40
30
20
10
0
0
30
60
90
120
150
180
210
240
270
300
Number of predictions

Figure 103

# Identification of a FSM of 8 states

## Four symbol alphabet

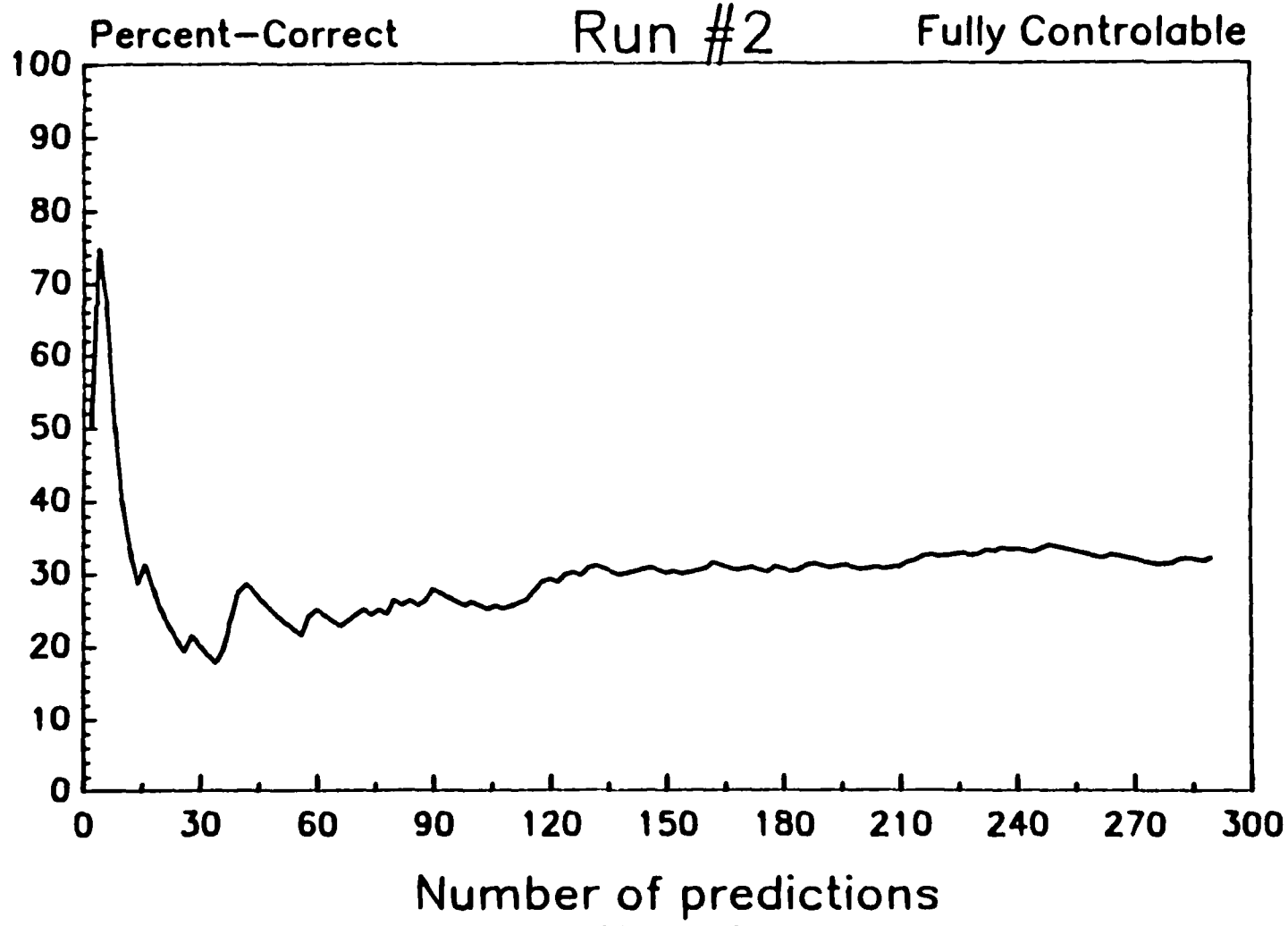


Figure 104

# Identification of a FSM of 8 states

## Four symbol alphabet

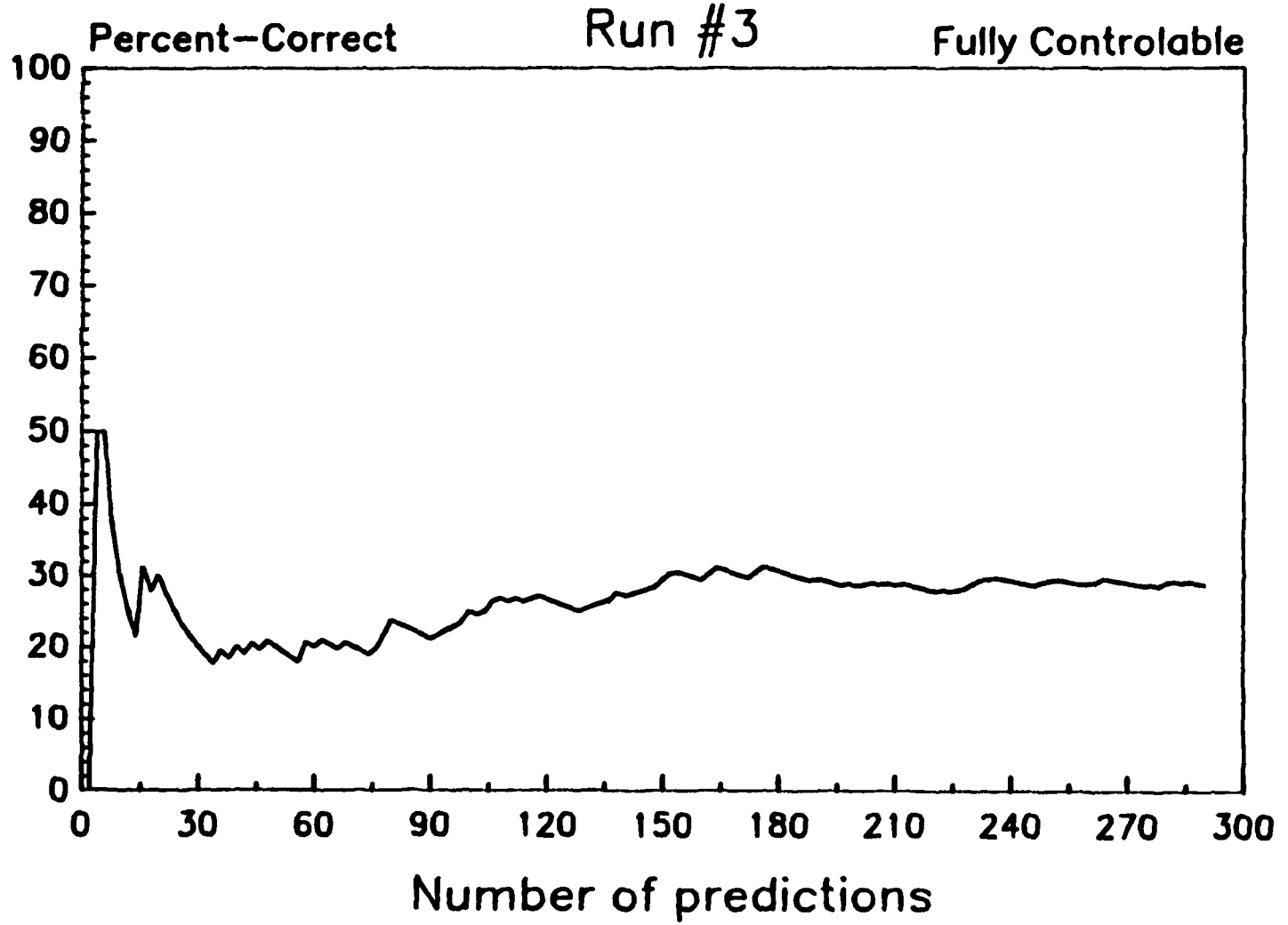


Figure 105

# Identification of a FSM of 8 states

## Four symbol alphabet

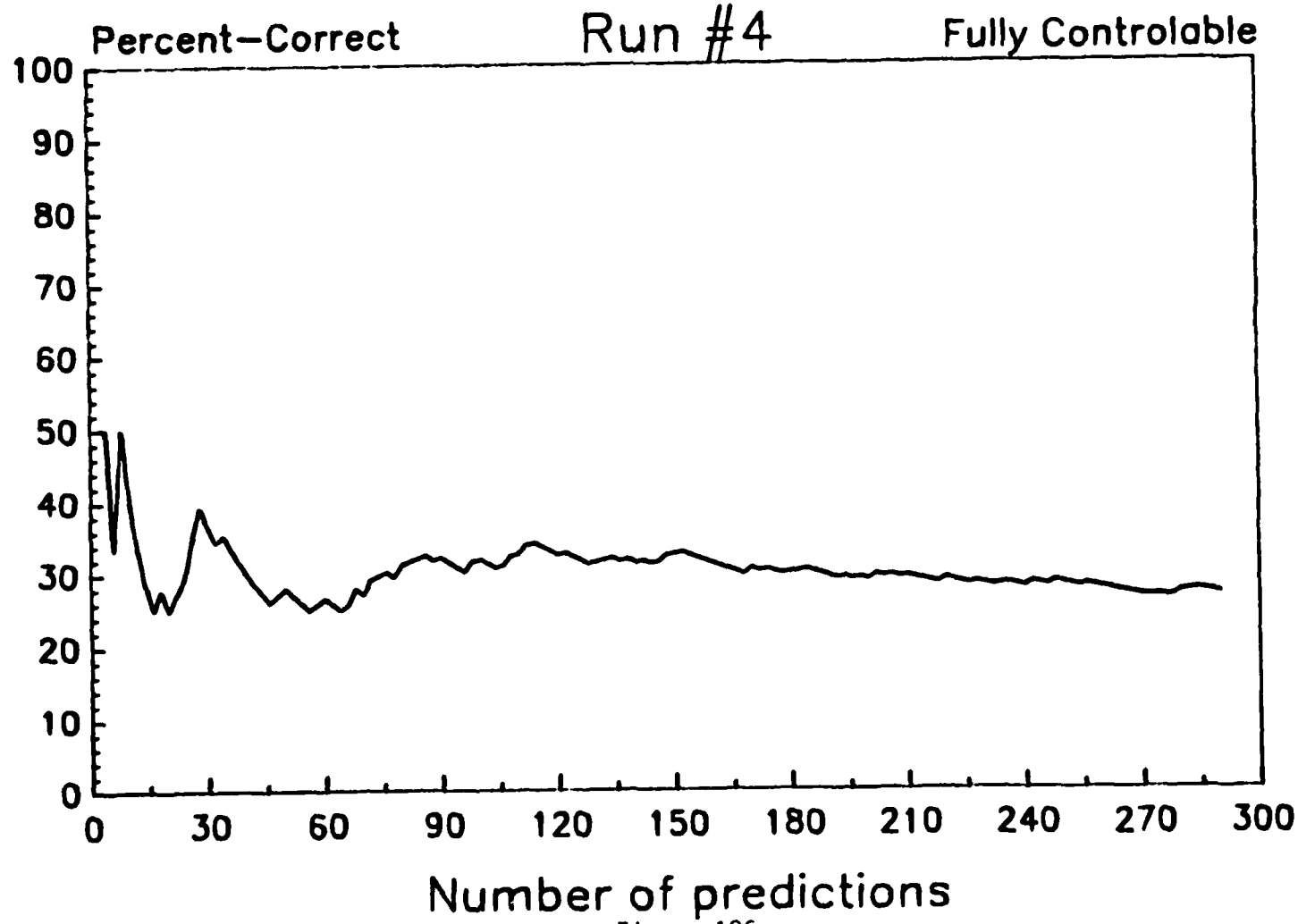


Figure 106

Figure 106

# Identification of a FSM of 8 states

## Four symbol alphabet

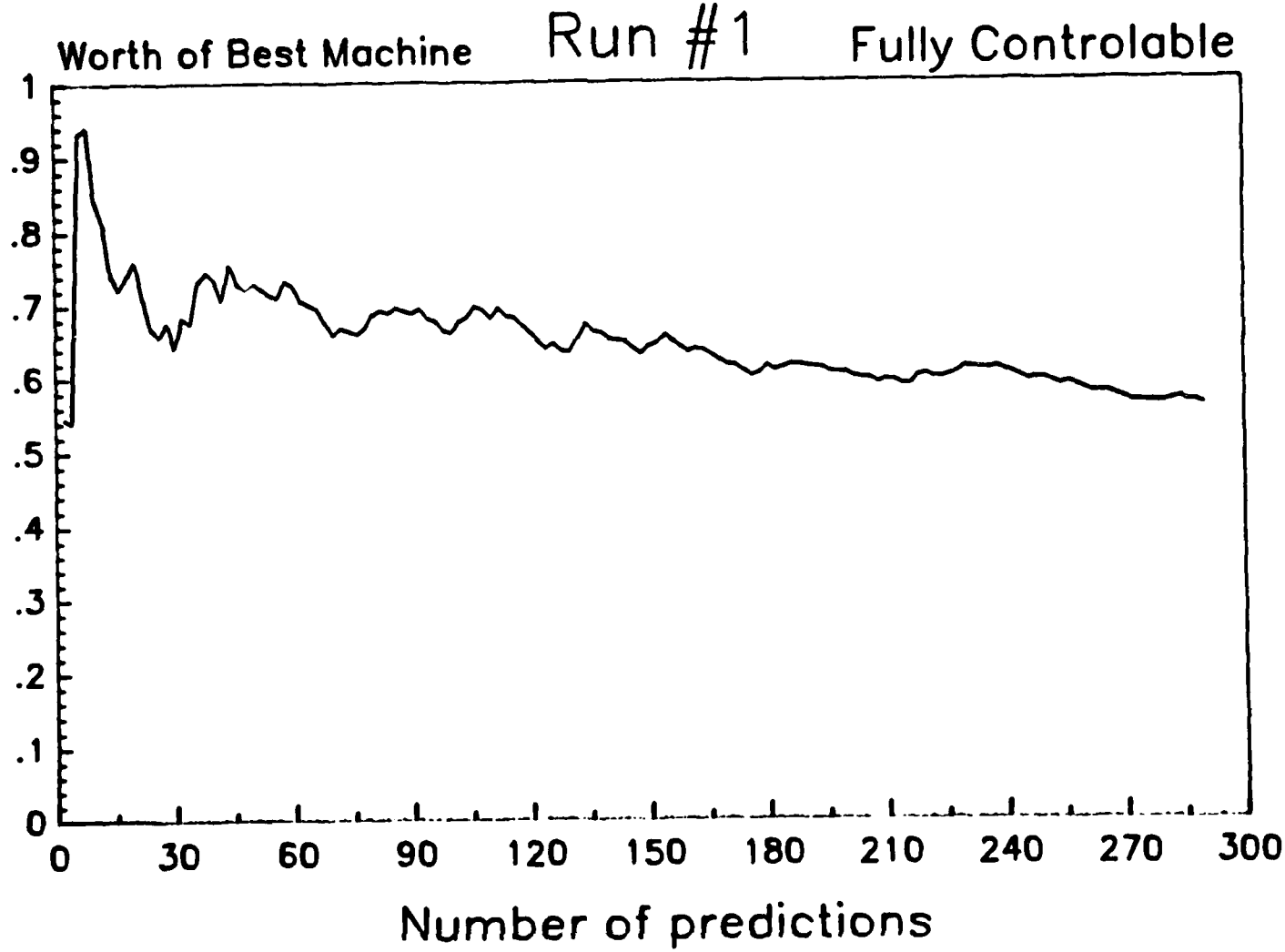


Figure 107

# Identification of a FSM of 8 states

## Four symbol alphabet

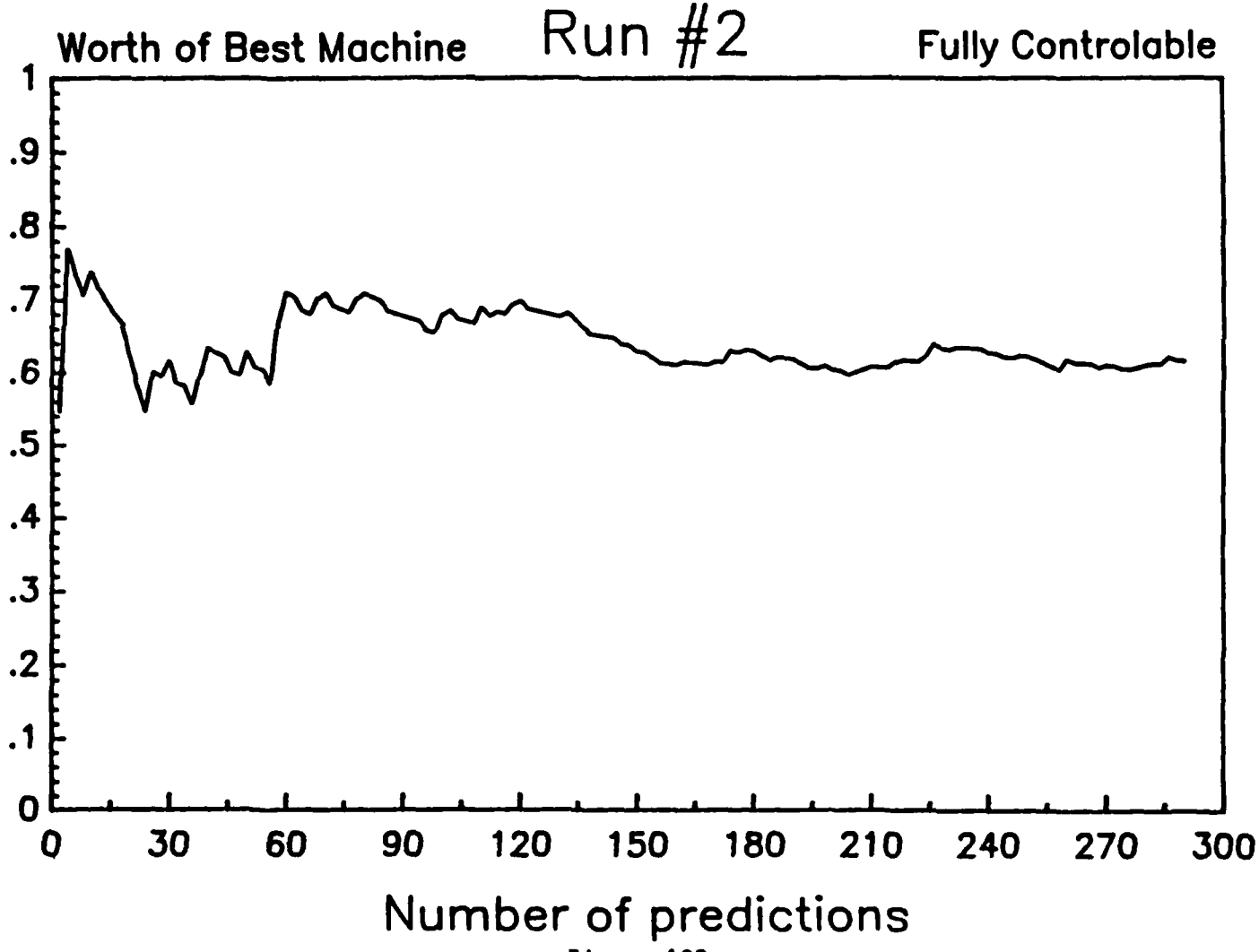


Figure 108

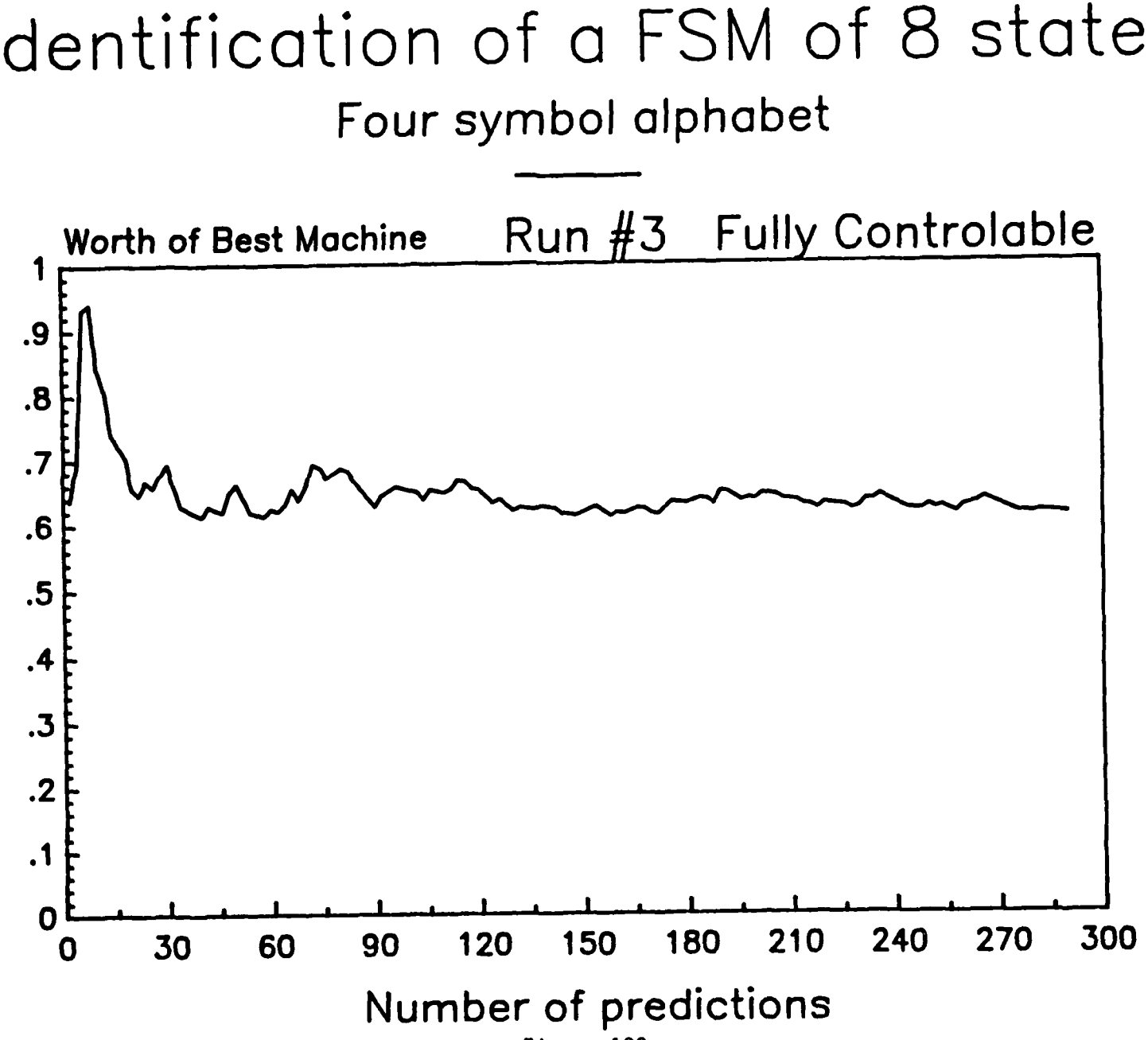
Identification of a FSM of 8 states
Four symbol alphabet
Worth of Best Machine
Run #3 Fully Controlable
1
.9
.8
.7
.6
.5
.4
.3
.2
.1
0
0
30
60
90
120
150
180
210
240
270
300
Number of predictions

Figure 109

# Identification of a FSM of 8 states

## Four symbol alphabet

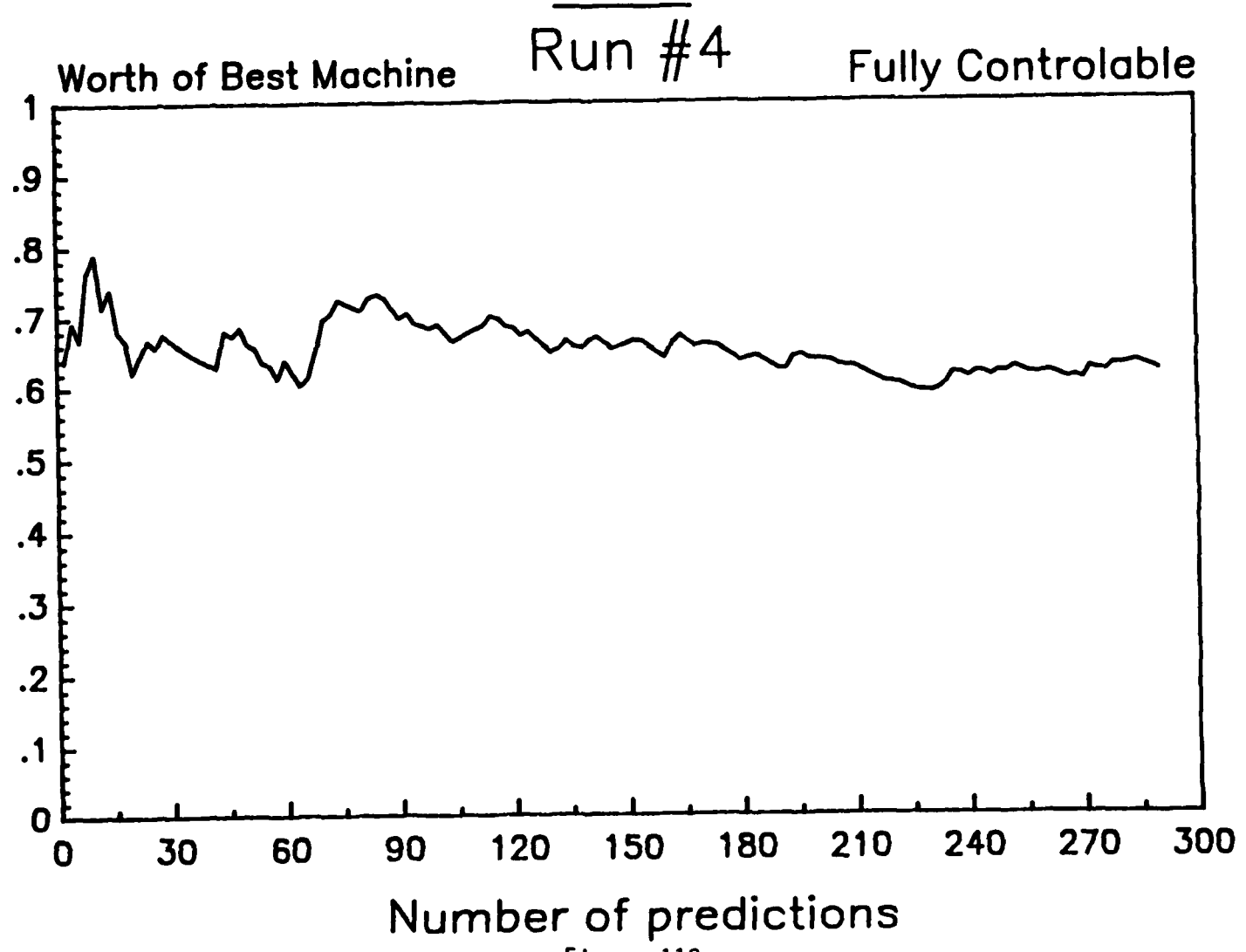


Figure 110

# Identification of a FSM of 8 states

## Four symbol alphabet

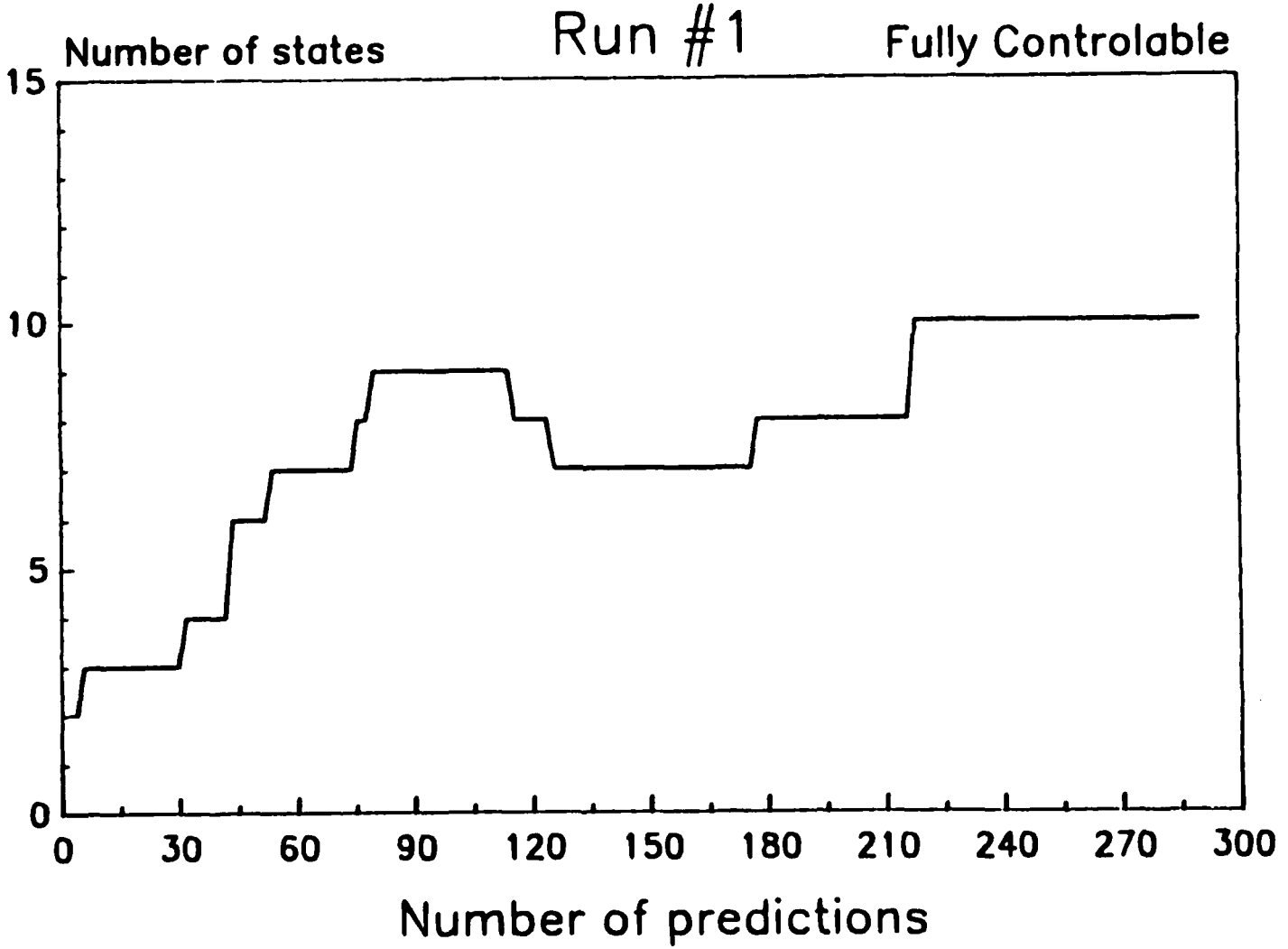


Figure 111

# Identification of a FSM of 8 states

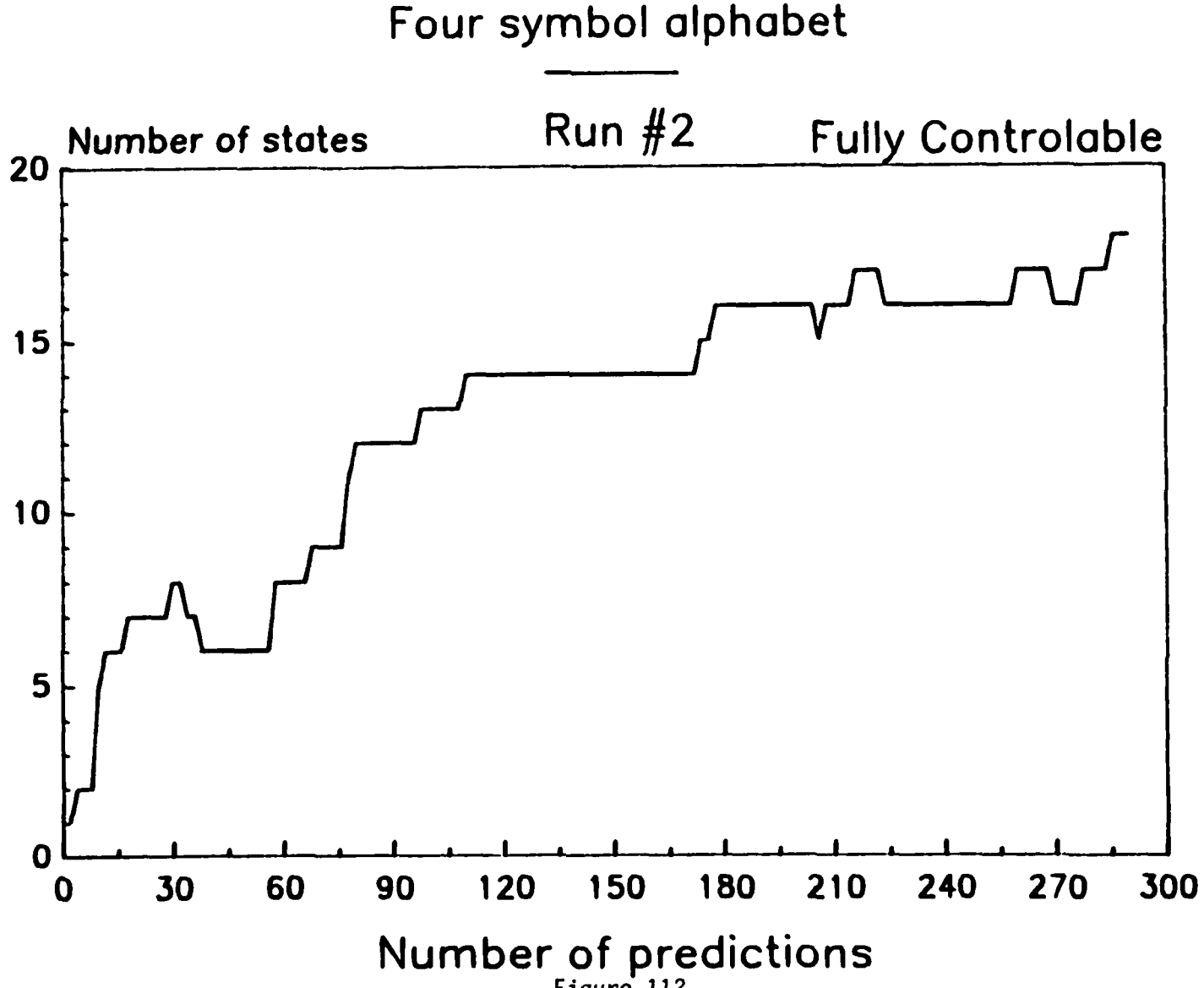


Figure 112

# Identification of a FSM of 8 states

## Four symbol alphabet

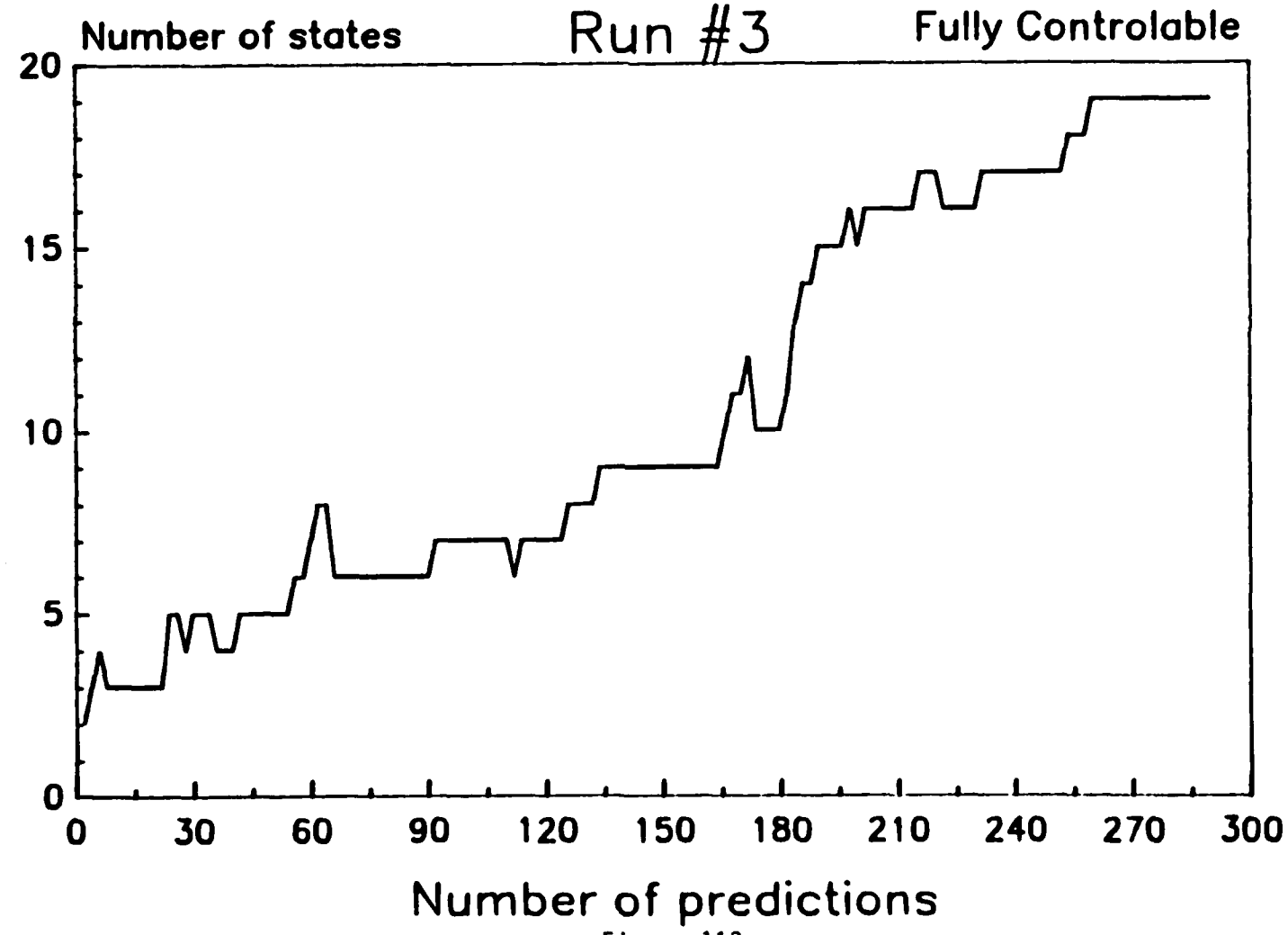


Figure 113

# Identification of a FSM of 8 states

## Four symbol alphabet

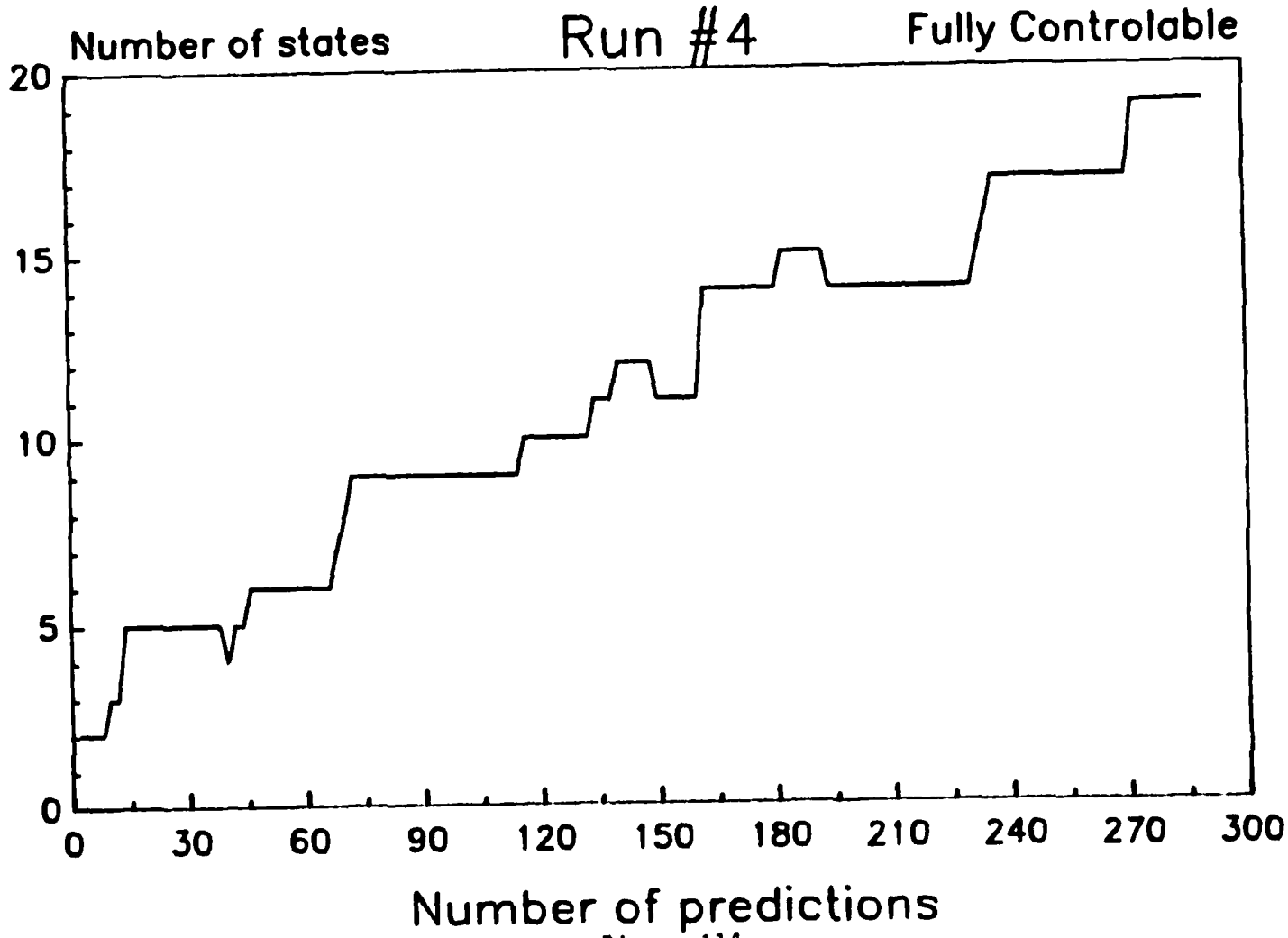


Figure 114

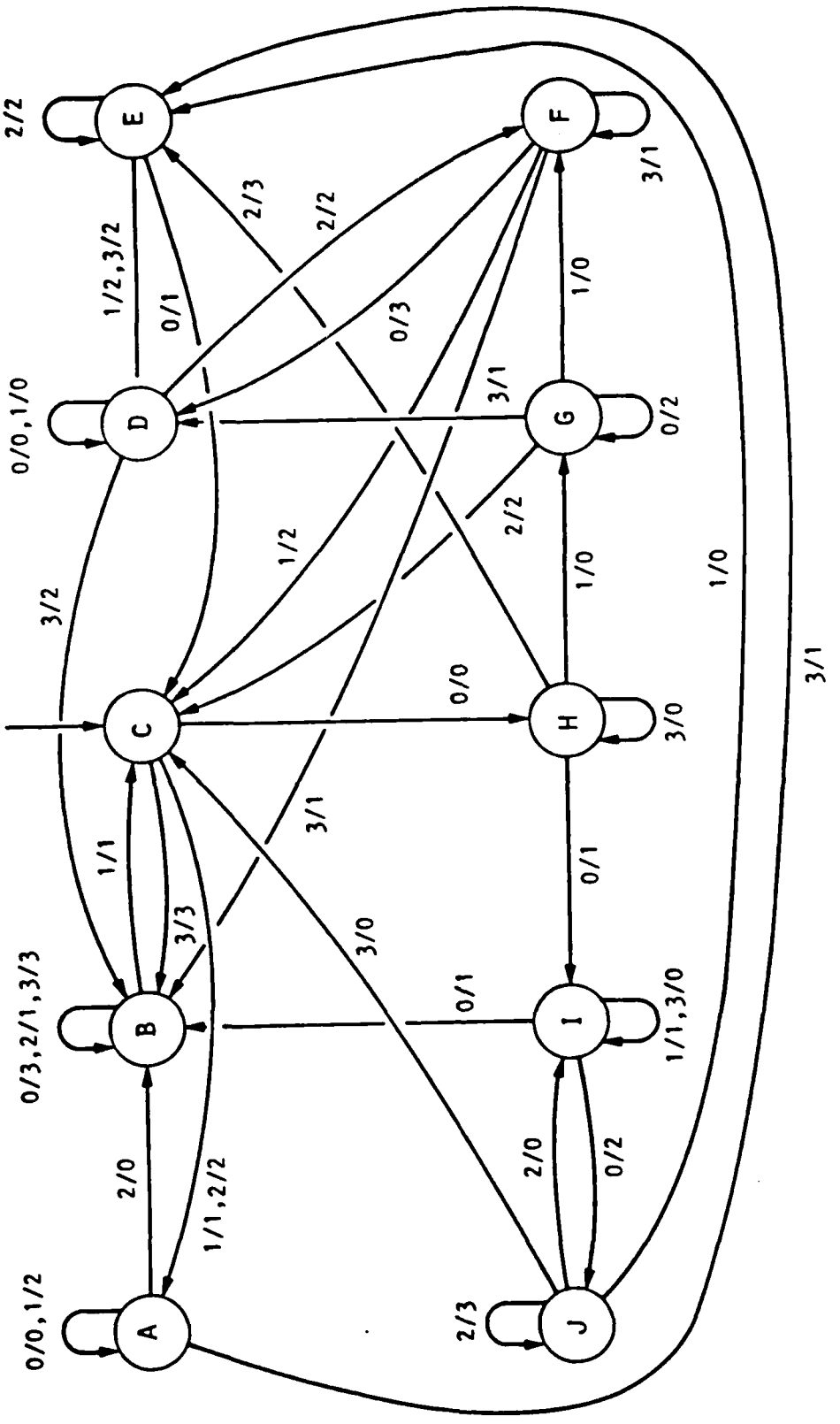
A
B
C
D
E
F
G
H
I
J
0/0,1/2
0/3,2/1,3/3
0/0,1/0
2/2
3/1
0/2
3/0
1/1,3/0
2/3
1/2,3/2
0/1
2/3
2/2
0/3
1/0
3/1
3/2
1/2
2/2
1/0
1/0
3/1
0/0
1/1
3/3
3/1
3/0
0/1
0/1
2/0
1/1,2/2
2/0
0/2

Figure 115

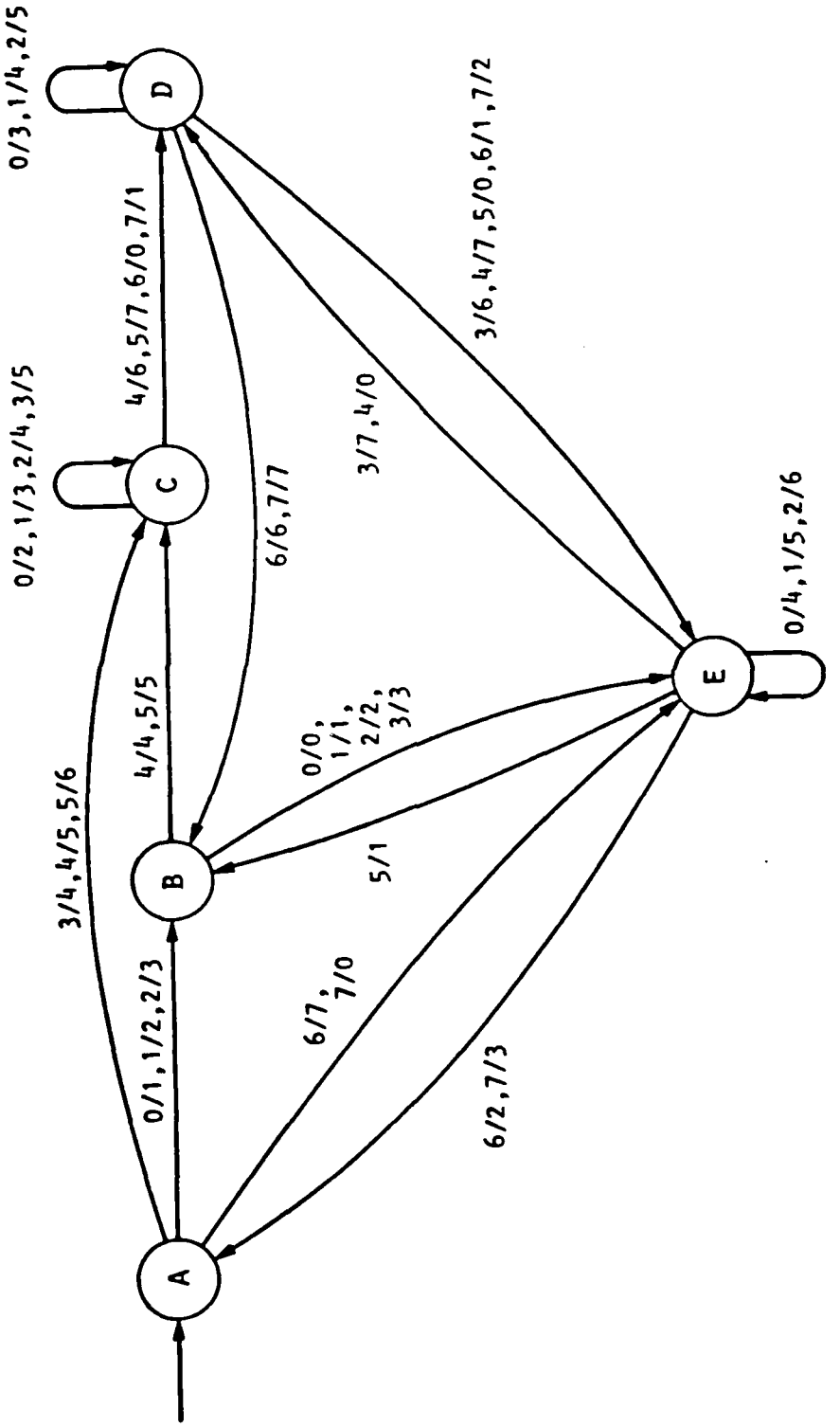


Figure 116

Figure 116

# Identification of a FSM of 5 states

## Eight symbol language

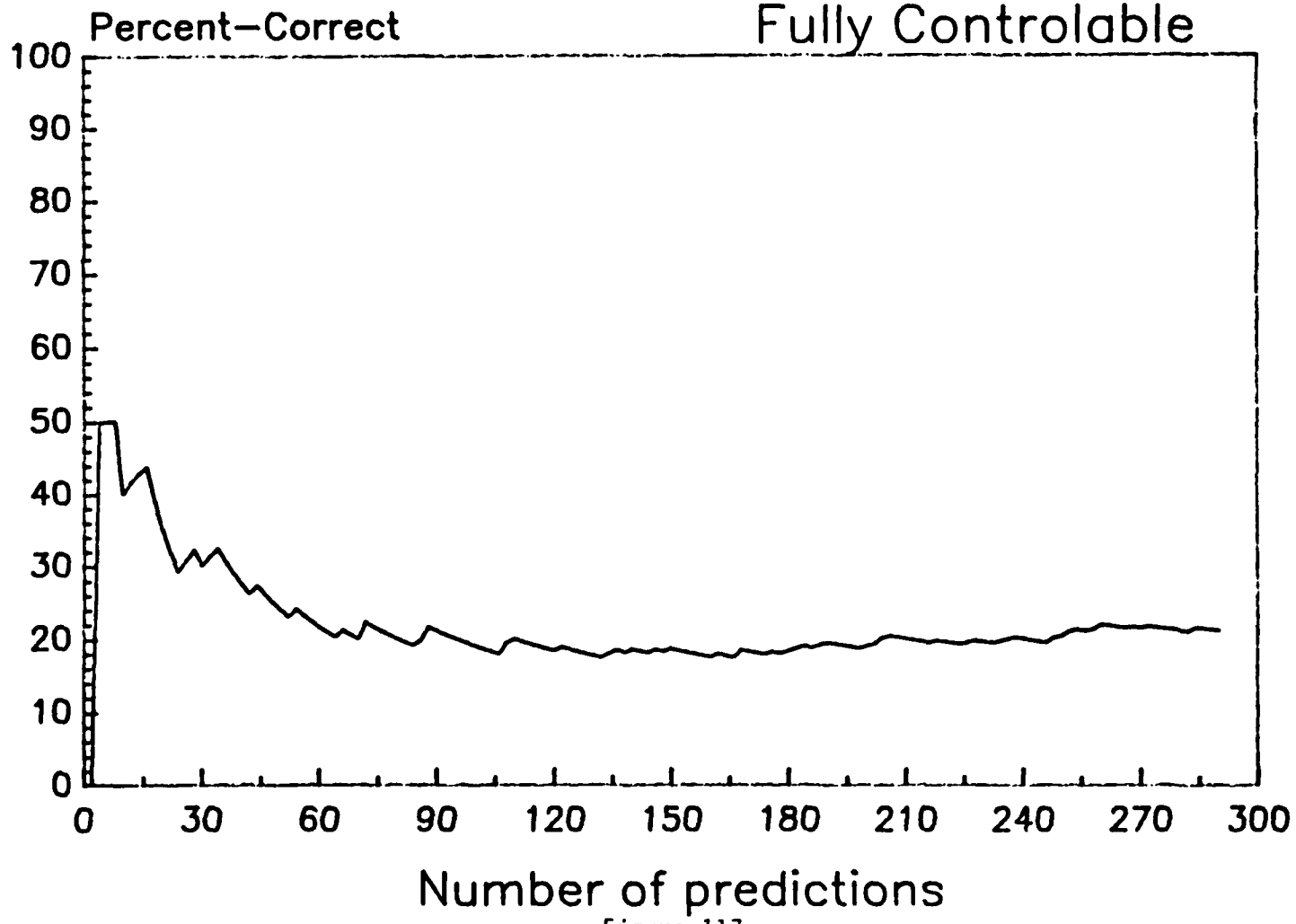


Figure 117

# Identification of a FSM of 5 states

## Eight symbol language

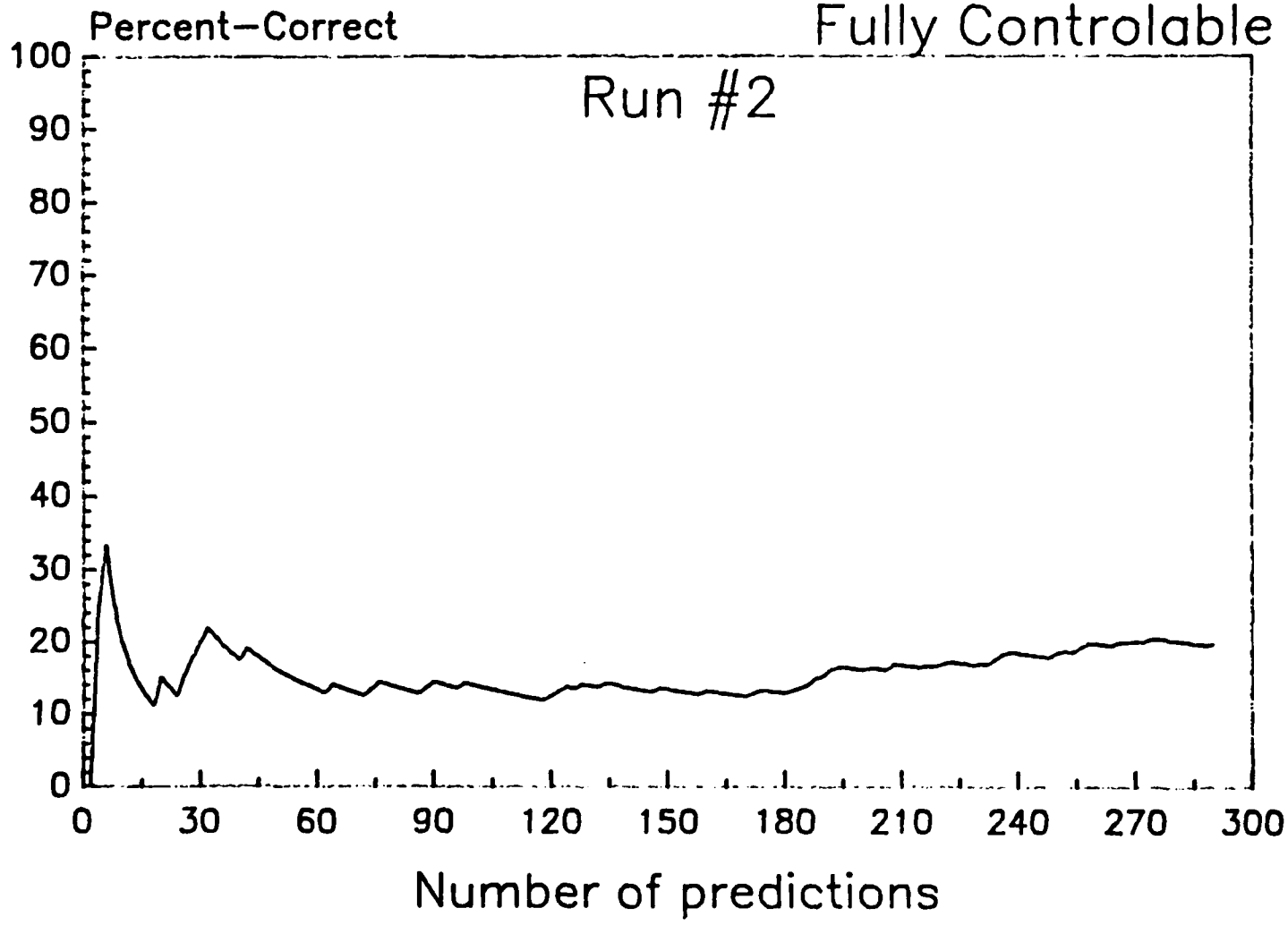


Figure 118

# Identification of a FSM of 5 states

## Eight symbol language

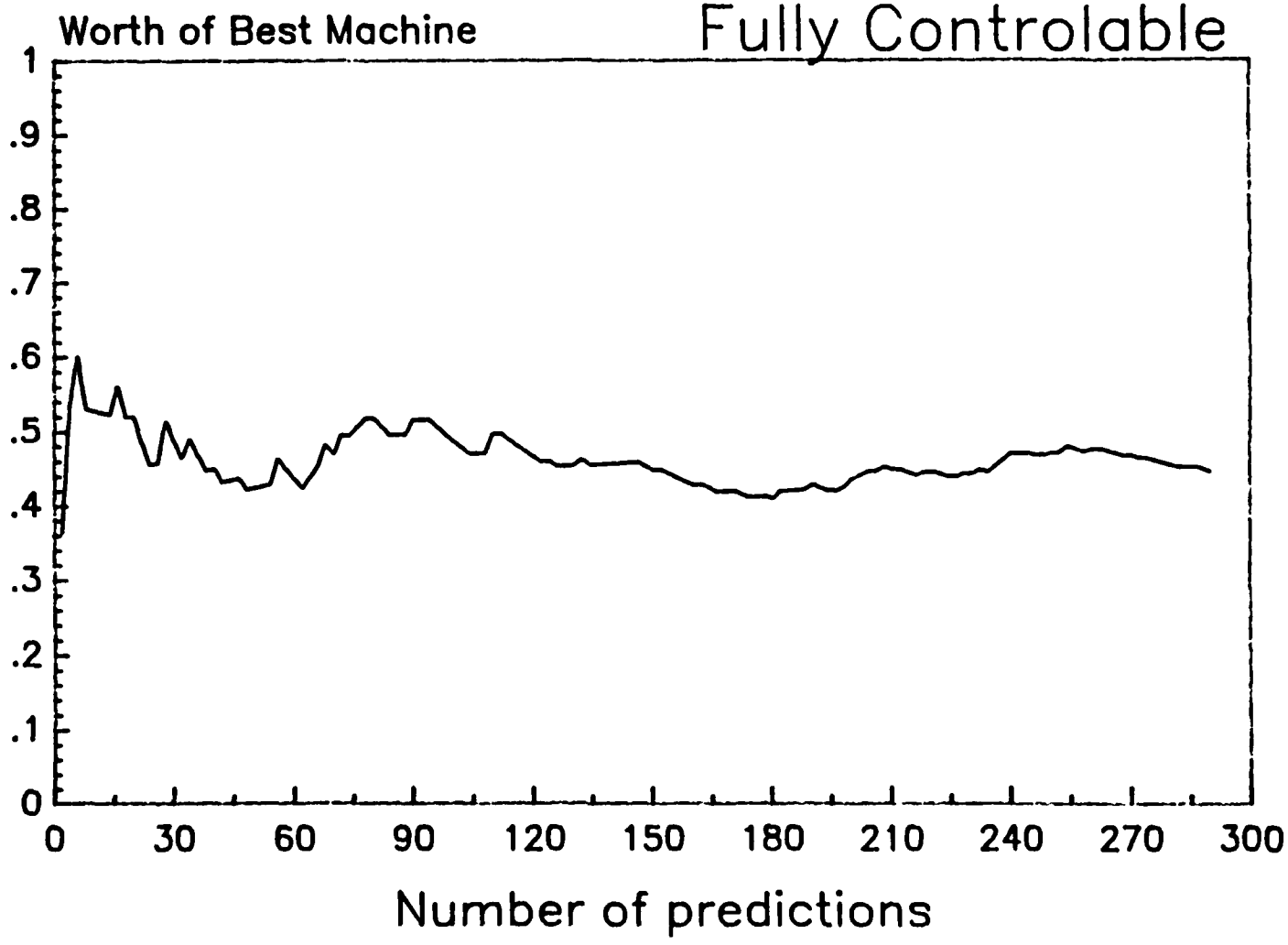


Figure 119

# Identification of a FSM of 5 states

## Eight symbol language

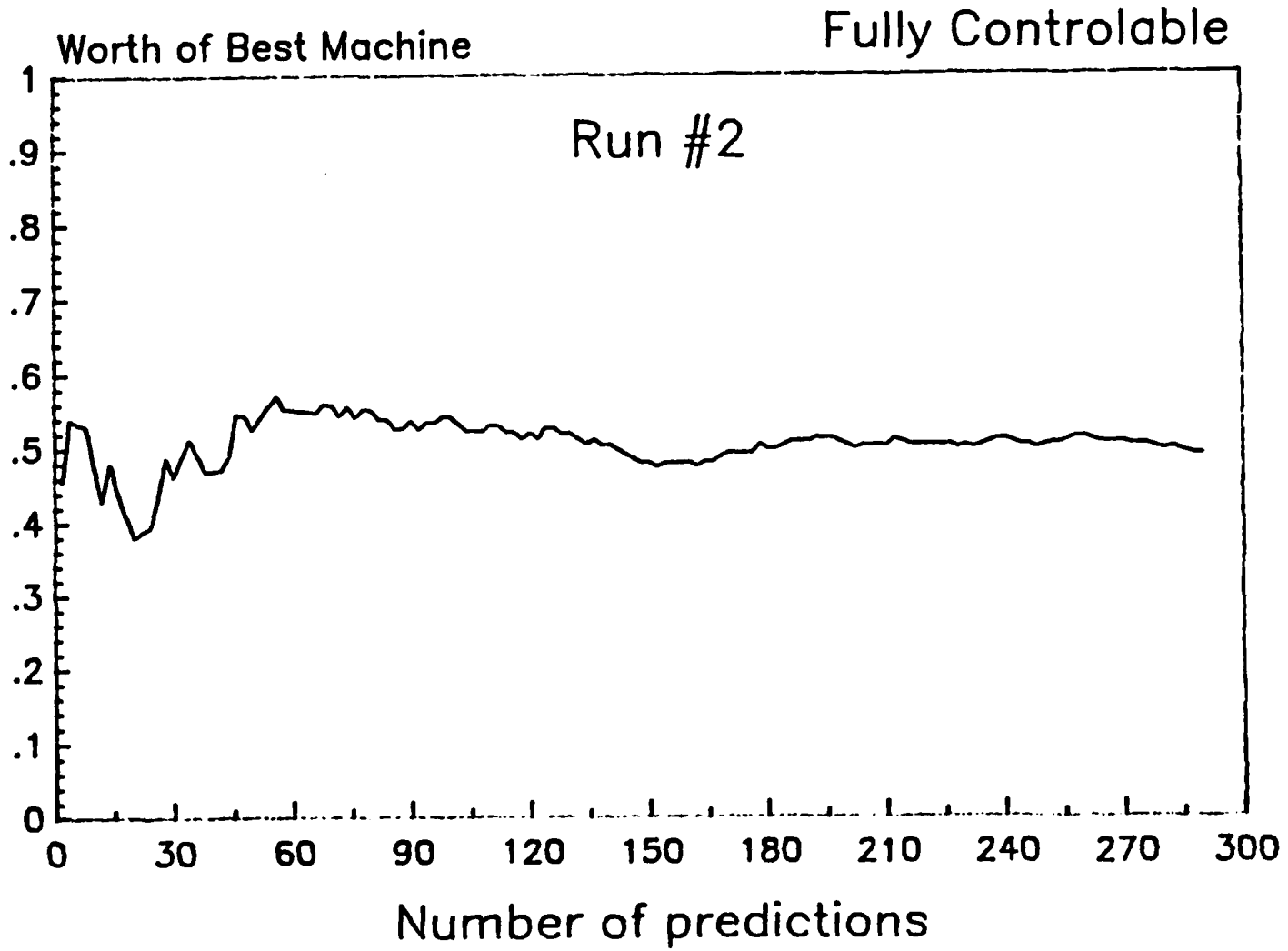


Figure 120

worth and the percent correct. Once again, there was a steady increase in the size of the evolving machine, see Figures 121 and 122. The final machines range from 16 to 18 states.

These experiments indicate successful identification through evolutionary programming. The logic of the unknown transducer was represented by finite state machines of greater size than required. This is especially true with a larger alphabet. However, no penalty for complexity was invoked. It should be possible to evolve machines of similar size to the unknown plant, but there might be some cost for this decreased specificity.

On the basis of these results, it seems worthwhile to explore the identification of time varying and noisy plants through pilot experiments, then design of experiments for predictive identification using the Cray computer.

# Identification of a FSM of 5 states

## Eight symbol language

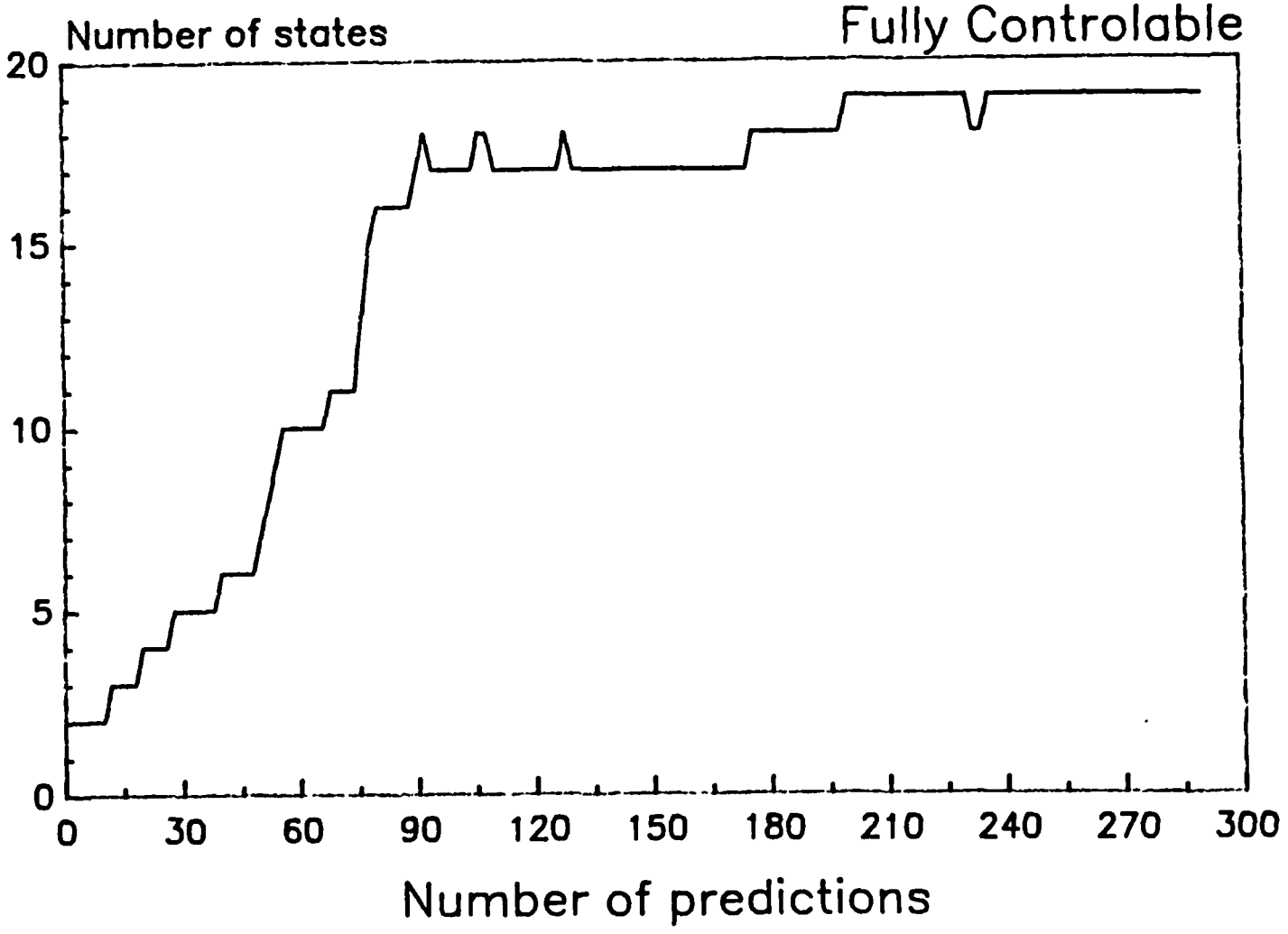


Figure 121

# Identification of a FSM of 5 states

## Eight symbol language

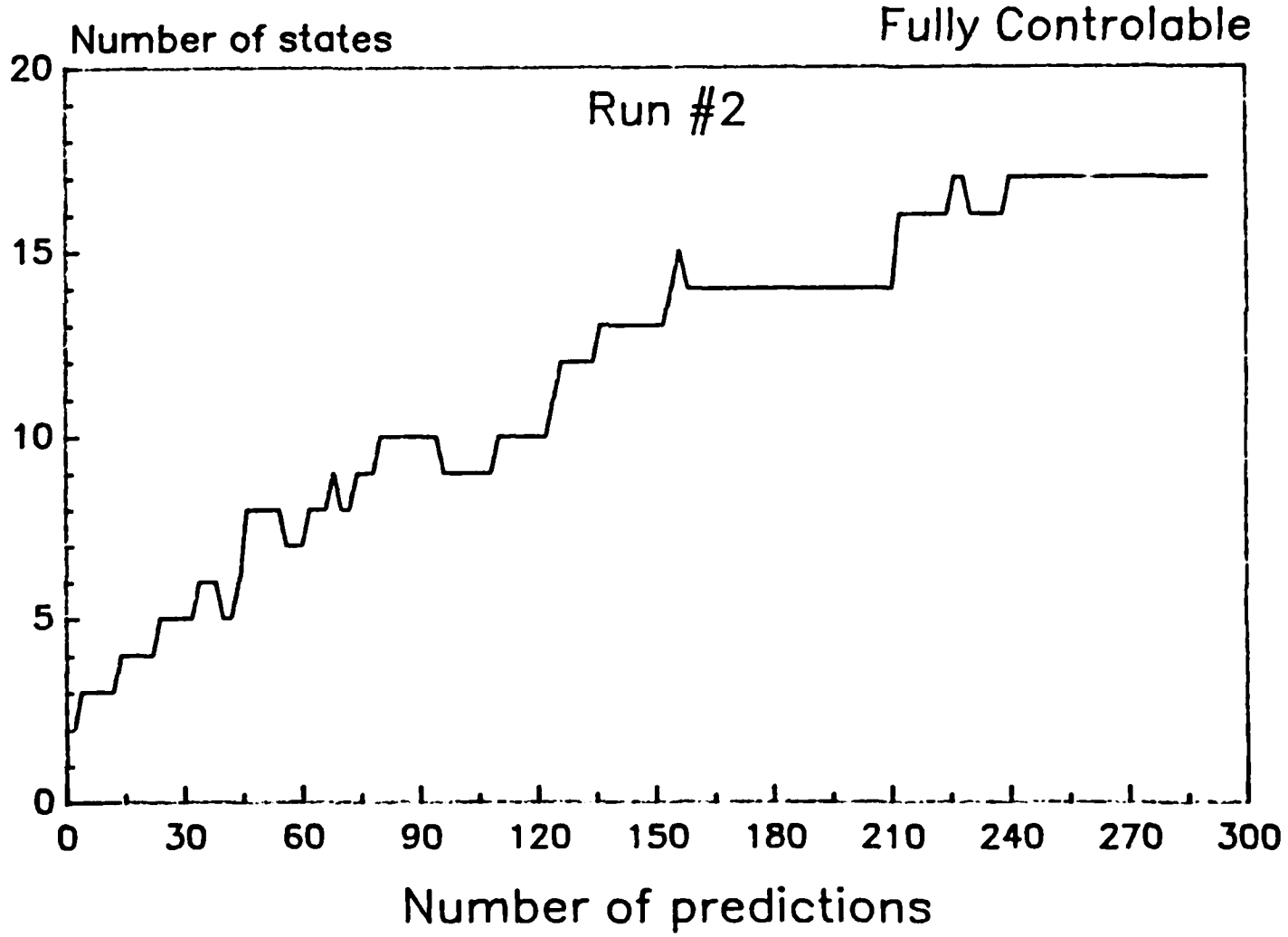


Figure 122

## CONCLUSION

Evolutionary Programming has now been demonstrated as a versatile means for the prediction of nonstationary time series and the identification of an unknown plant. It is fully recognized that much remains to be done to dimensionalize the program as a function of the presented problem in both prediction and identification. Arrangements are underway to use the NASA Ames Cray computer for such work. In the mean time, further pilot experiments are planned to guide these larger-scale demonstrations.

It is now considered appropriate to explore the application of prediction and identification to real world problems in a preliminary manner. Medical, financial and other data sources are being reviewed in this regard. It is of particular interest to focus attention on situations wherein the least mean squared payoff function is notably inappropriate. This is surely the case for epidemiologic and economic time series where *equally correct predictions* are not of equal worth, and the cost of a "cry wolf" error is clearly different from the cost of a missed catastrophe.

Plans are being made to move from identification to control through a series of pilot experiments. Success in this regard might well influence the design of weapon systems.

# Appendix

Addressing
The Traveling Salesman Problem
Through Adaptation

# ADDRESSING
# THE TRAVELING SALESMAN PROBLEM
# THROUGH ADAPTATION

By David Fogel

## INTRODUCTION

The traveling salesman problem has received a great deal of attention in recent years. The task is to arrange a tour of n cities such that each city is visited only once and the length of the tour (or some other cost function) is minimized. Here is a simple yet recalcitrant combinatorial optimization problem. For an exact solution the only known algorithms require the number of steps to grow at least exponentially with the number of elements in the problem. Brute force finding of the shortest path by which a traveling salesman can complete a tour of n cities requires compiling a list of (n-1)!/2 alternative tours...a number that grows faster than any finite power of n. The task quickly becomes unmanageable.

## BACKGROUND

Two recent papers[1] addressed the traveling salesman problem

---

1- "Proceedings of an International Conference on Genetic Algorithms and Their Applications," John J. Grefenstette, Editor, Carnegie-Mellon University, July 1985.

through use of the genetic algorithm as proposed by J.H. Holland in 1975.[2] This algorithm is an offshoot of the evolutionary programming concept offered by L.J. Fogel in 1962,[3] then demonstrated in his doctoral dissertation[4] and described in the book Artificial Intelligence through Simulated Evolution.[5] Here, intelligent behavior was viewed as requiring prediction of an environment then the use of such predictions for the sake of its control (at least to the extent possible). The logical process of iterative mutation and selection of behavior is simulated in fast time to eventually evolve a logic most suitable for resolving the given problem.

The behavior of each artificial organism is portrayed by a finite state machine... a mathematical function that does not constrain the represented transduction. It need not be linear, passive, or without hysteresis. The original "machine" (an arbitrary logic or a "hint") is measured in its ability to predict each next event in its "experience" with respect to whatever payoff function has been prescribed. An offspring is now created through random mutation of this "parent" machine. It is scored in a similar manner to the parent in predictive ability. If the parent is better than its offspring, the parent is used to generate another offspring. If, however, the offspring is better than the parent, the offspring becomes the new parent. This assures non-regressive evolution.

---

2- Adaptation in Natural and Artificial Systems, J.H. Holland, University of Michigan Press, Ann Arbor, 1975.

3- "Autonomous Automata," L.J. Fogel, *Industrial Research*, February 1962.

4- "On the Organization of Intellect," L.J. Fogel, Ph.D. Dissertation, U.C.L.A. 1964.

5- Artificial Intelligence through Simulated Evolution, L.J. Fogel, A.J. Owens, M.J. Walsh, John Wiley & Sons, New York, 1966.

An actual prediction is made when the predictive fit score demonstrates that a sufficient level of credibility has been achieved. The surviving machine generates a prediction, indicates the logic of this prediction and becomes the progenitor for the next sequence of progeny, this in preparation for the next prediction. In this way, randomness is selectively incorporated into the surviving logic. The sequence of predictor machines constitutes phyletic learning... inductive generalization... generation of new hypotheses concerning the relevant regularities found within the experienced environment, this in the light of the given payoff function. Prediction can be used for the purpose of identifying an unknown transducer. Feedback of the evolved model then forms a basis for control.

Rather than describe each organism only in terms of its behavior, Holland chose to evolve the code which generates such organisms. According to D.H. Ackley,[6] Holland's genetic algorithms search a parameter space where "any point in the parameter space can be represented as an n bit vector. The technique manipulates a set of such vectors to record information" about the parameter space. "There are two primary operations applied to the population by a genetic algorithm. Reproduction changes the contents of the population by adding copies of genotypes with above-average figures of merit. In addition to this...it is necessary to generate new, untested genotypes and add them to the population, else the population will simply converge on the best one it started with. Crossover is the primary mean of generating plausible new genotypes for addition to the population."

---

6- "A Connectionist Algorithm for Genetic Search," D. Ackley, in *Proceedings of an International Conference on Genetic Algorithms and Their Applications*, J.J. Grefenstette, Editor, Carnegie-Mellon University, July 1985.

As defined by Holland, crossover takes two structures, $A_1=a_{11}a_{12}...a_{1n}$ and $A_2 = a_{21}a_{22}...a_{2n}$, and at a random point "x" between 1 and n, exchanges the set of attributes to the right of this position yielding offspring of the form: $A^* = a_{11}a_{12}...a_{1x}a_{2(x+1)}...a_{2n}$. Ackley continues to state: "This 'offspring' is added to the population, displacing some other genotype according to various criteria where it has the opportunity to flourish or perish depending on its fitness." "Mutation provides a chance for any allele to be changed to another randomly chosen value." "If the mutation rate is too low, possibly critical alleles missing from the initial population will have only a small change of getting...into the population. However, if the probability of a mutation is not low enough, information... will be steadily lost to random noise."

Holland likened the actual code being mutated to that of the genetic code that defines a given organism. While Fogel, et al. only used small degrees of "background" mutation, Holland examined the operations of gene "crossover" and "inversion" among other actual biologic genetic recombinations. Although Holland's work went largely unnoticed for some time, today a great deal of attention is being given to genetic algorithms.

At the above referenced conference D. Goldberg and R. Lingle, Jr.[7] offered several observations:

"1) Simple genetic algorithms work well in problems which can be

---

7- "Alleles, Loci, and the Traveling Salesman Problem," D.E. Goldberg, R. Lingle, Jr., in *Proceedings of an International Conference on Genetic Algorithms and Their Applications*, J. J. Grefenstette, Editor, Carnegie-Mellon University, July 1985.

coded so the underlying building blocks (highly fit, short defining length schemata) lead to improved performance.

2) There are problems (more properly codings for problems) that are GA-hard -- difficult for the normal reproduction + crossover + mutation processes of the simple genetic algorithm.

3) Inversion is the conventional answer when genetic algorithmists are asked how they intend to find good string ordering, but inversion has never done much in empirical studies to date.

4) Despite numerous rumored attempts, the traveling salesman problem has not succumbed to genetic algorithm-like solution."

The authors then suggested a new type of crossover operator, "partially-mapped crossover (PMX)," that would lead to a more efficient solution of the traveling salesman problem.

Specifically, consider two possible codings of a tour of eight cities, $A_1$ and $A_2$, a return to the initial city being implicit:

$A_1$: 3 5 1 2 7 6 8 4

$A_2$: 1 8 5 4 3 6 2 7

PMX would proceed as follows: Two positions are determined randomly along the $A_1$ coding. The actual cities located between these positions along $A_1$ are exchanged with the cities located between the same

positions along $A_2$. For example, if the positions three and five are chosen, the sub-coding along $A_1$ is 1-2-7, and the sub-coding along $A_2$ is 5-4-3. Each of these cities is then exchanged, leading to the new tours, $A^*_1$ and $A^*_2$:

$A^*_1$: 7 1 5 4 3 6 8 2

$A^*_2$: 5 8 1 2 7 6 4 3.

They reported two experiments on ten cities where the PMX operator enabled the search to efficiently find either the absolute or near optimal solution. Goldberg and Lingle stated that this operator was more complex than "simple crossover," as proposed by Holland. As will be shown, in fact, it is not.

In another paper by J.J. Grefenstette, R. Gopal, B. Rosmaita and D. Van Gucht,[8] Holland's "simple crossover" was utilized. This required the formation of a special coding structure. Clearly, using this operator on two valid tours could result in an "offspring" that was not a valid tour. As A.K. Dewdney[9] related, the authors' method for devising the appropriate coding was ingenious.

---

8- "Genetic Algorithms for the Traveling Salesmen Problem," J.J. Grefenstette, R. Gopal, B. Rosmaita, D. Van Gucht, in *Proceedings of an International Conference on Genetic Algorithms and Their Applications*, J.J. Grefenstette, Editor, Carnegie-Mellon University, July 1985.

9- "Computer Recreations: Exploring the field of genetic algorithms in a primordial computer sea full of flibs," A.K. Dewdney, *Scientific American*, November 1985.

"The representation for a five-city tour such as *a, c, e, d, b* turns out to be 12321. To obtain such a numerical string reference is made to some standard order for the cities, say, *a, b, c, d, e*. Given a tour such as *a, c, e, d, b*, systematically remove cities from the standard list in the order of the given tour: remove *a*, then *c, e* and so on. As each city is removed from the special list, note its position just before removal, *a* is first, *c* is second, *e* is third, *d* is second and, finally *b* is first. Hence the chromosome 12321 emerges. Interestingly, when two such chromosomes are crossed over, the result is always a tour." Dewdney continued further to report that unfortunately the experiments with this representation were "not very encouraging." The authors conducted larger experiments than those of Goldberg and Lingle, including 50, 100 and 200 cities. In the three reported experiments, after a large number of trials (approximately 14000, 20000 and 25000, respectively), the best tours were still far away from the expected optimal solutions.

At this point it is natural to ask "why?". After all, the traveling salesman problem only requires discovery of a logical pattern. This seems completely analogous to what occurs in nature. If the crossover of genes works in natural evolution, why shouldn't it work here? The answer is, in fact, as noted in observation #2 by Goldberg and Lingle: The traveling salesman problem is GA-hard -- difficult for the normal reproduction + crossover + mutation. This is due to the fact that Holland's crossover operation does not mimic the biologic crossover of genes.

As defined by C.K. Levy,[10] crossover is the phenomenon where "old

---

10 - Elements of Biology, C.K. Levy, third edition, Addison-Wesley Publishing Company Inc., Reading Massachusettes, 1982.

linkages between genes on homologous chromosomes are broken, and new linkages are established. Genes that reside on the same chromosome and move together are said to be 'linked.' A linkage group is any group of genes physically linked on one chromosome." Levy goes on to state: "Changes in linkage groups are not truly mutations, however, since neither the amount nor the function of genetic material is altered."*

Crossover allows for different combinations of alleles. Alleles, by definition, control the same characteristic and occupy the same place on similar chromosomes. Holland's crossover treats the entire tour as a chromosome and each city in a tour as a gene. While Holland's crossover does not change the amount of coding, it greatly alters the function of the coding! A more appropriate biologic interpretation of a tour would be that it is itself a gene. Crossover inside a gene is a nonsequetor. The tour is absolutely not analogous to a chromosome and each city in a tour is not analogous to a gene. These relations are in fact anomolous.

The result of Holland's crossover, therefore, is a near random search throughout the entire space of possible tours. Perhaps Dewdney said it best when he stated that by using crossover "there is so much juggling of genes and cracking of chromosomes that...(a parent)...is hard put to recognize its own grandchildren." This is, of course, the very essence of the difficulty! Adaptive plans must retain already made advances in order to ensure that an optimal solution will be found. As the number of cities grows larger, Holland's crossover effectively destroys the link between each parent and its offspring. The results can even be worse than a

---

*- Underline added by this author.

complete enumeration of all possible tours (reference the Addendum).

## AN ALTERNATIVE APPROACH

As an alternative solution, consider the Adaptive Algorithm, so named because it does not include any of the pseudo-genetic operators that Holland has suggested. In this algorithm, which is equivalent to evolutionary programming restricted to single state machines, only slight mutations are made to an existing tour by removing just one city from a given list and replacing it in a different randomly chosen position. This mutation is only slightly more complicated than the simplest possible mutation...swapping adjacent cities. It is clearly less complex than both the PMX operator and Holland's crossover. Through multiple mutation, this single alteration can be made equivalent to either of these crossover operators.

According to Holland: "If successive populations are produced by mutation alone (without (genetic) reproduction), the result is a random sequence of structures drawn from (all possible structures)."[11] This is only partly correct. The Adaptive Algorithm does result in a random search, but only in a portion of the space relatively close to the parent that generates the offspring. This increases the effectiveness of the algorithm dramatically as it allows for the retention of advances.

---

11- Adaptation in Natural and Artificial Systems, J.H. Holland, University of Michigan Press, Ann Arbor, 1975.

But more is still required. Not only must advances be retained but "dead-ends" must be circumvented. Because there is a finite number of offspring that can be generated through mutation, there might well be stagnation on a local optimum. To prevent this it is useful to randomly alter the adaptive topography* that is being searched. This can be accomplished in various ways. One of these is to occasionally allow for the survival of offspring that are slightly worse than their parents. In effect, the scoring function is made "noisy." What results is analogous to the searching of a maze; when a dead-end is reached some backtracking is allowed and the overall search is reinitiated.

Unfortunately, the topography is much like an upside-down bed of nails, with some nails being longer (better) than others. From any given nail, it is possible to travel to any of n(n-1) other nails in a single mutation. Thus, unlike a maze, when the evolving phyletic line reaches a non-optimal nail from which no single mutation results in a better tour, it is impossible to determine the "direction" in which to backtrack. A given nail can be reached in a multitude of ways; therefore, bactracking is allowed in any direction. To further aid in the circumvention of stagnation, the degree of "poorer quality" offsping that are occasionally accepted should gradually be reduced as the adaptive process proceeds.

## EXPERIMENTAL FINDINGS

Experiments were performed to demonstrate the effectiveness of the

---

* - "Adaptive Topography" refers to the scoring function on the hyperspace of possible "organisms."

Adaptive Algorithm. Initially, 128 independent trials were performed on a 24 city traveling salesman problem where the cities were positioned on the periphery of a rectangle. Clearly, the minimum length tour is equal to the perimeter of the rectangle, here, 250. Of the 128 trials, 90.625% found the optimum solution in an average of 5297.48 iterations, see Figure 1, the maximum number of iterations being arbitrarily set at 14,000. Figure 2 indicates the results of the remaining 9.375% of the trials. Here, at least temporarily, the evolving tours were trapped on a local optimum. Despite the seemingly non-complex arrangement of cities, the numerous local optima make this traveling salesman problem difficult.

The cities were then distributed at random. First, 19 experiments were conducted requiring a tour of 50 cities. The cities were redistributed for each experiment. In each, no optimum tours were discovered in 20,000 iterations, but it was clear that the evolutionary process was "solving the problem." Figures 3 through 21 indicate the results of each experiment. Figure 22 indicates a typical example of the evolutionary process discovering more and more suitable tours as offspring are evaluated. Note that "backtracking" plays an integral part of the search.

Experiments were then performed requiring a tour of 100 cities under similar conditions. Again, while none of the eight experiments found a perfect tour, the evolutionary process performed well. Figures 23 through 30 indicate the results of the eight experiments while Figure 31 indicates the reduction in tour length as offspring are evaluated.

Example of Successful Discovery of the Optimal Solution in Series #1

Experiment #81

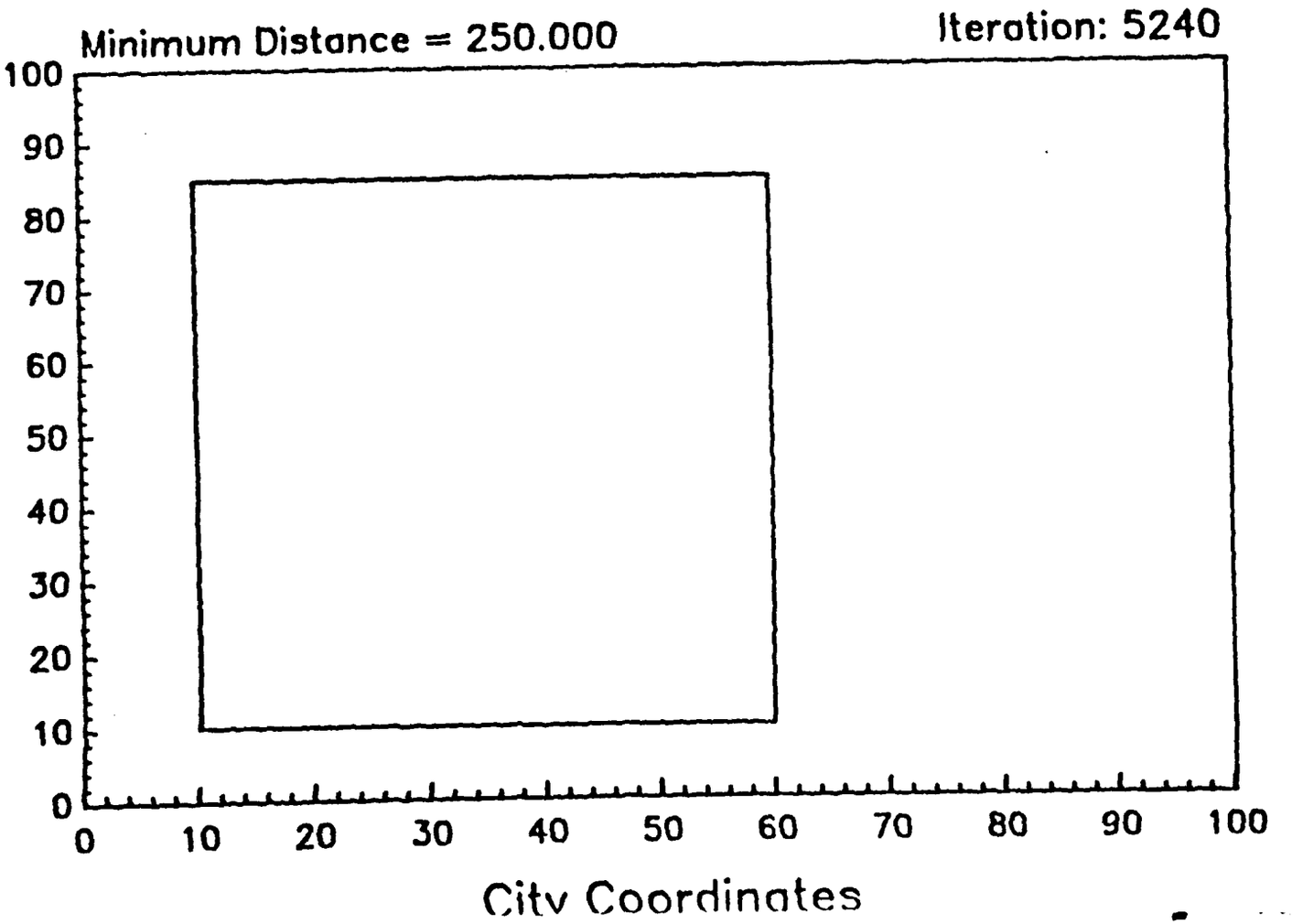


Figure 1

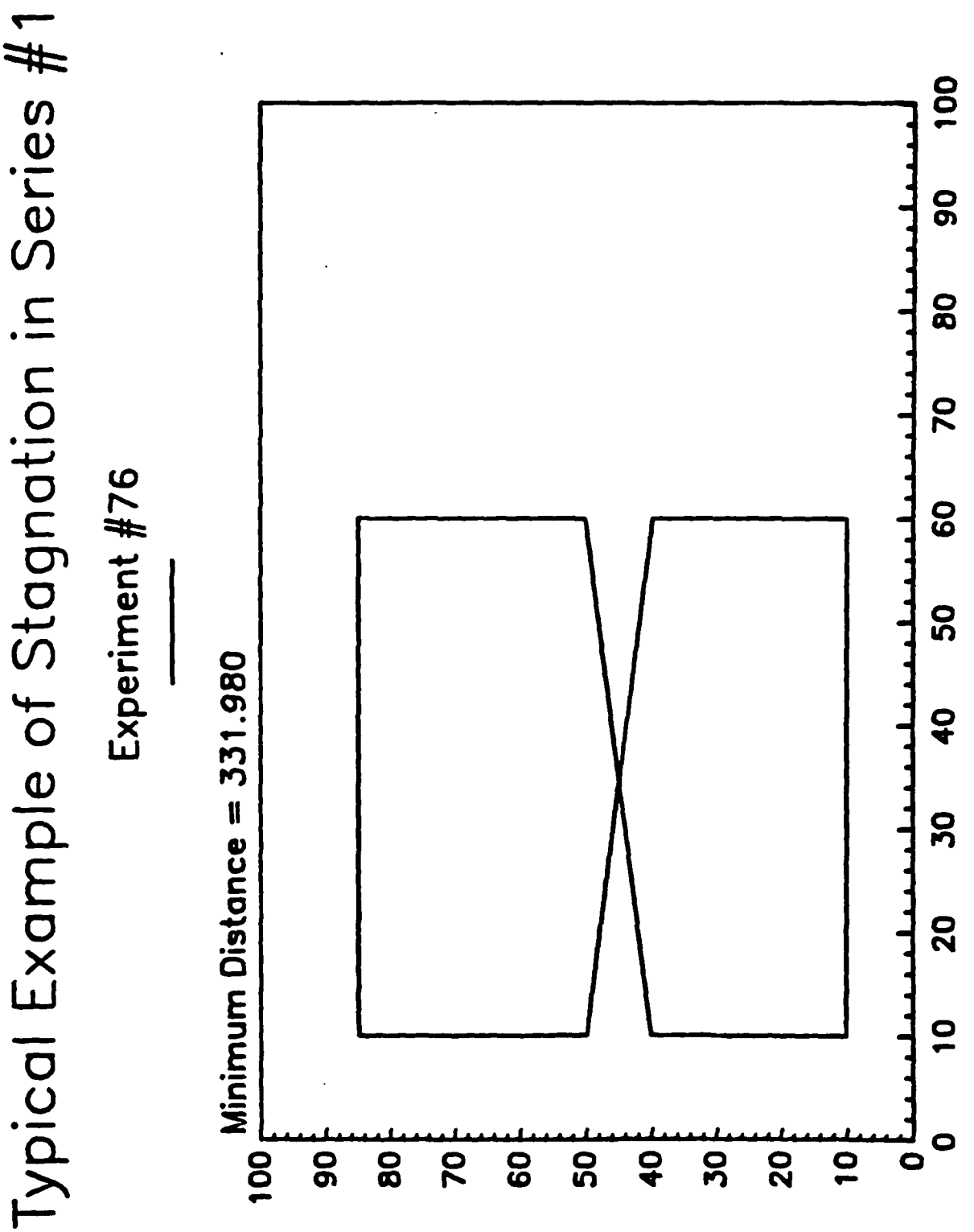
Typical Example of Stagnation in Series #1
Experiment #76
Minimum Distance = 331.980
100
90
80
70
60
50
40
30
20
10
0
0
10
20
30
40
50
60
70
80
90
100
City Coordinates

Figure 2

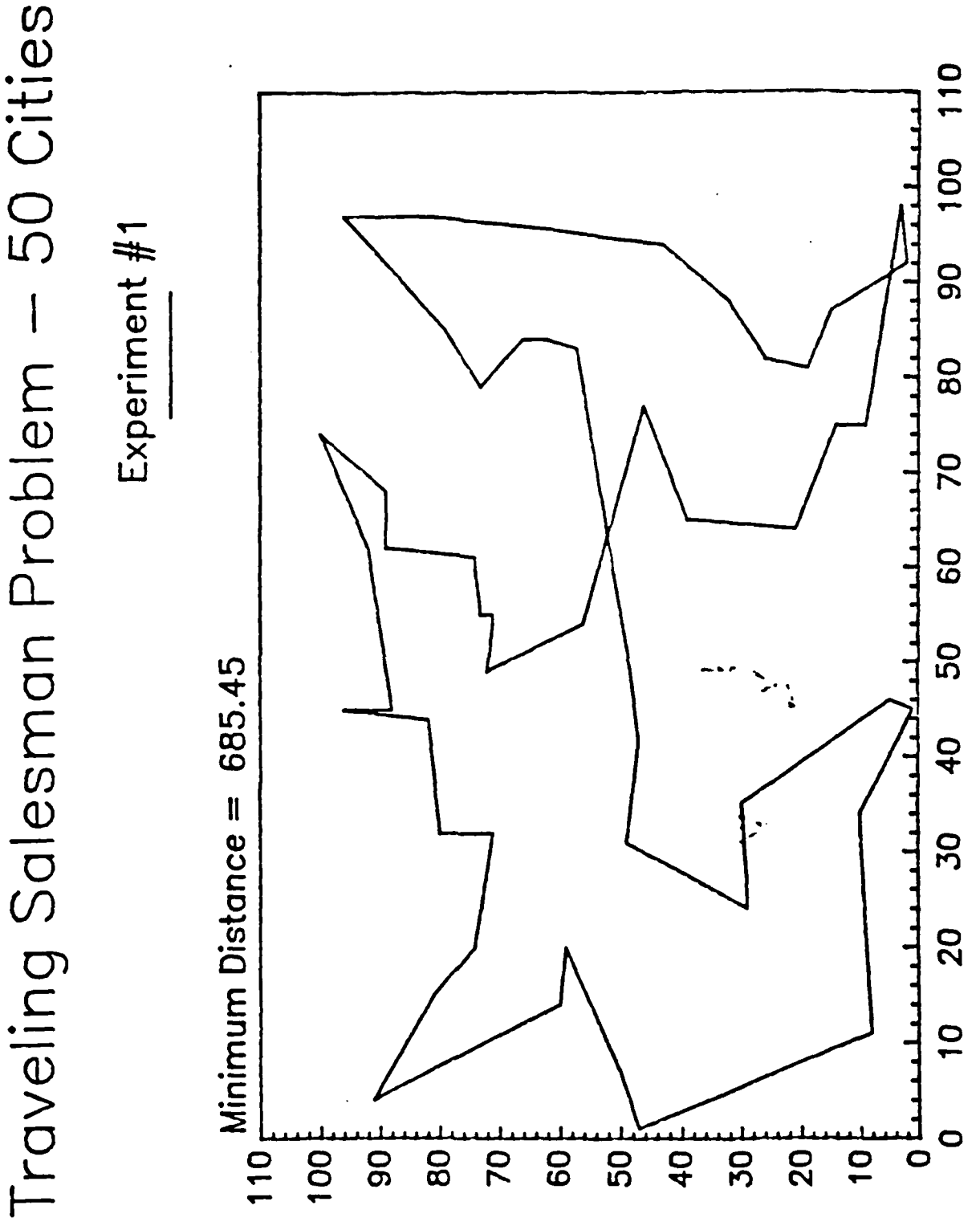
Traveling Salesman Problem – 50 Cities
Experiment #1
Minimum Distance = 685.45
110
100
90
80
70
60
50
40
30
20
10
0
0
10
20
30
40
50
60
70
80
90
100
110
City Coordinates

Figure 3

# Traveling Salesman Problem – 50 cities

## Experiment #2

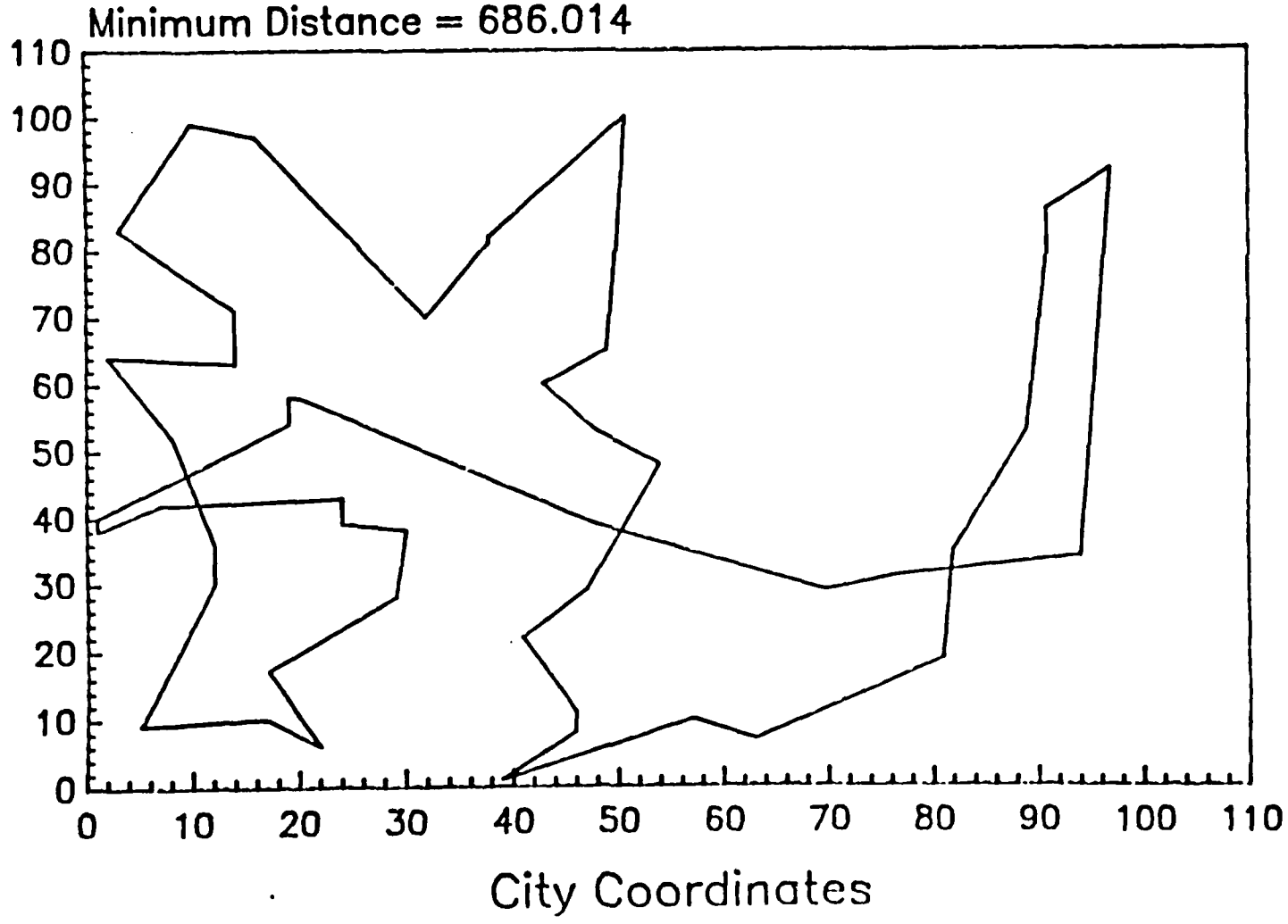


Figure 4

# Traveling Salesman Problem – 50 cities

Experiment #3

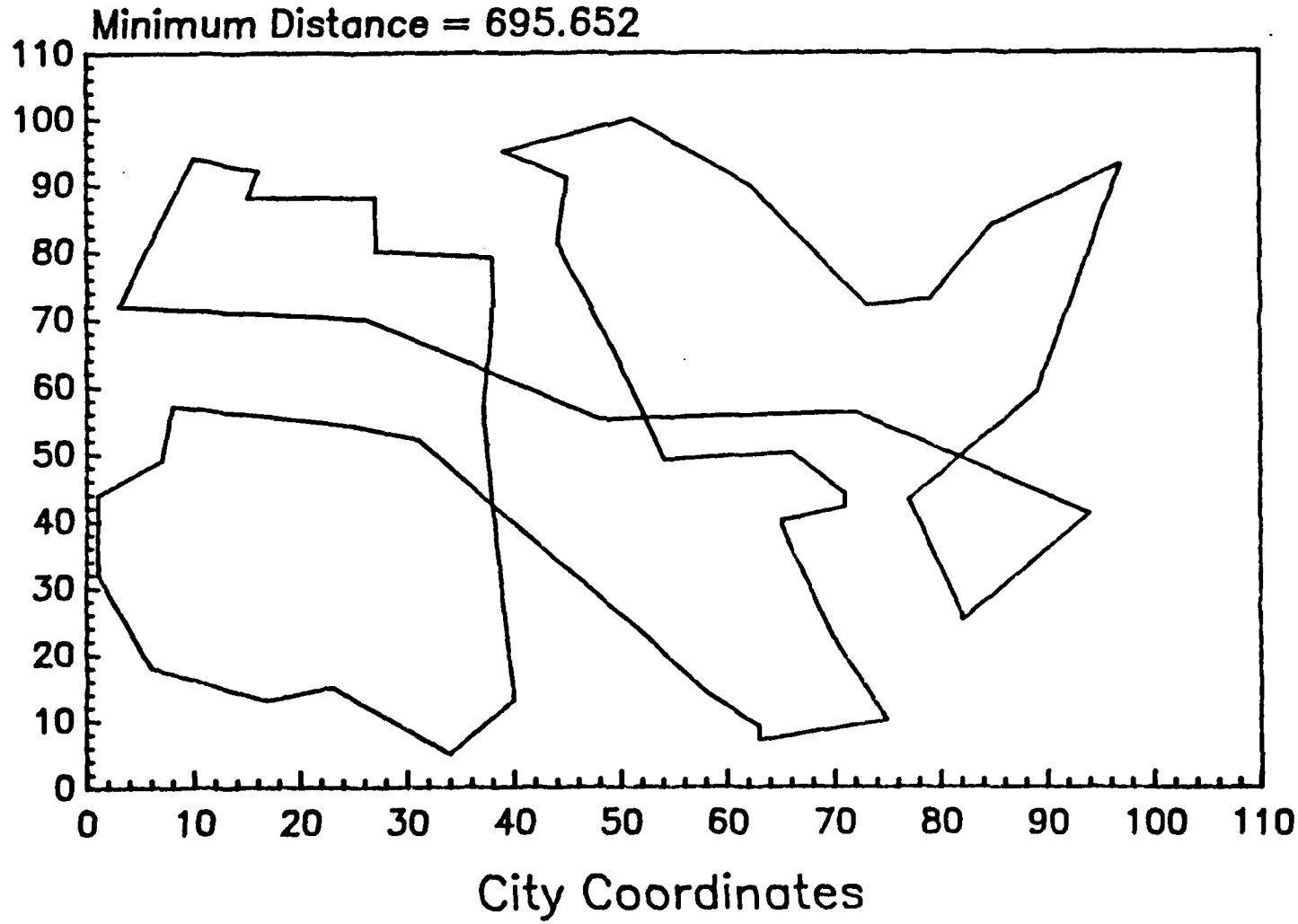


Figure 5

# Traveling Salesman Problem – 50 cities

## Experiment #4

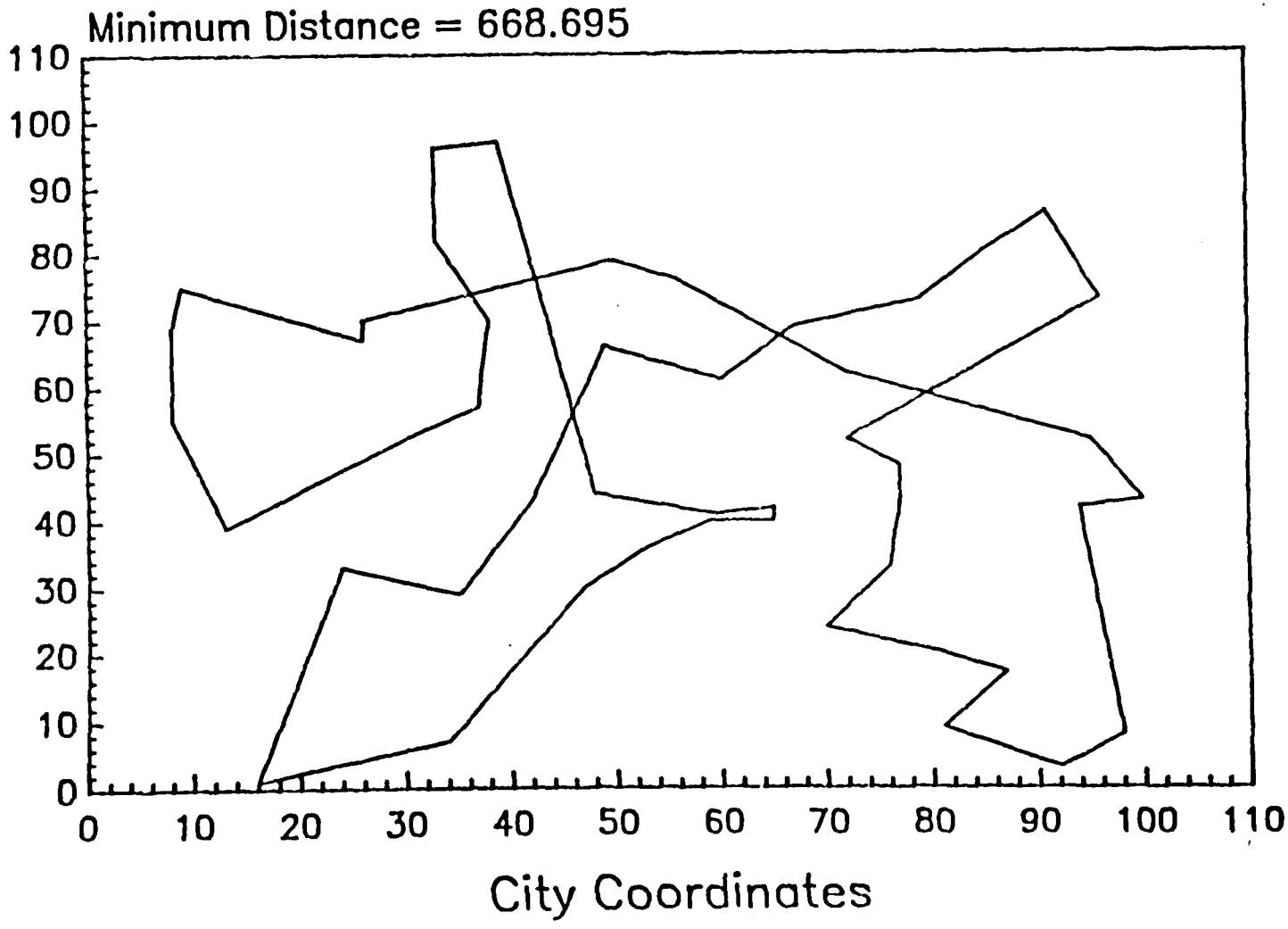


Figure 6

# Traveling Salesman Problem – 50 cities

Experiment #5

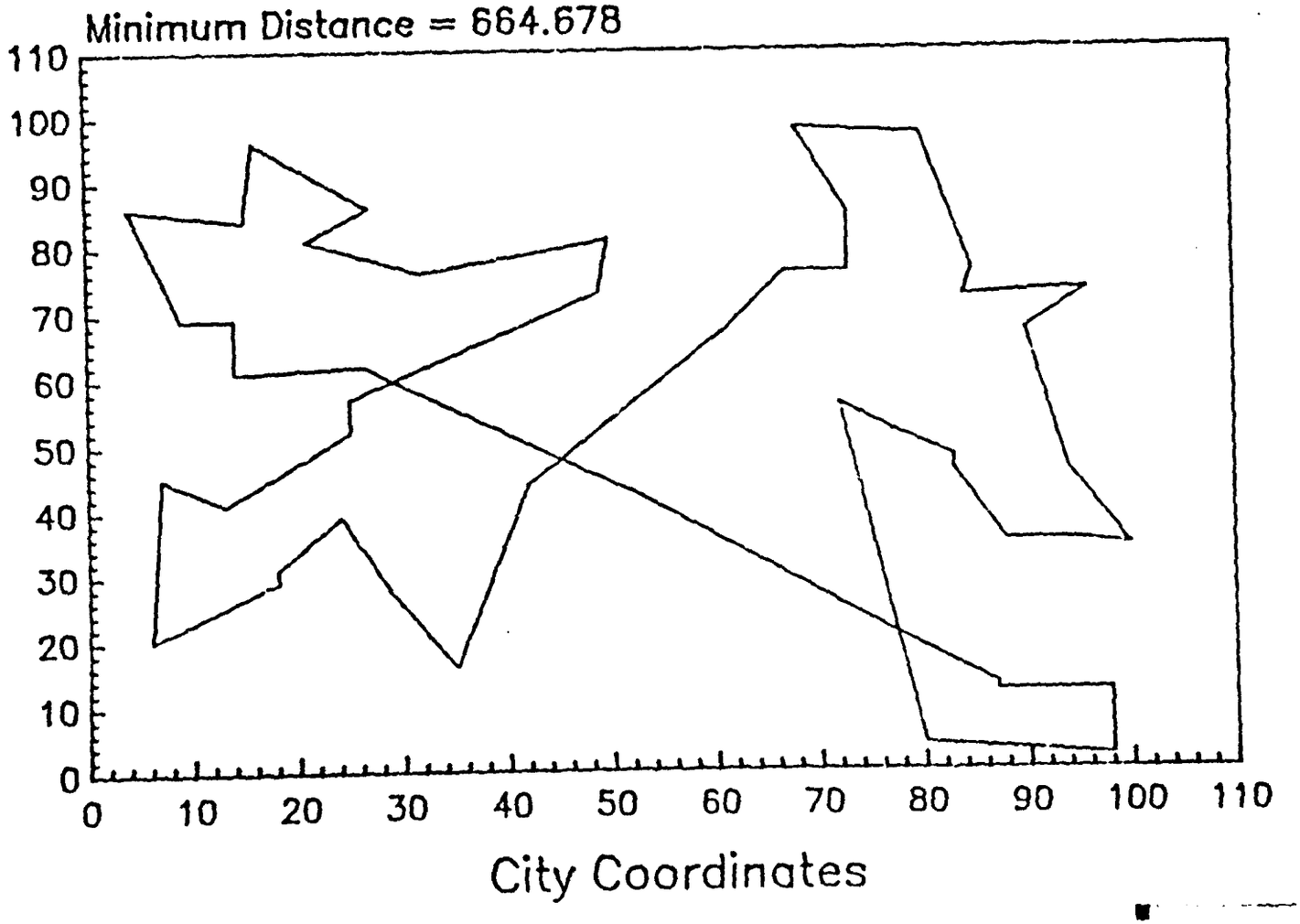


Figure 7

# Traveling Salesman Problem – 50 cities

## Experiment #6

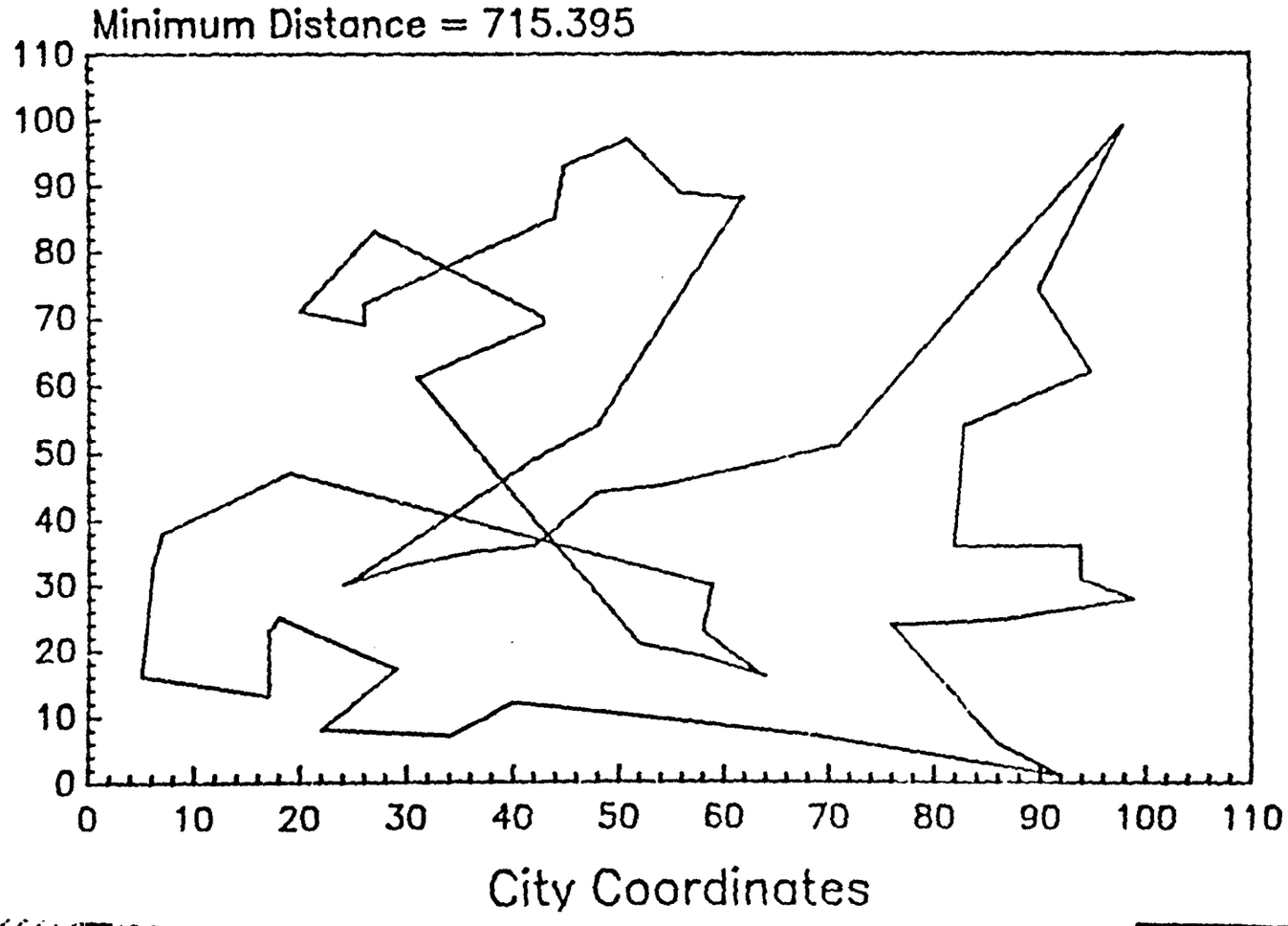


Figure 8

# Traveling Salesman Problem – 50 cities

Experiment #7

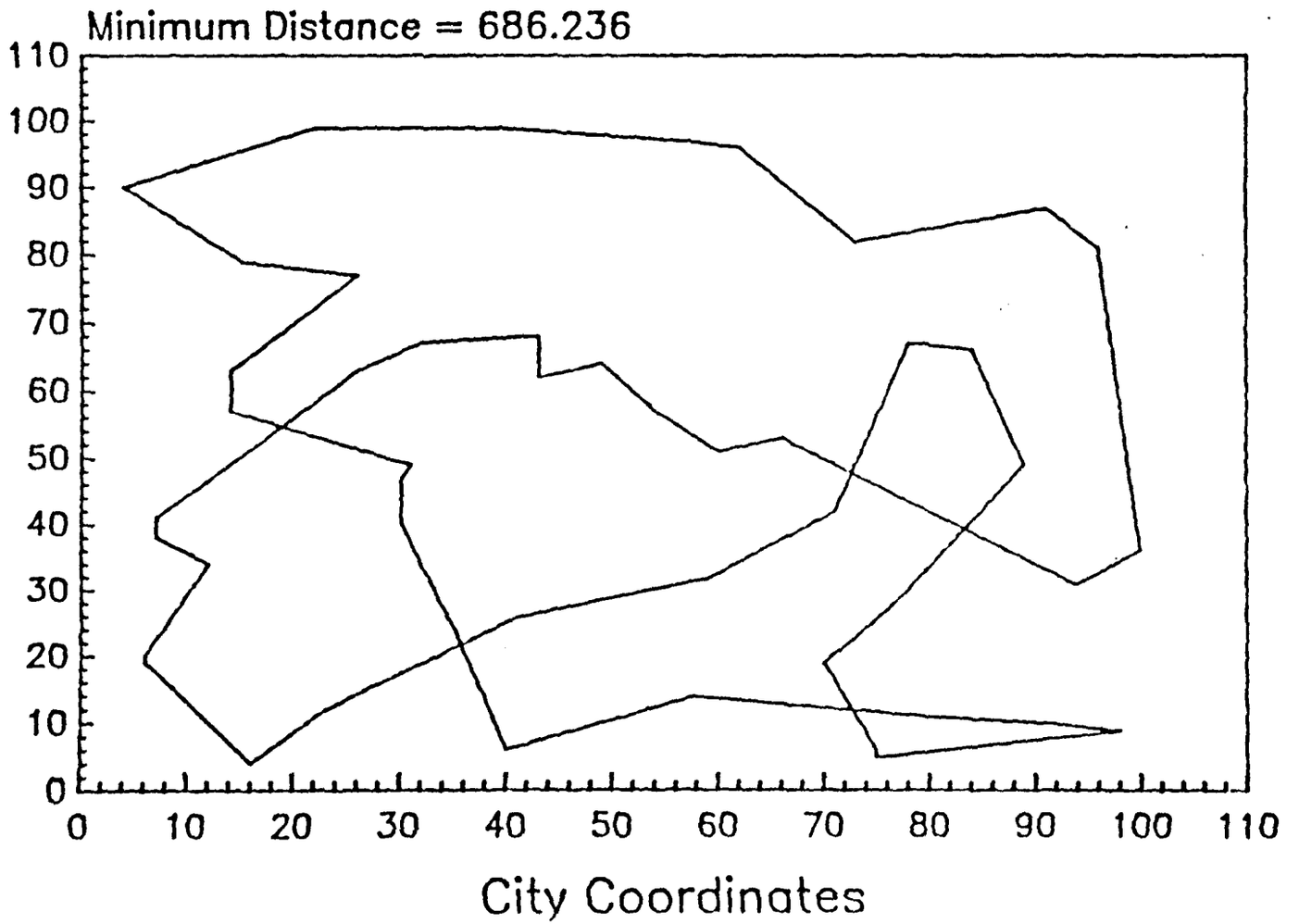


Figure 9

# Traveling Salesman Problem – 50 cities

## Experiment #8

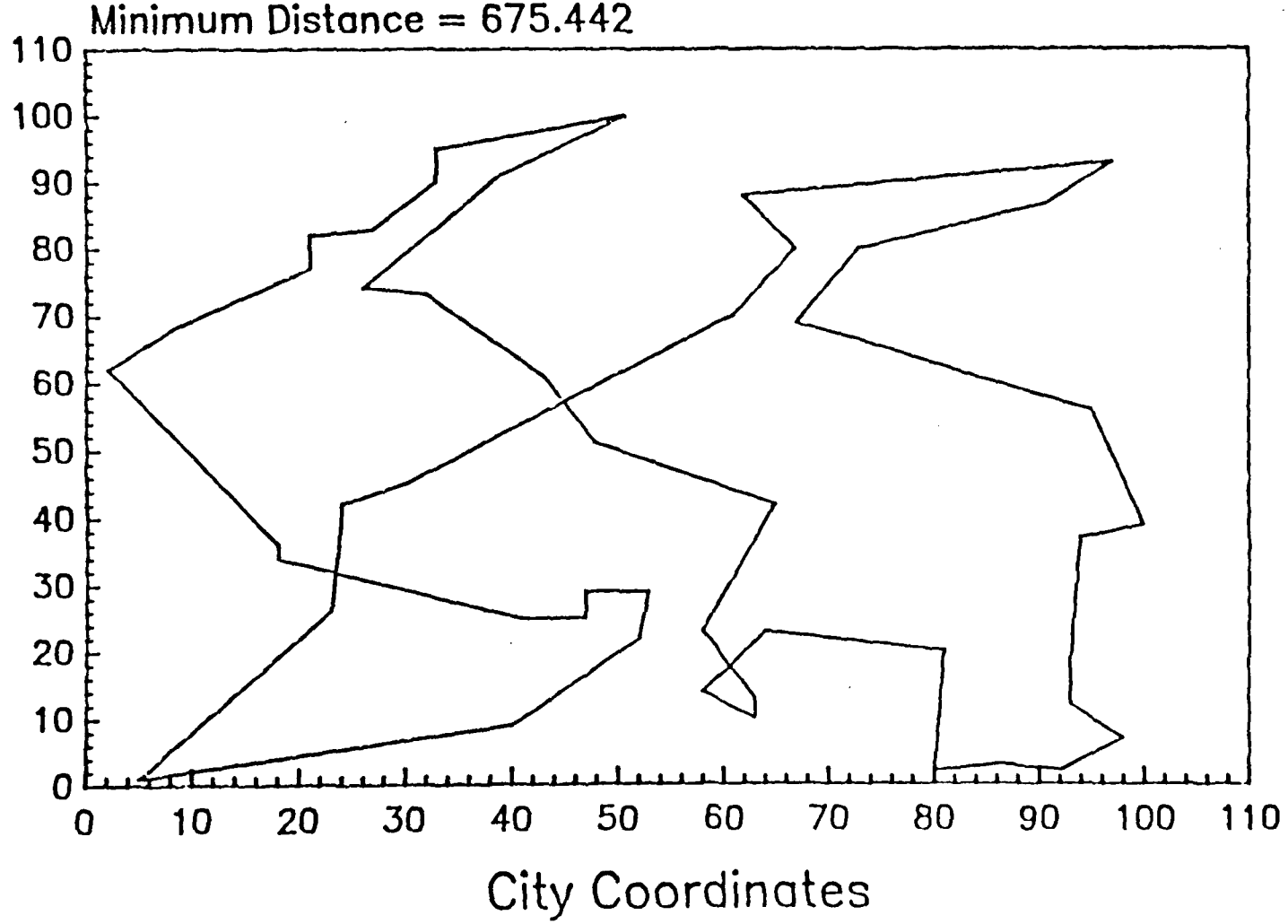


Figure 10

# Traveling Salesman Problem – 50 cities

## Experiment #9

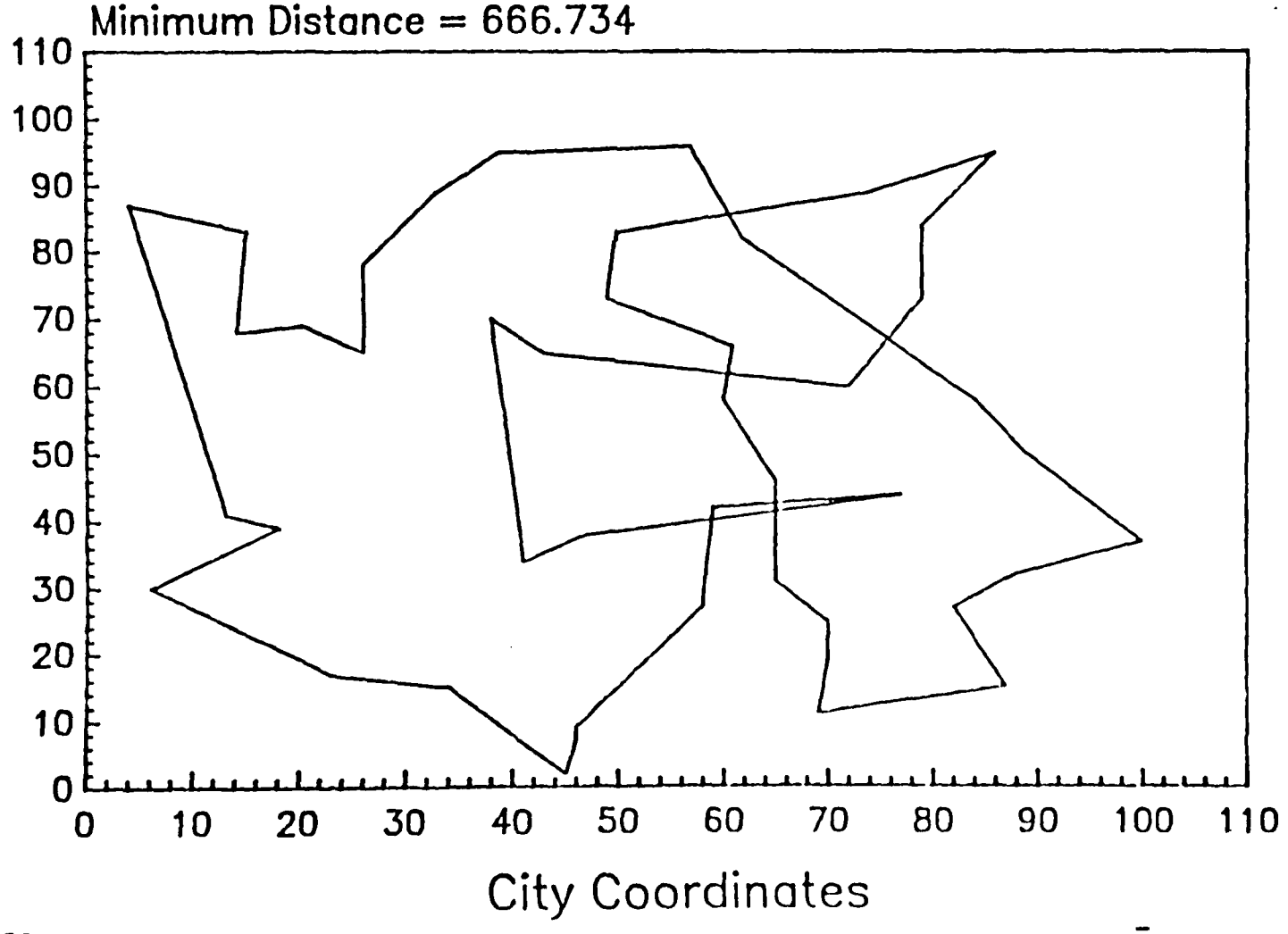


Figure 11

# Traveling Salesman Problem – 50 cities

Experiment #10

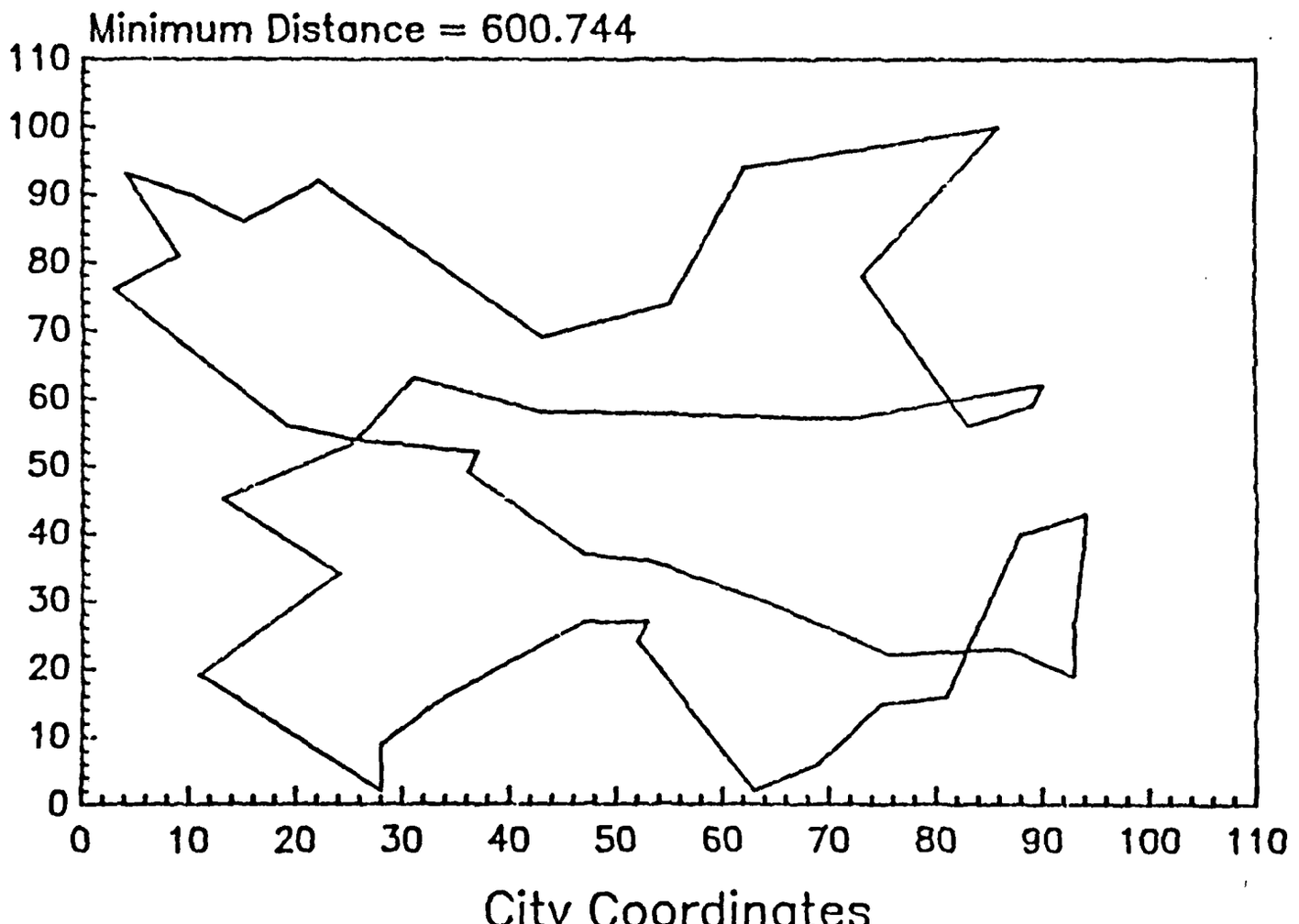


Figure 12

# Traveling Salesman Problem – 50 cities

Experiment #11

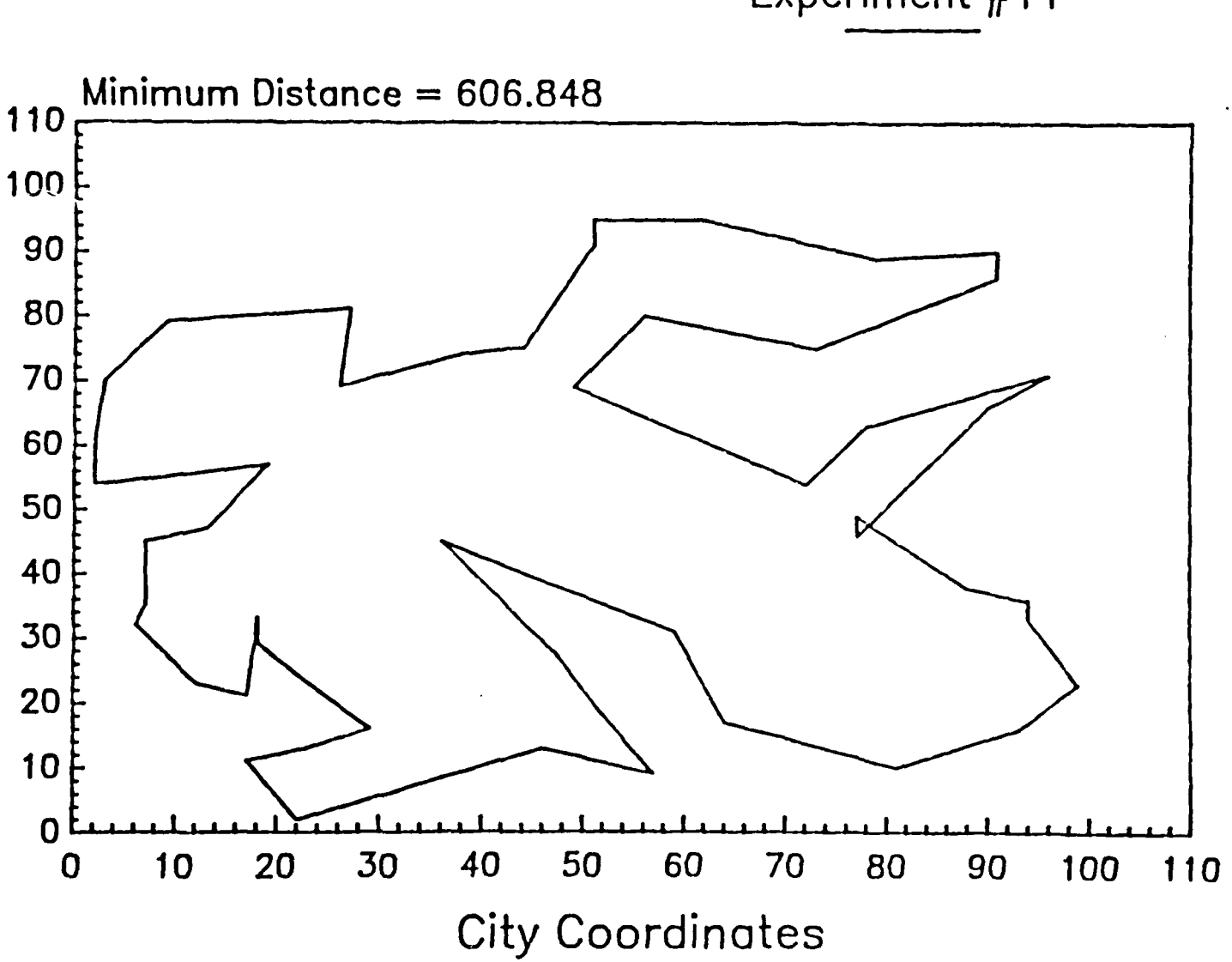


Figure 13

# Traveling Salesman Problem – 50 cities

Experiment #12

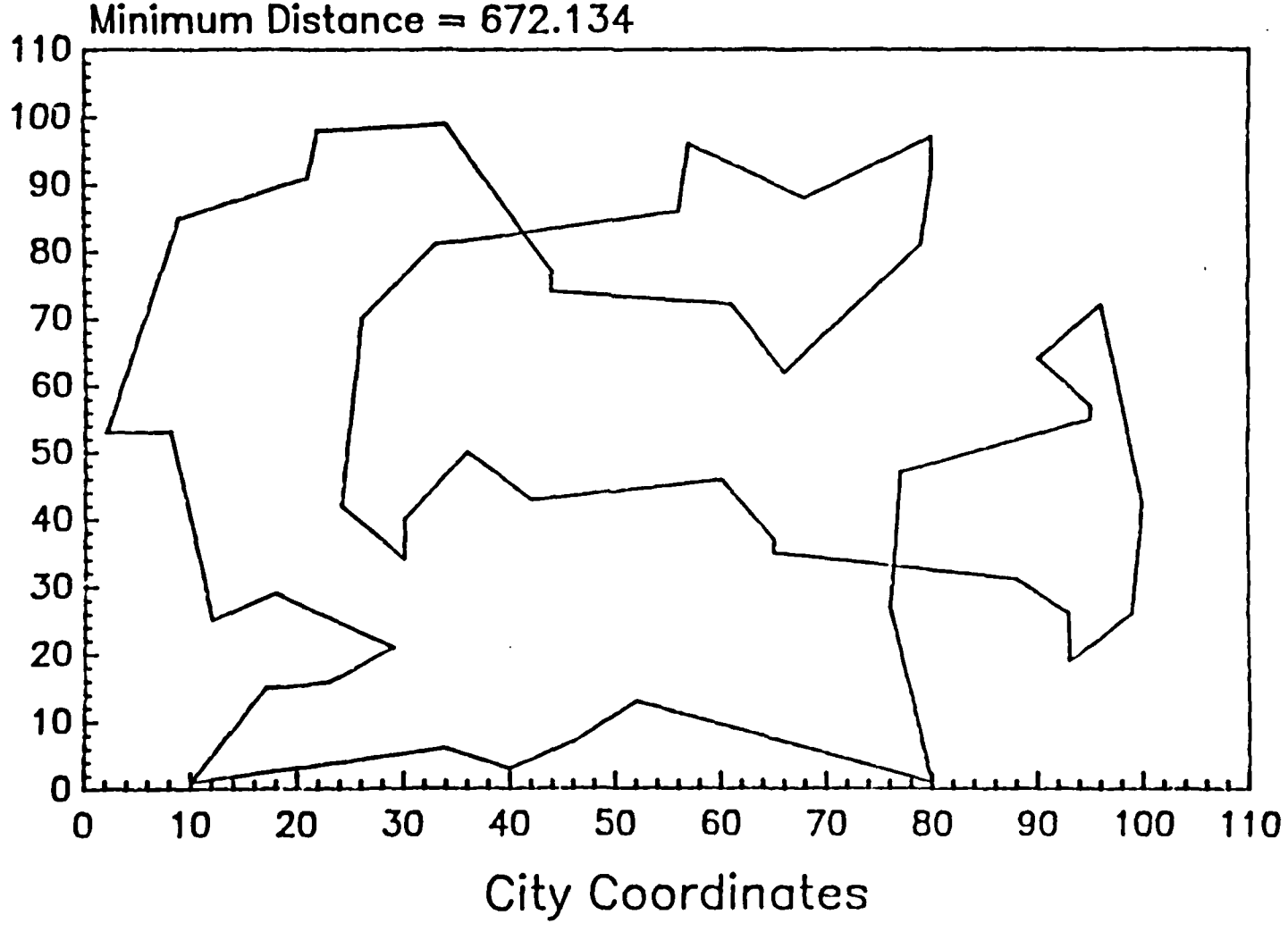


Figure 14

# Traveling Salesman Problem – 50 cities

Experiment #13

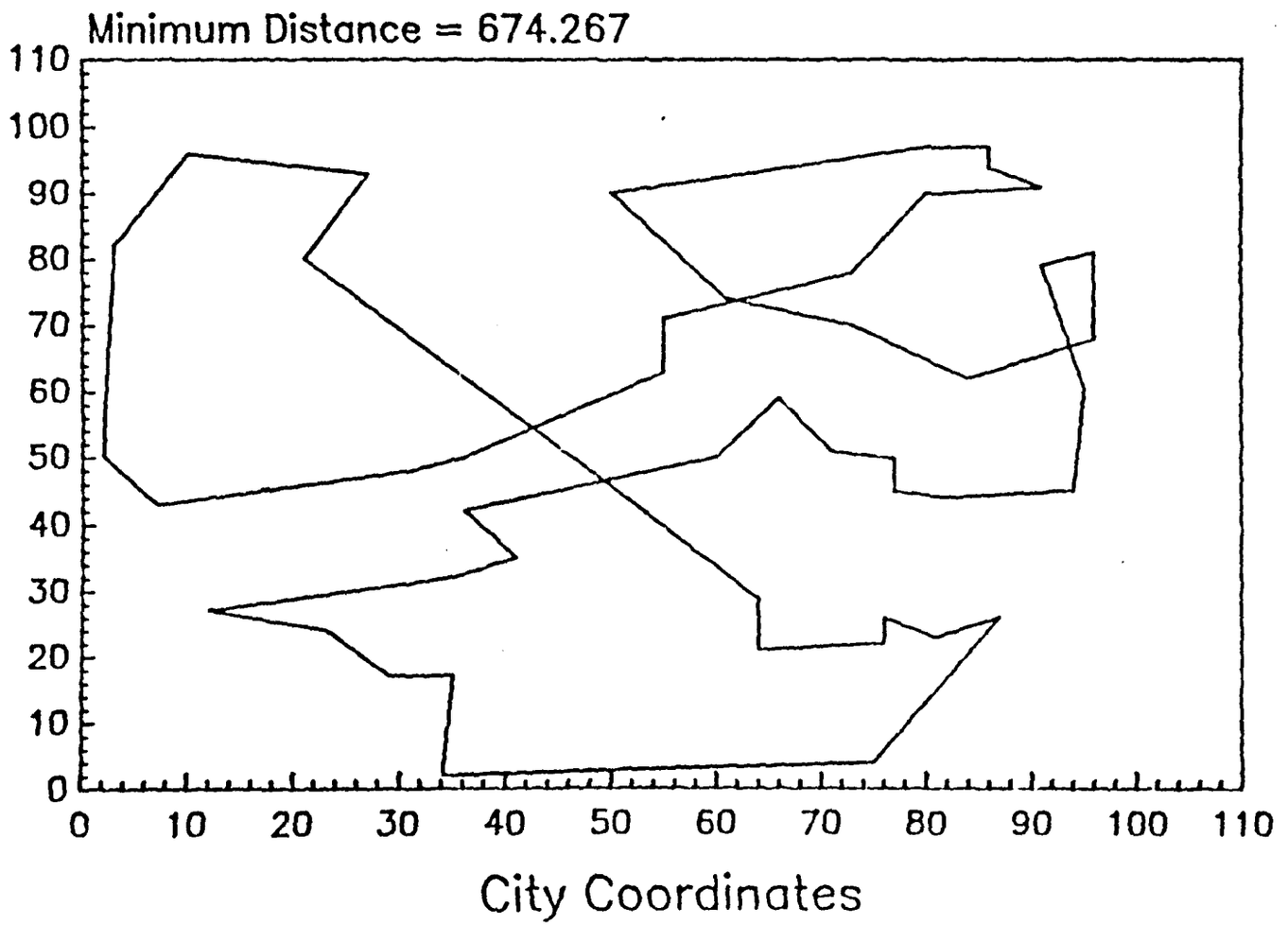


Figure 15

# Traveling Salesman Problem – 50 cities

## Experiment #14

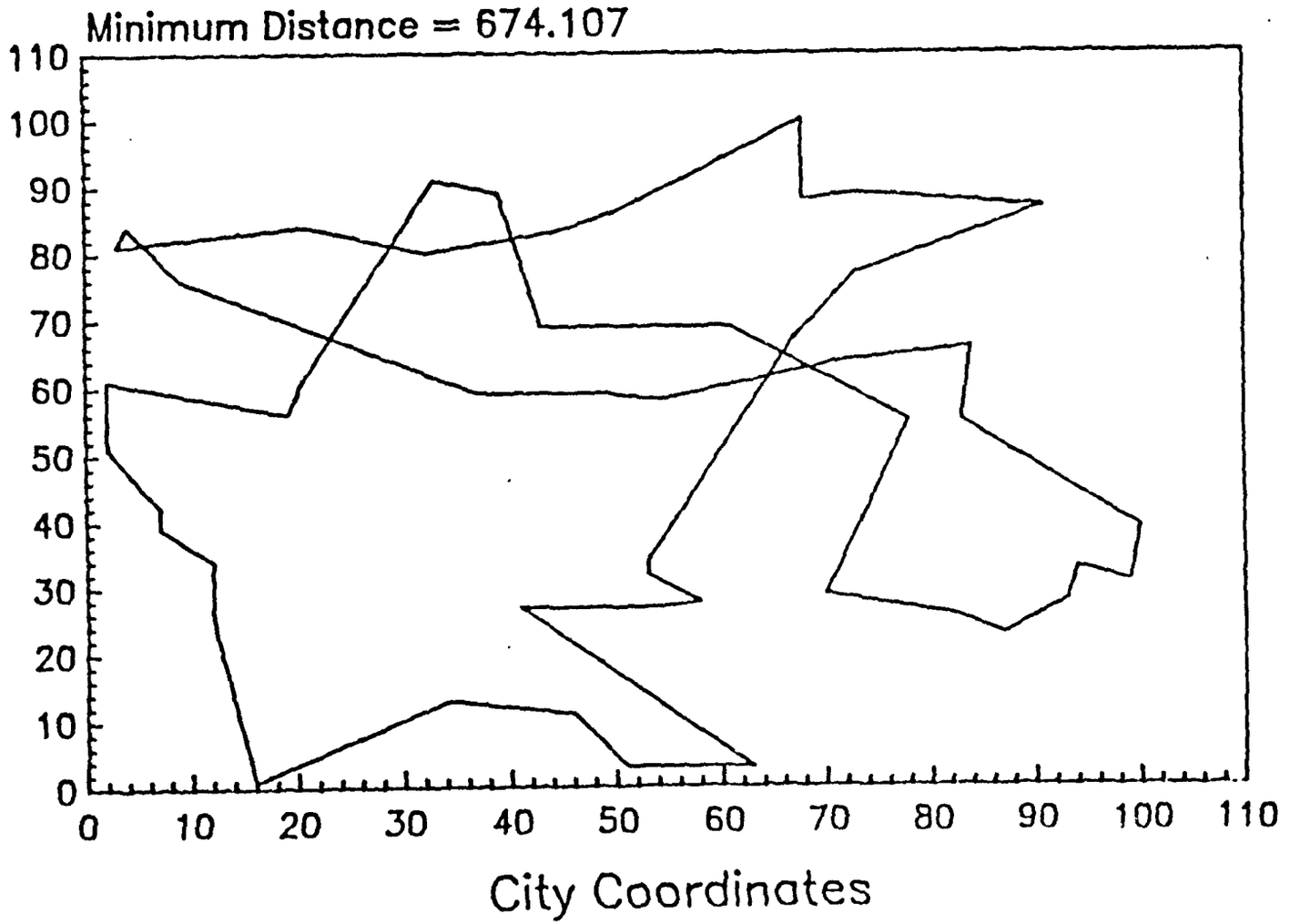


Figure 16

# Traveling Salesman Problem – 50 cities

Experiment #15

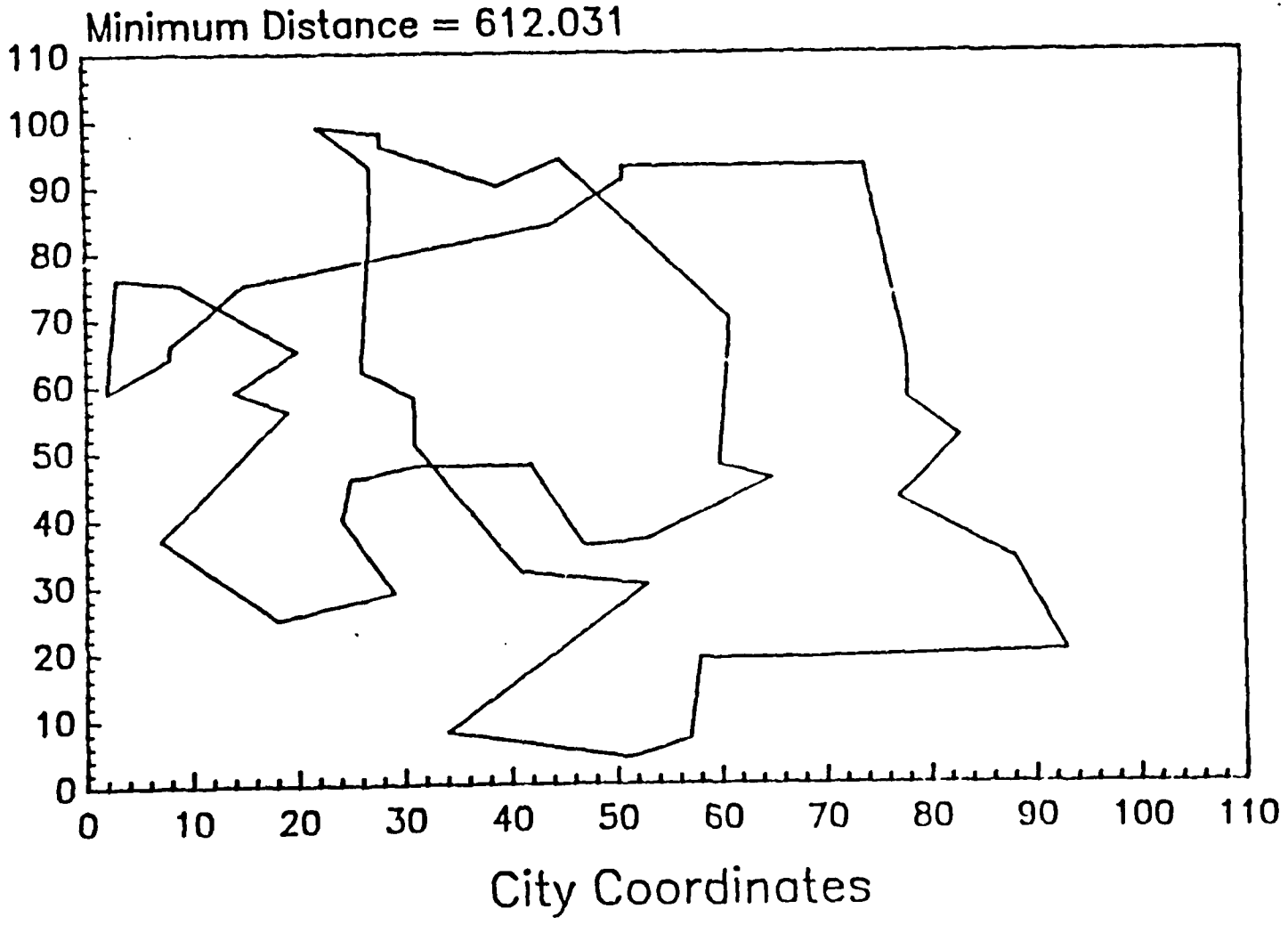


Figure 17

# Traveling Salesman Problem – 50 cities

Experiment #16

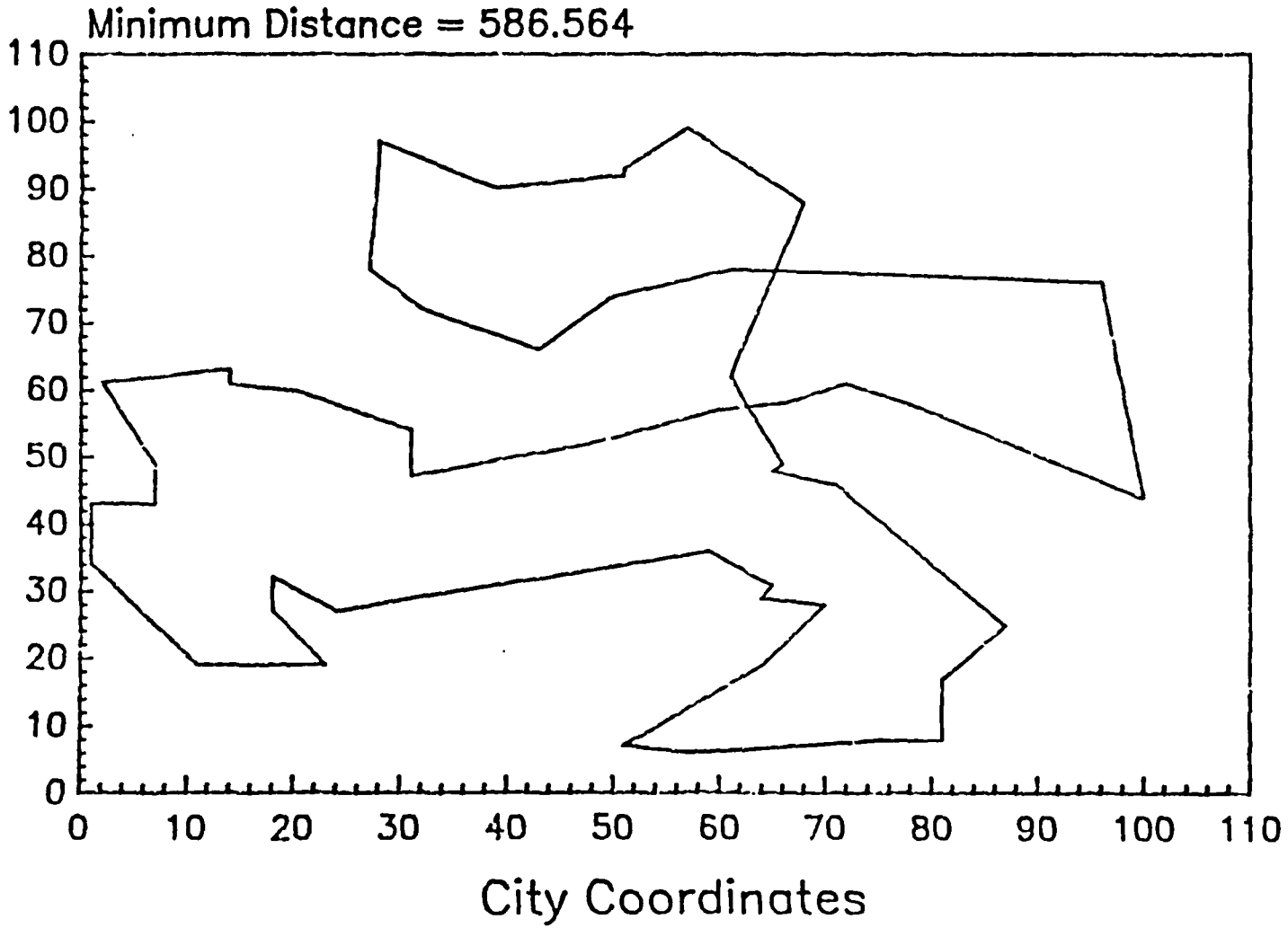


Figure 18

# Traveling Salesman Problem – 50 cities

Experiment #17

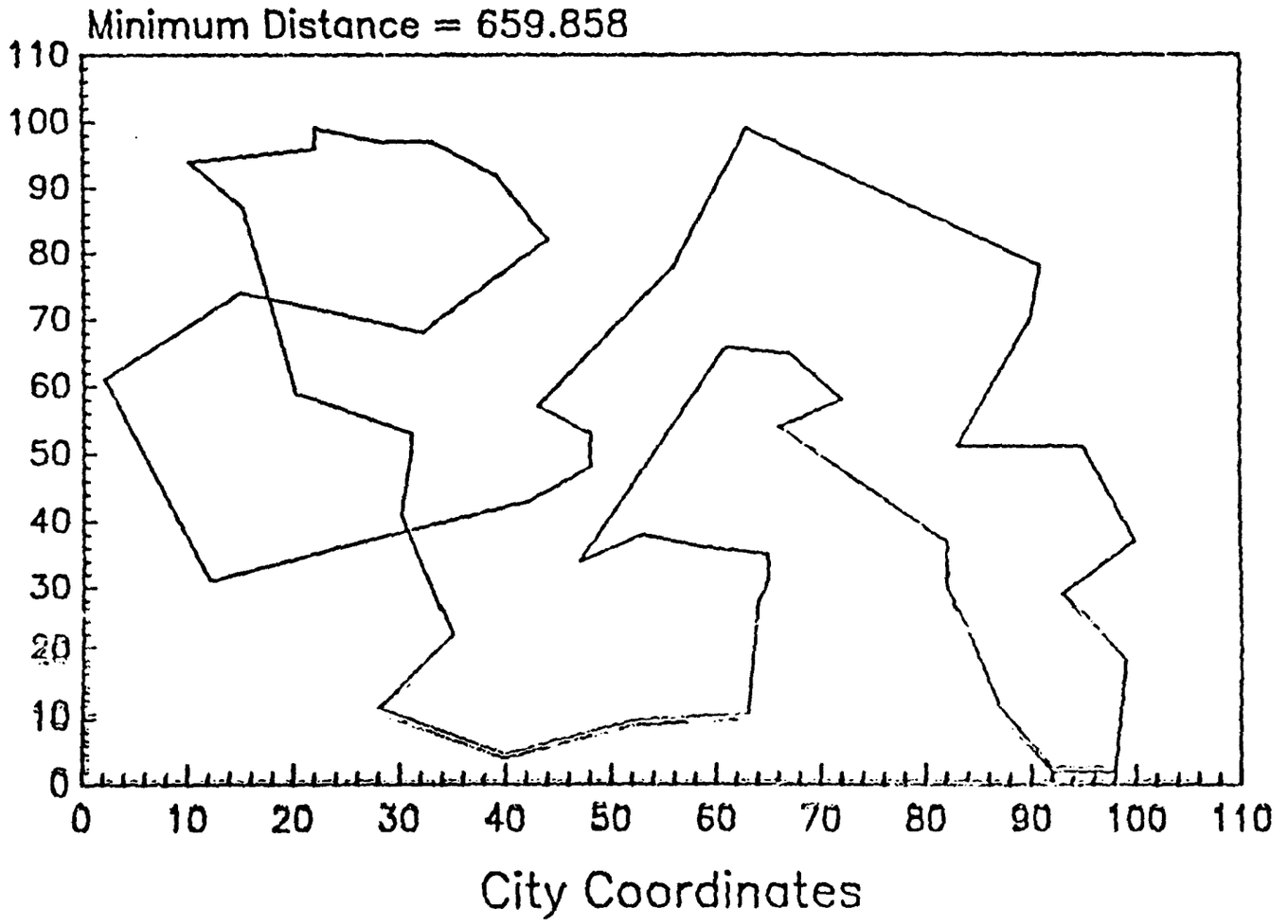


Figure 19

# Traveling Salesman Problem – 50 cities

Experiment #18

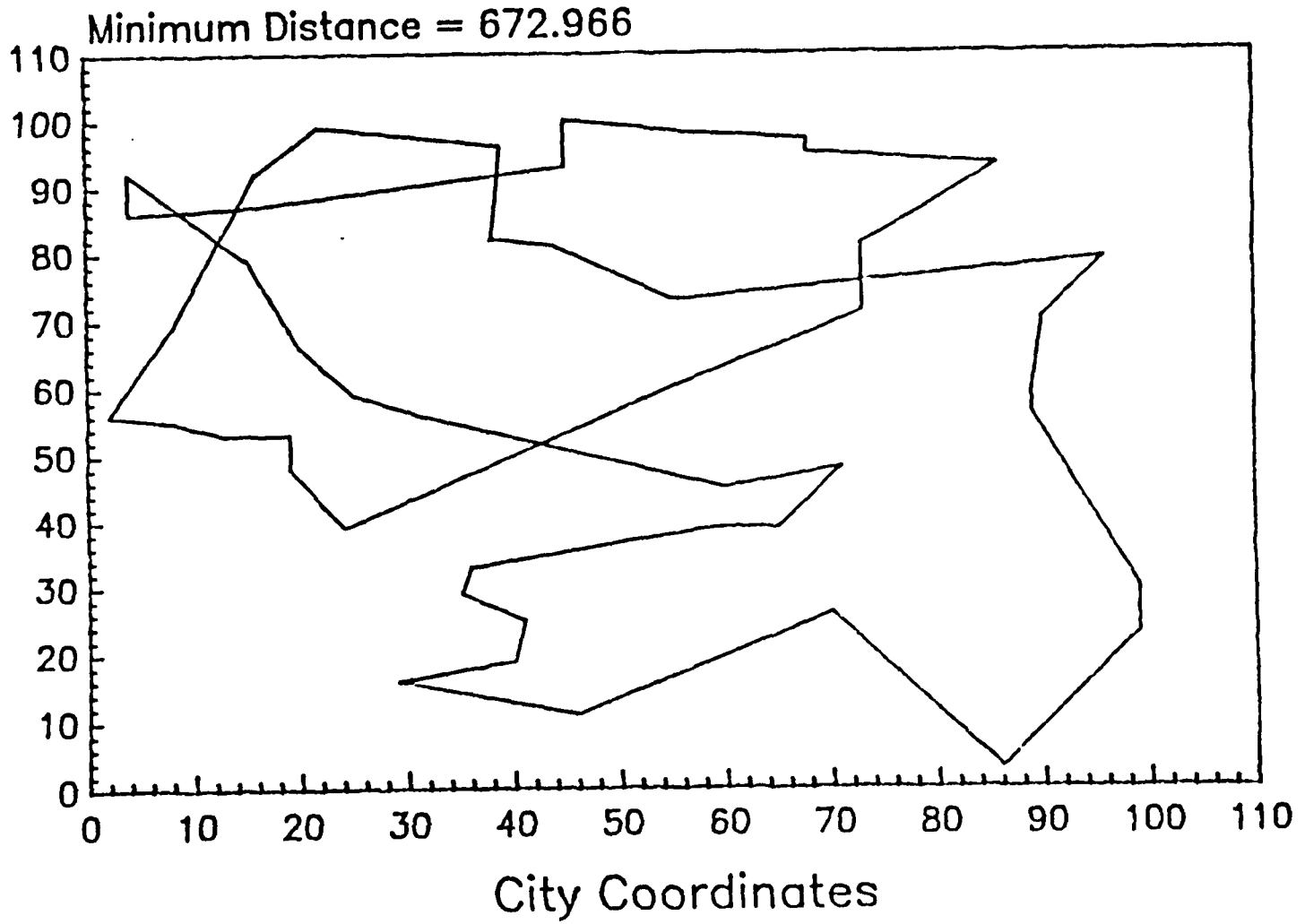


Figure 20

# Traveling Salesman Problem – 50 cities

Experiment #19

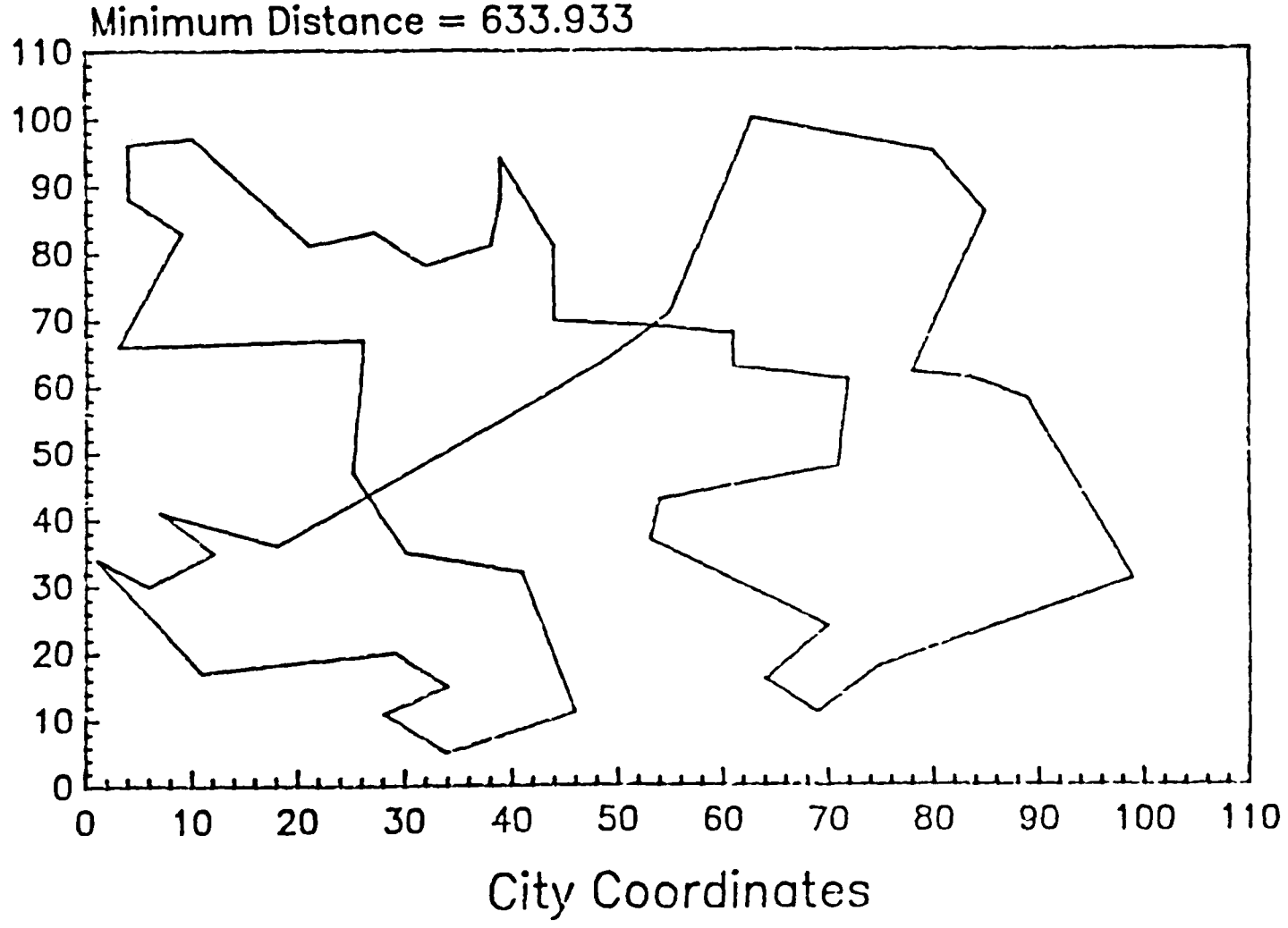


Figure 21

# Traveling Salesman Problem – 50 cities

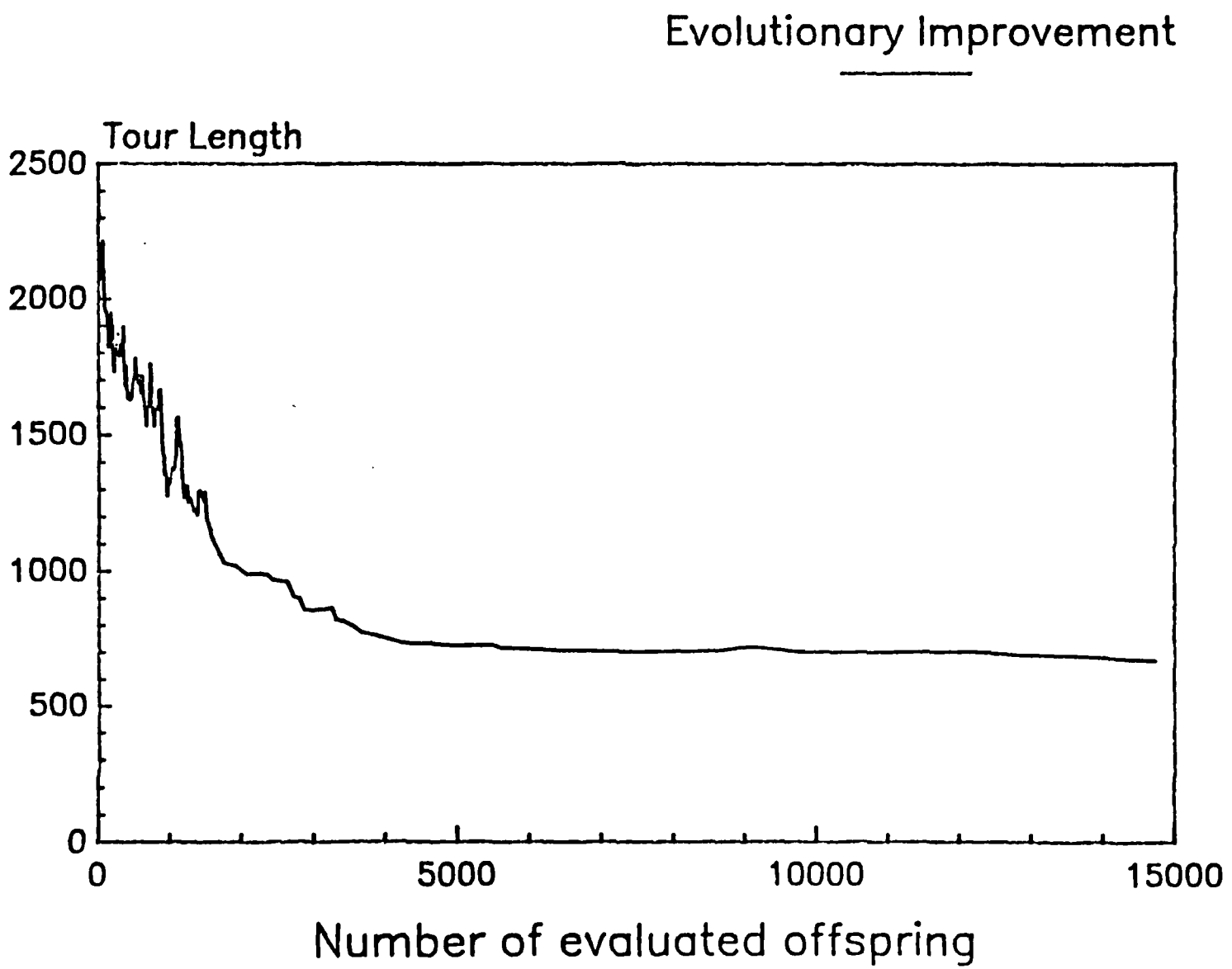


Figure 22

# Traveling Salesman Problem – 100 cities

## Experiment #1

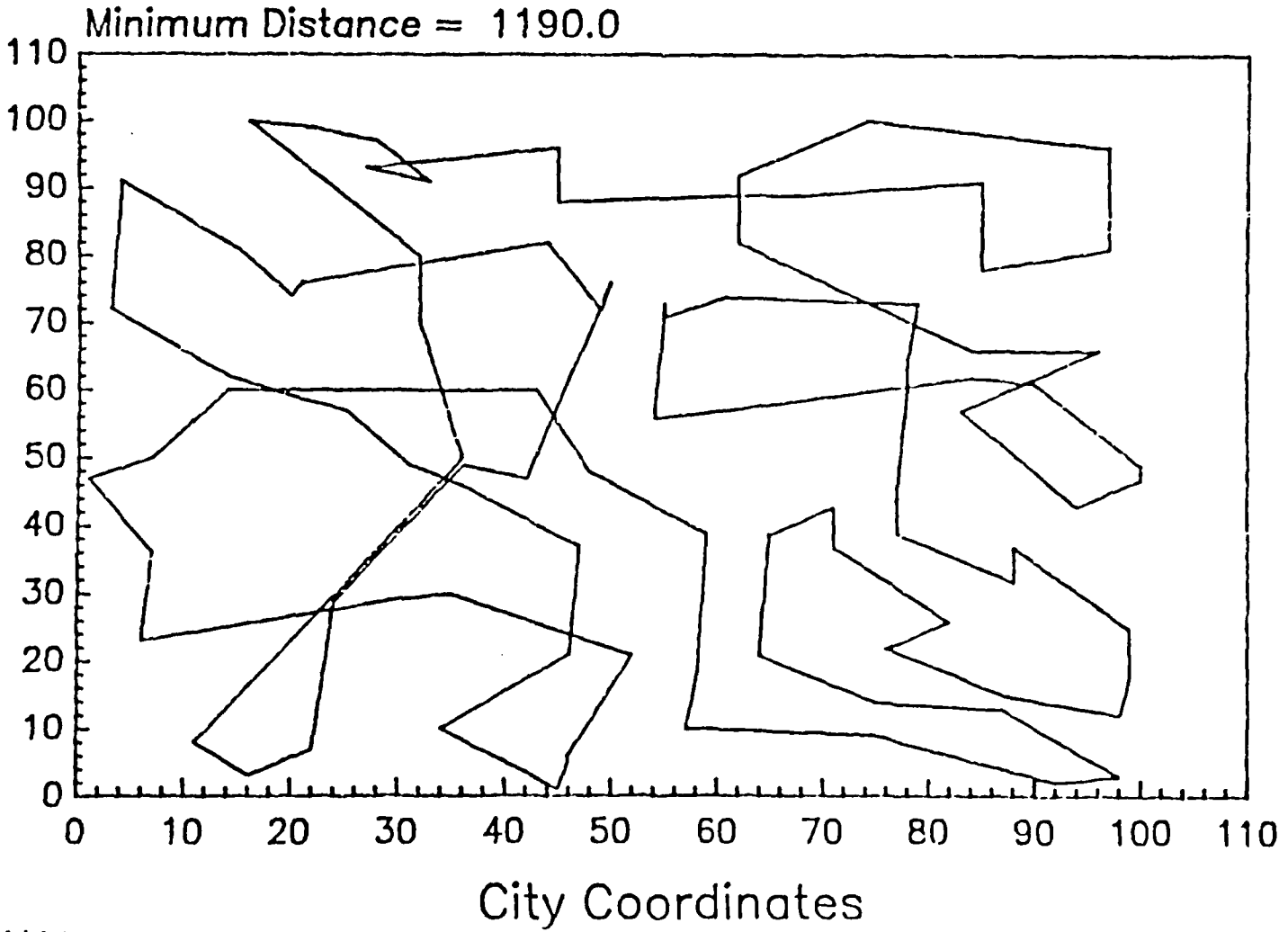


Figure 23

# Traveling Salesman Problem – 100 cities

Experiment #2

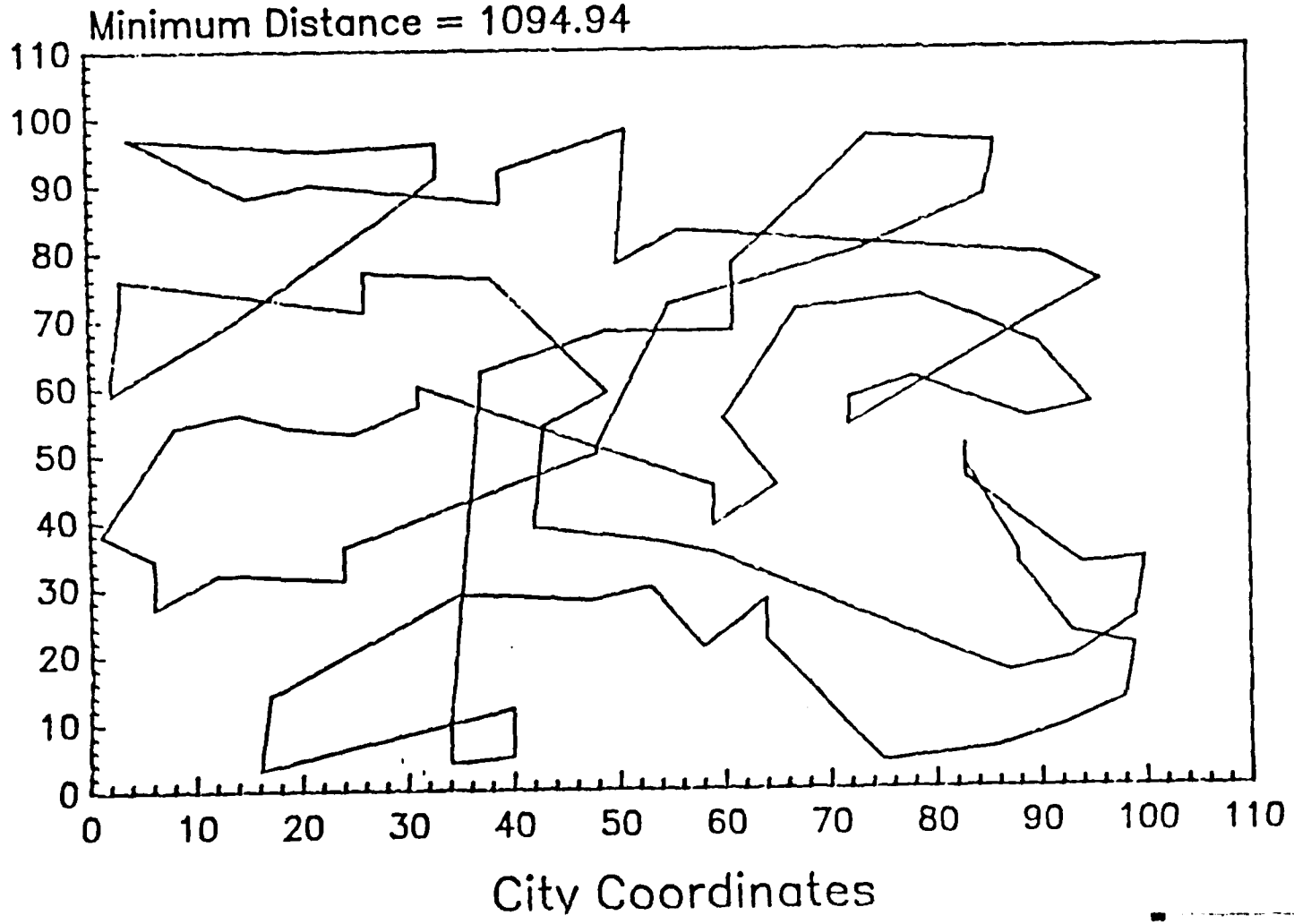


Figure 24

# Traveling Salesman Problem – 100 cities

## Experiment #3

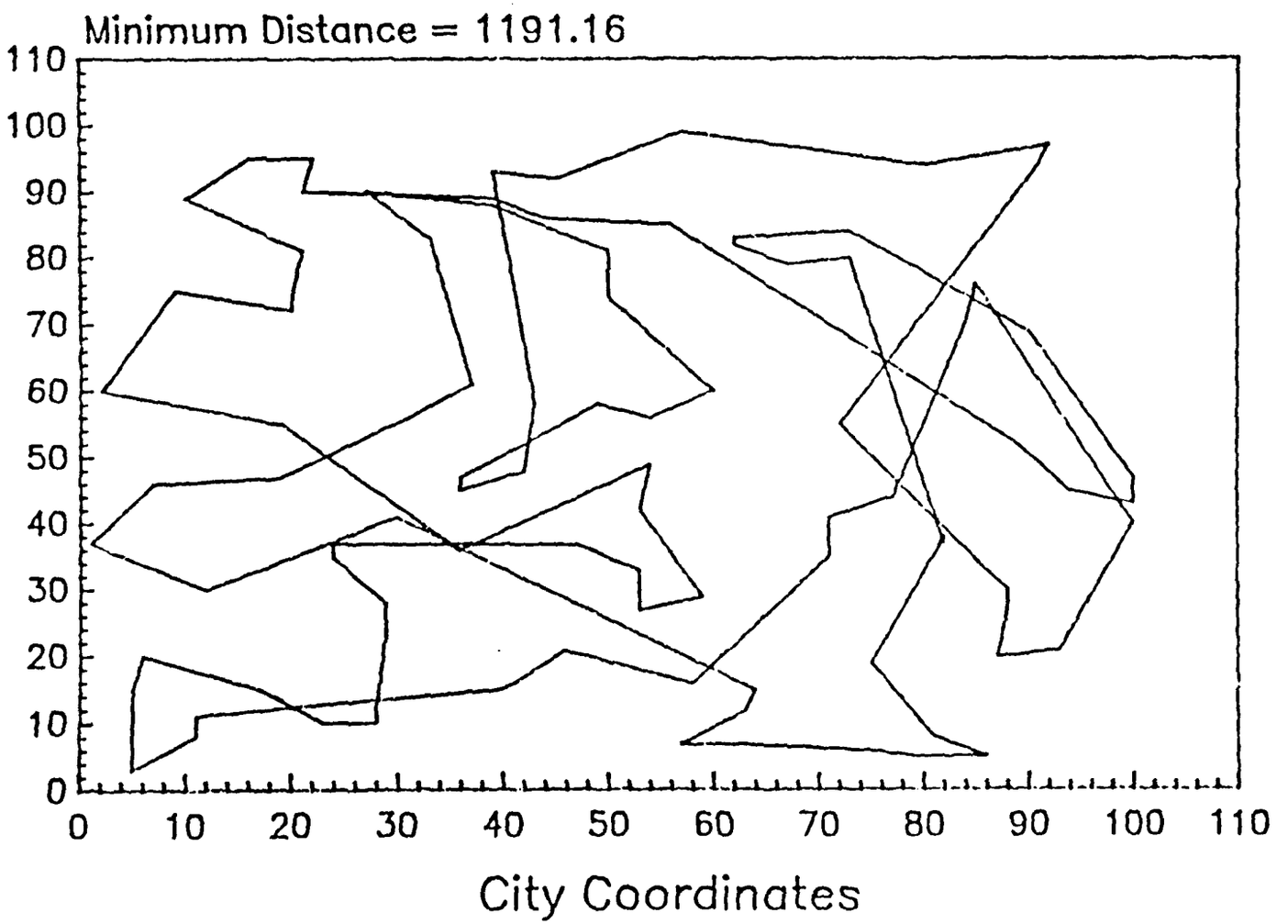


Figure 25

# Traveling Salesman Problem – 100 cities

## Experiment #4

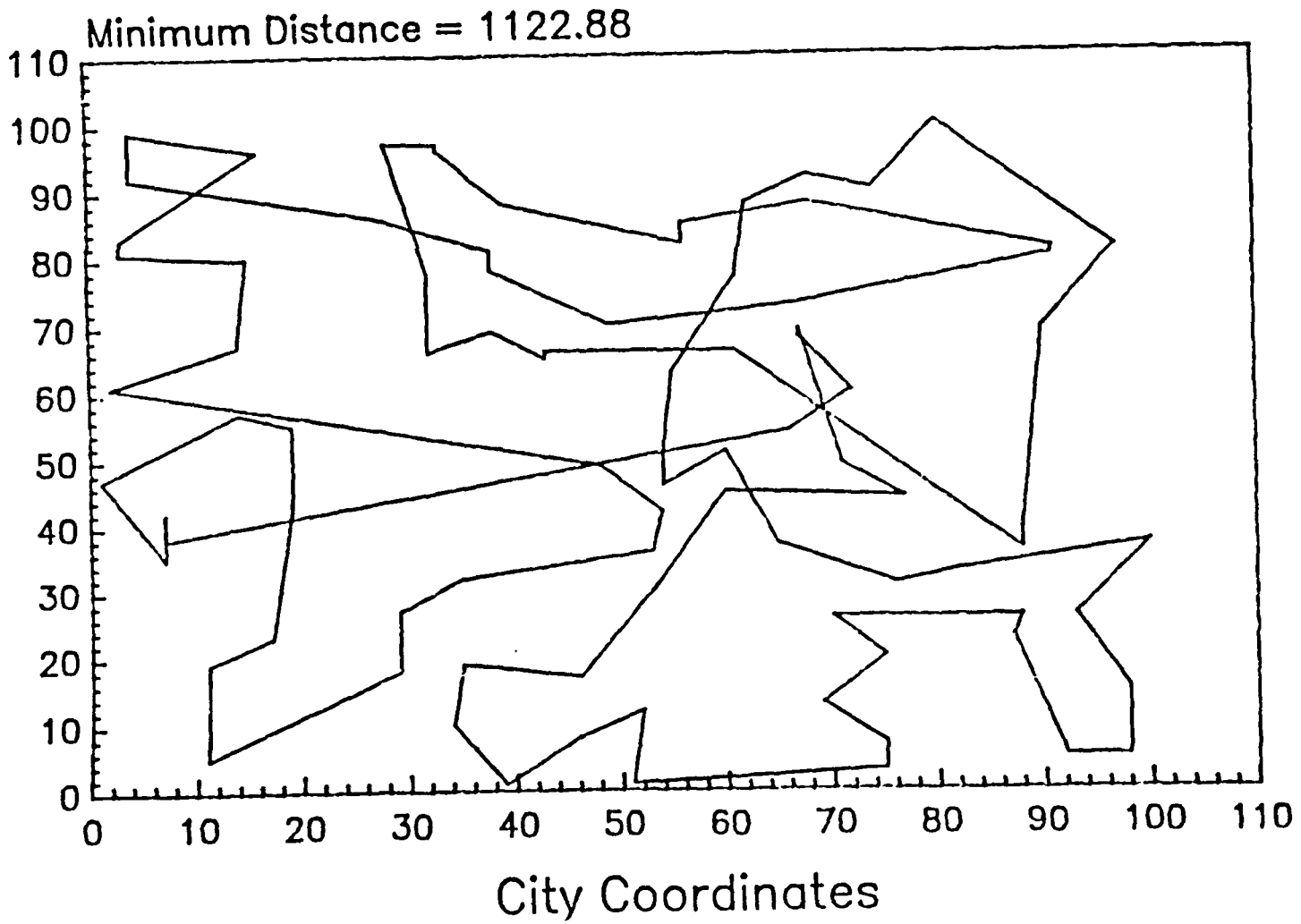


Figure 26

# Traveling Salesman Problem – 100 cities

Experiment #5

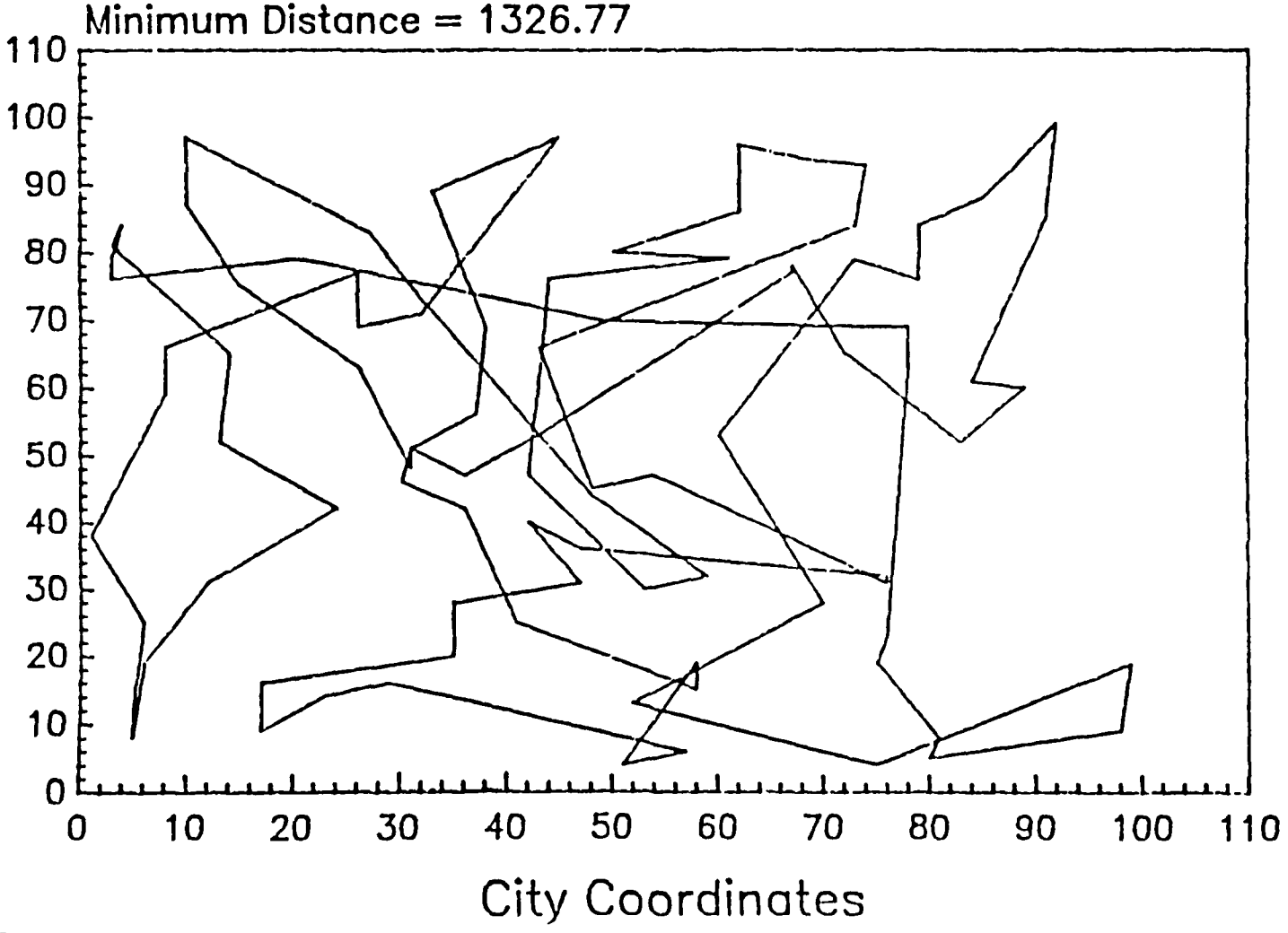


Figure 27

# Traveling Salesman Problem – 100 cities

Experiment #6

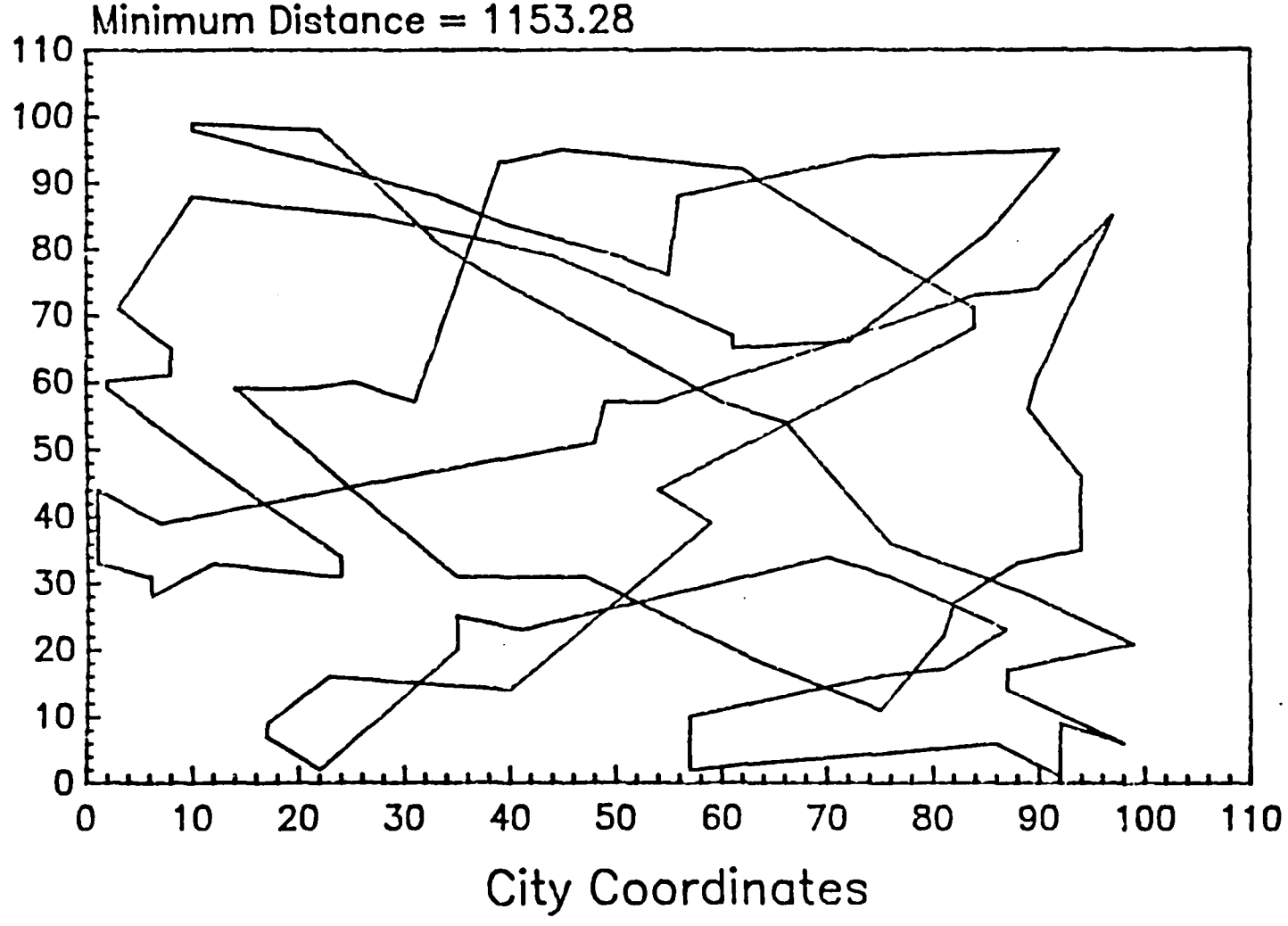


Figure 28

# Traveling Salesman Problem – 100 cities

## Experiment #7

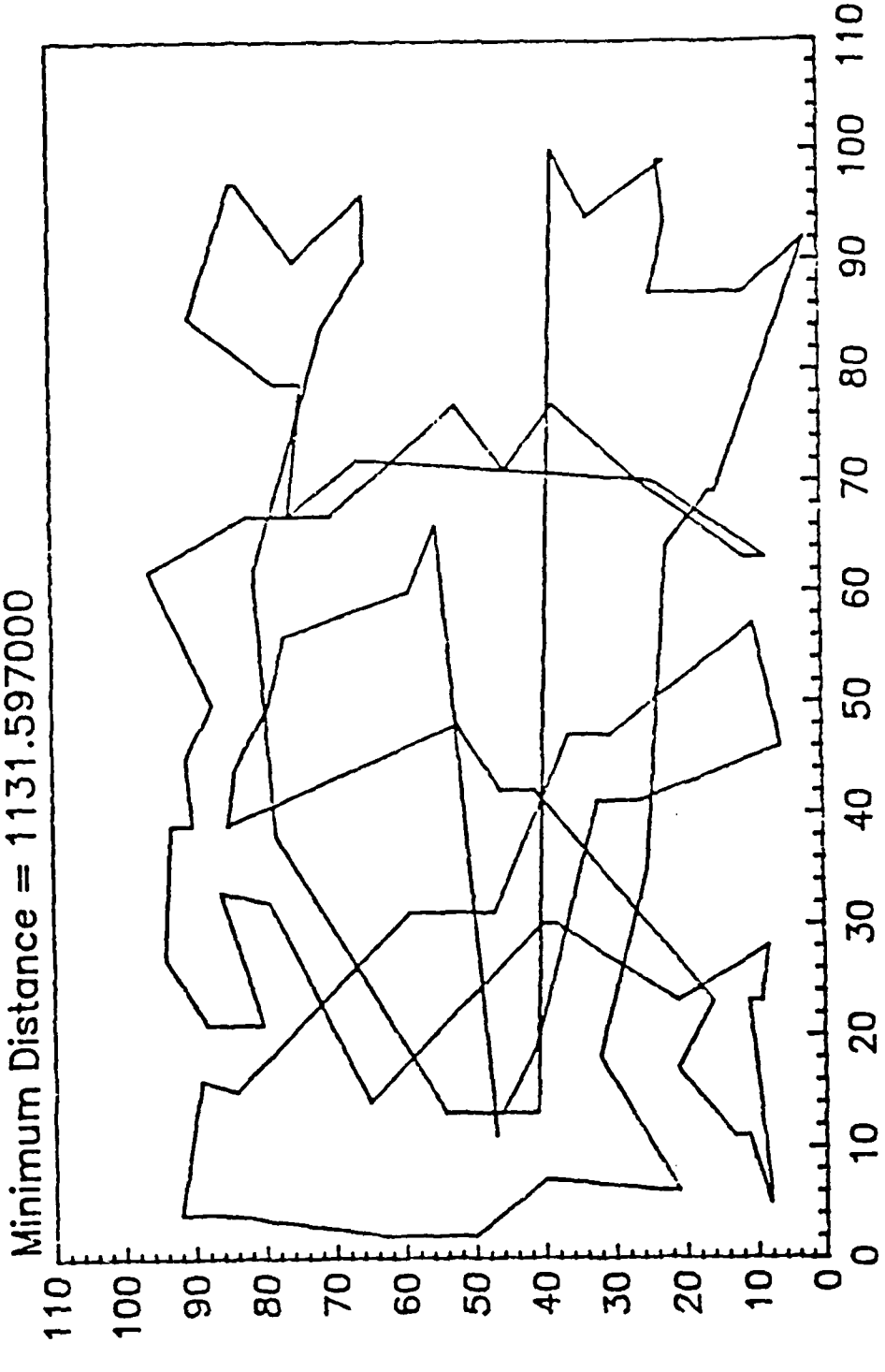


Figure 29

# Traveling Salesman Problem – 100 cities

Experiment #8

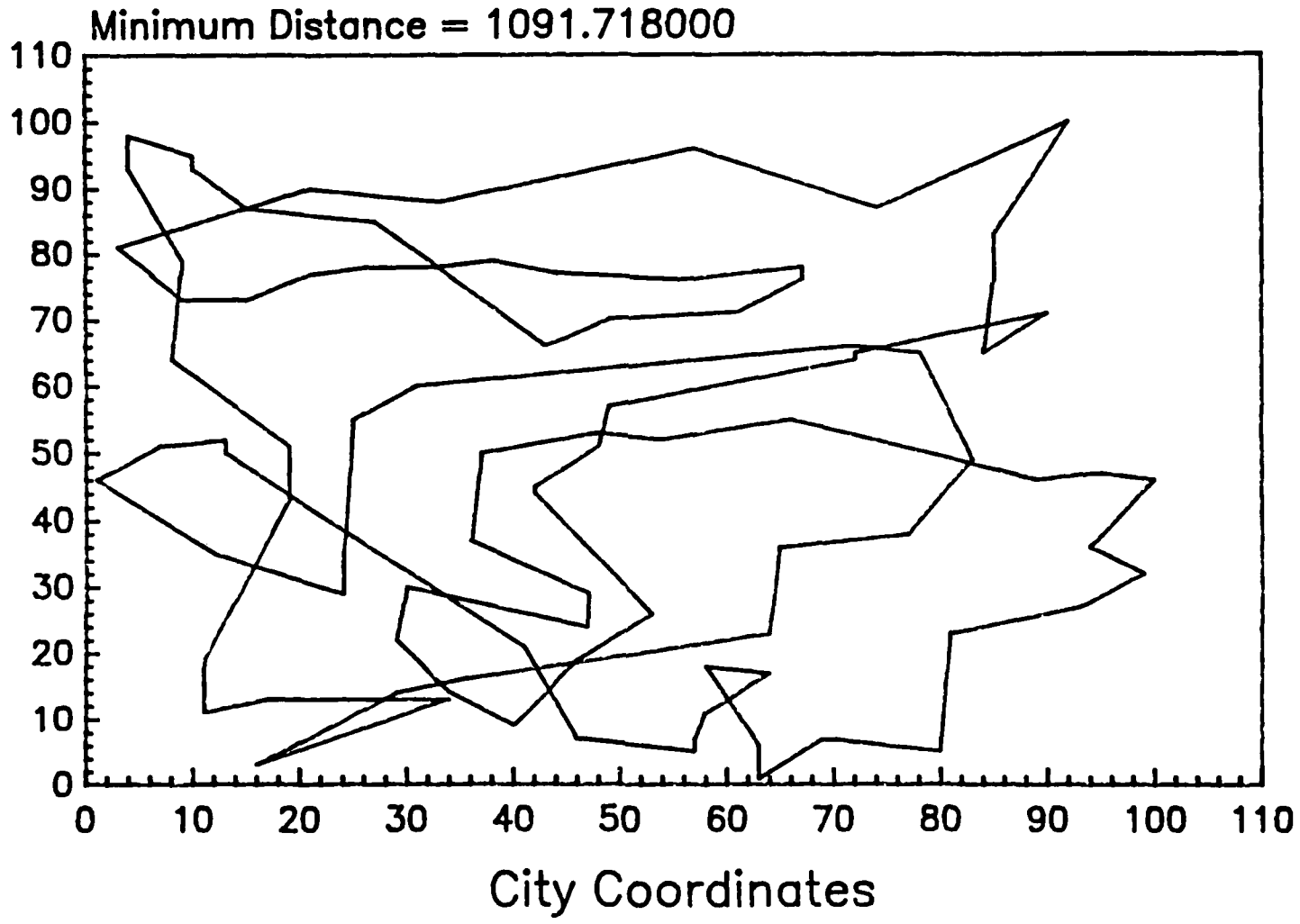


Figure 30

# Traveling Salesman Problem – 100 cities

## Evolutionary Improvement

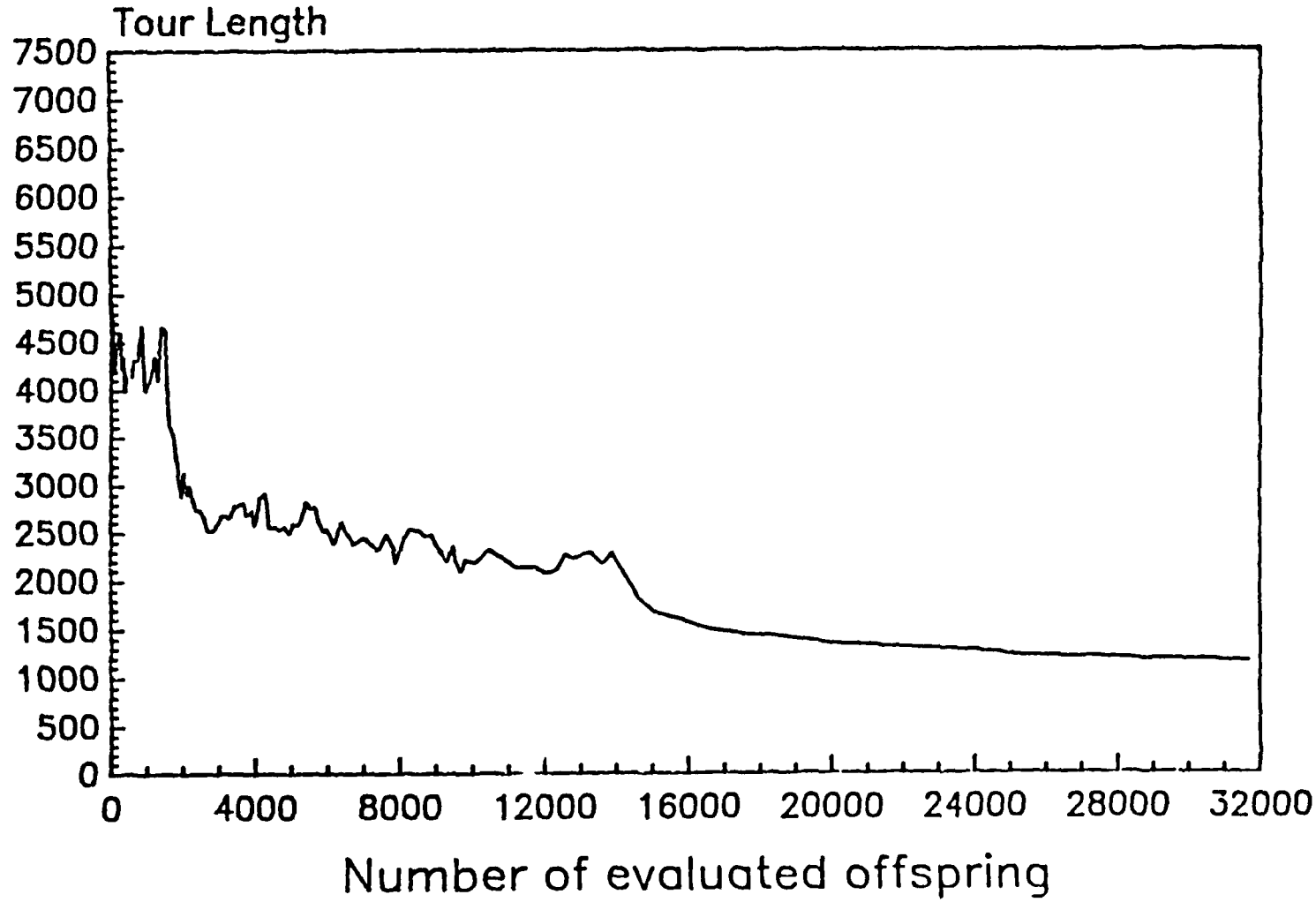


Figure 31

Another experiment required a tour of 90 cities. Here, ten groups of nine cities were randomly placed on the coordinate grid. The process was allowed to evolve 32,000 offspring. While the optimum solution remained undiscovered, it is of interest to note that the problem was evidently addressed at two distinct levels. The evolutionary process initially solved the problem at a gross level, discovering the minimum tour between the groups of cities, see Figure 32. Insufficient time was allowed to sort out the problem at a finer level of detail.

Finally, an extremely large traveling salesman problem was analyzed. Here, 256 cities were randomly distributed. The previous results indicated that the Adaptive Algorithm would not discover the optimum solution; however, in only 10,000 iterations it reduced the initial tour length by roughly 50%. Figure 33 indicates the surviving tour after evaluating 10,000 offspring while Figure 34 indicates the success of the evolutionary process in discovering better and better tours. The available computation time limited the analysis, however the results were certainly encouraging.

## CONCLUSION

Clearly, the Adaptive Algorithm is an effective method for addressing the traveling salesman problem. Several important conclusions can be drawn from the previous experiments:

- "Sophisticated" mutation operations are not only unnecessary, but are detrimental. The experiments point up the necessity for

# Traveling Salesman Problem – 90 grouped cities

Solved At A Gross Level

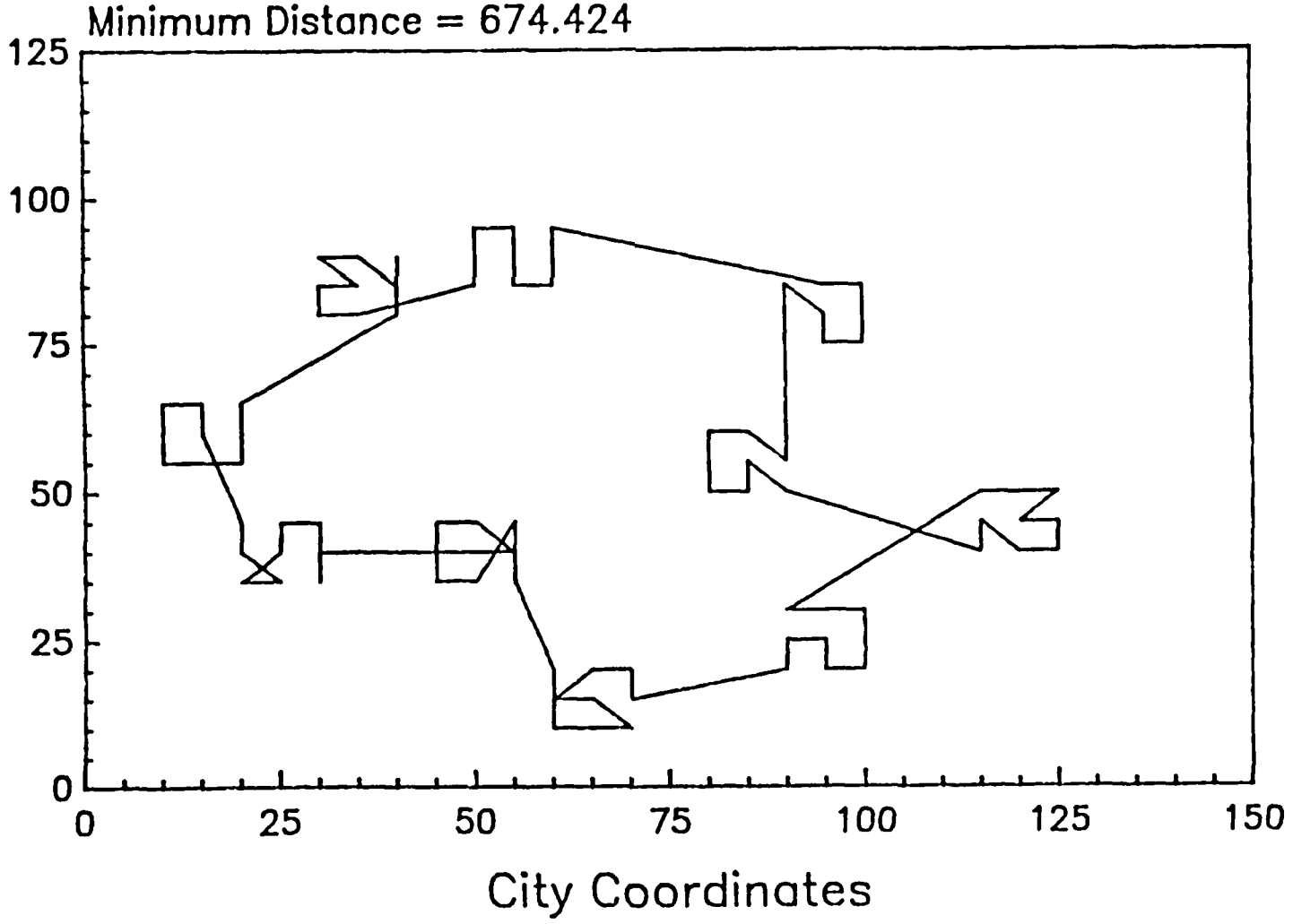


Figure 32

# Traveling Salesman Problem – 256 cities

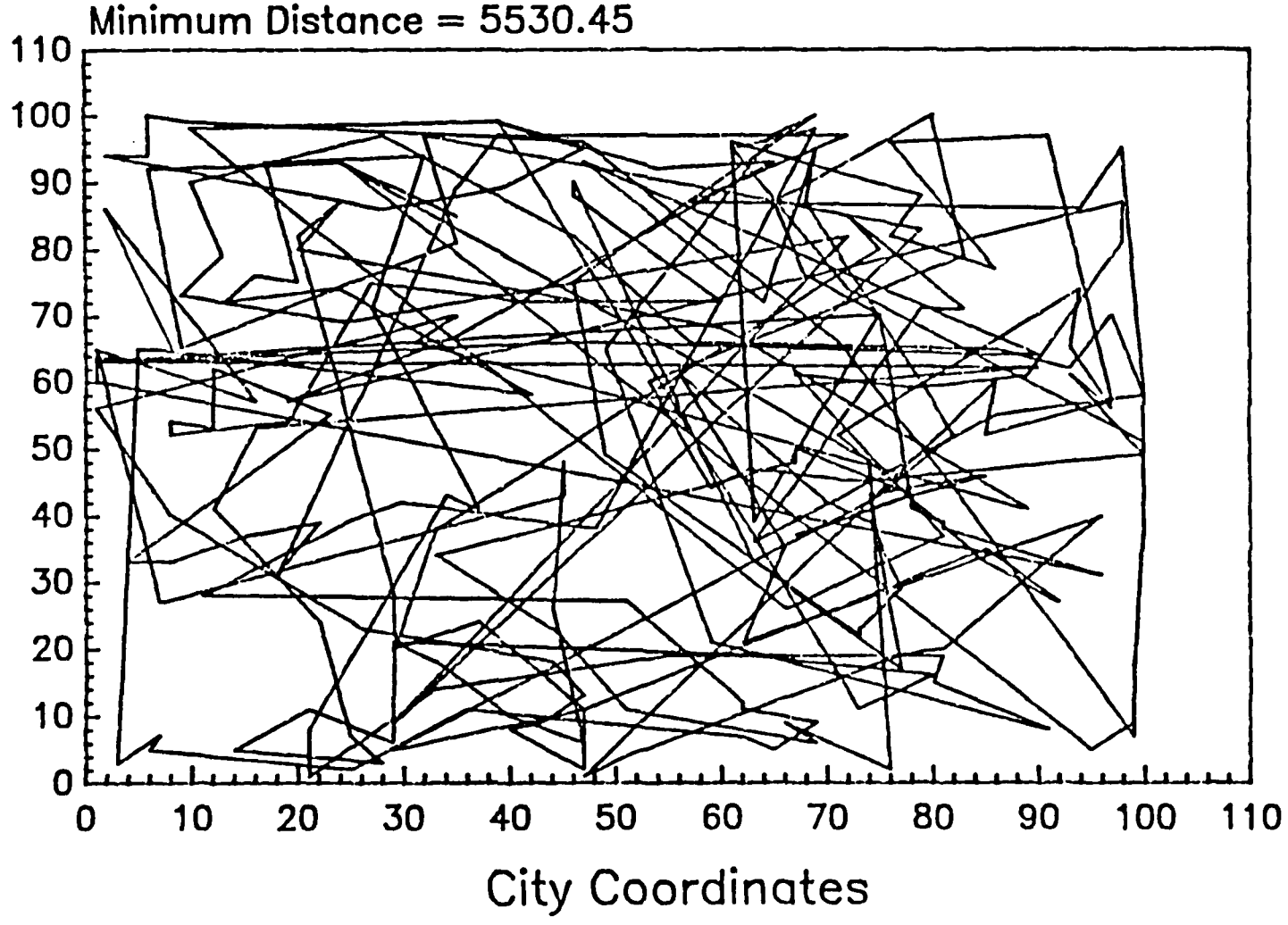


Figure 33

# Traveling Salesman Problem – 256 cities

## Evolutionary Improvement

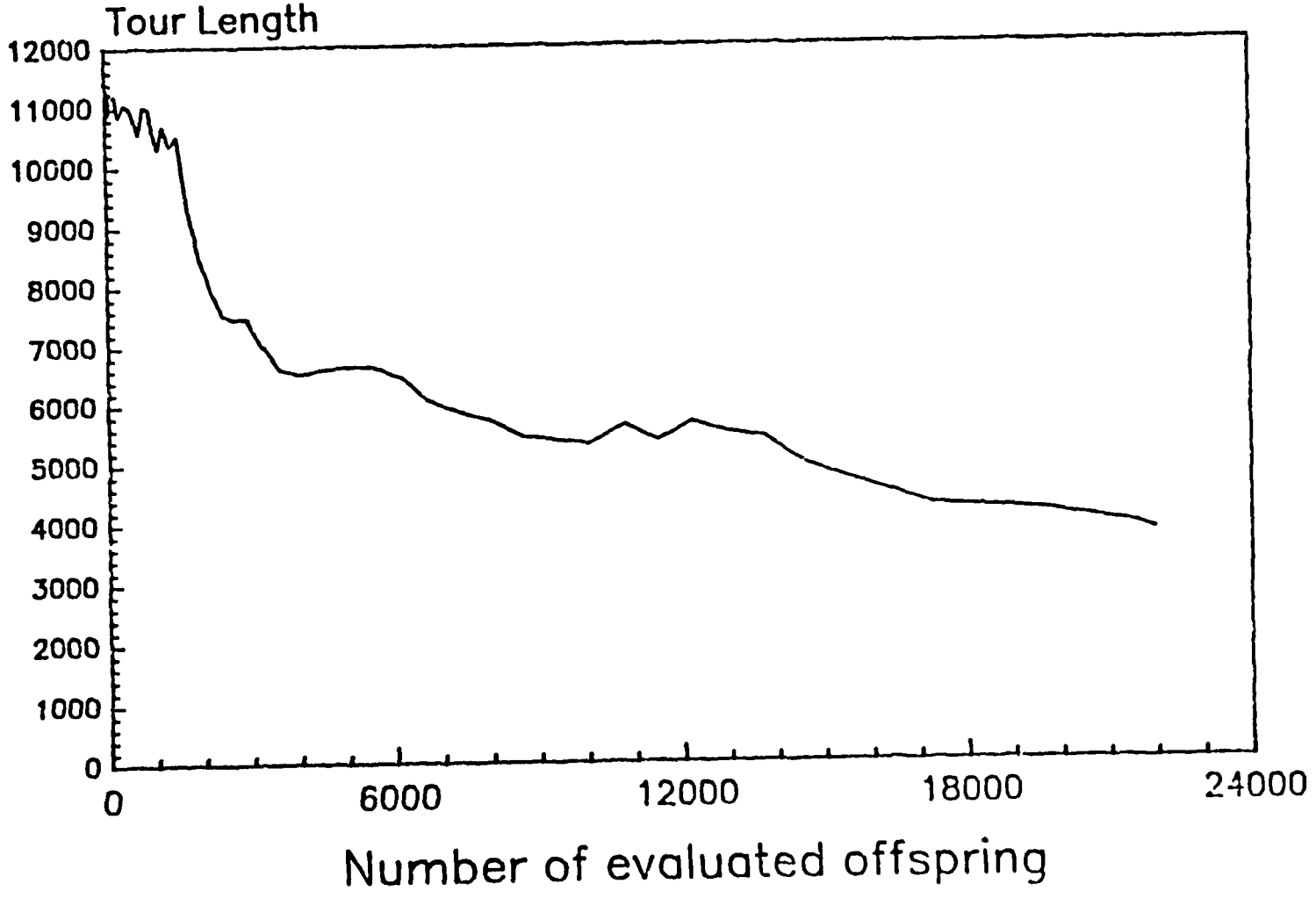


Figure 34

maintaining a substantial link between parent and offspring. The more sophisticated and complex mutation operations destroy this link. The PMX operation may perform generally superior to Holland's crossover operation because it tends to retain more information during each generation.

- There is a beneficial effect of using a noisy payoff function. The concept of a noisy payoff function is similar to that suggested by S. Kirkpatrick, C.D. Gelatt and M.P. Vecchi,[12] for optimizing simulated annealing, but it is not necessary to resort to such specific analogies. In a constantly changing environment, the rewards and penalties for different behaviors vary. The search for better and better solutions is never ending. Evolution is a continuous process with no truly optimum solution.

- Further, it appears unlikely that any specific noisy payoff function exists that will allow discovery of the optimum solution in every traveling salesman problem. Each such problem offers a different adaptive topography; therefore the appropriate distribution and amount of noise cannot be determined *a priori*.

Although no single Adaptive Algorithm can optimally solve <u>every</u> traveling salesman problem, a unique Adaptive Algorithm can be developed to address <u>each</u> traveling salesman problem in a very efficient manner.

---

12 - "Optimization by Simulated Annealing," S. Kirkpatrick, C.D. Gelatt Jr., M.P. Vecchi, *Science*, May 1983.

## References:


1. D. Ackley, "A Connectionist Algorithm for Genetic Search," in *Proceedings of an International Conference on Genetic Algorithms and Their Applications*, J.J. Grefenstette, Editor, Carnegie-Mellon University, July 1985.

2. A.K. Dewdney, "Computer Recreations: Exploring the field of genetic algorithms in a primordial computer sea full of flibs," *Scientific American*, November 1985.

3. L.J. Fogel, "Autonomous Automata," *Industrial Research*, February 1962.

4. L.J. Fogel, "On the Organization of Intellect," Ph.D. dissertation, U.C.L.A. 1964.

5. L.J. Fogel, A.J. Owens, M.J. Walsh, *Artificial Intelligence Through Simulated Evolution*, John Wiley & Sons, New York, 1966.

6. D.E. Goldberg, R. Lingle, Jr., "Alleles, Loci, and the Traveling Salesman Problem," in *Proceedings of an International Conference on Genetic Algorithms and Their Applications*, J.J. Grefenstette, Editor, Carnegie-Mellon University, July 1985.

7. J.J. Grefenstette, R. Gopal, B. Rosmaita, D. Van Gucht, "Genetic Algorithms for the Traveling Salesman Problem," in *Proceedings of an International Conference on Genetic Algorithms and Their Applications*, J.J. Grefenstette, Editor, Carnegie-Mellon University, July 1985.

8. J.H. Holland, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, 1975.

9. S. Kirkpatrick, C.D. Gelatt, Jr., M.P. Vecchi, "Optimization by Simulated Annealing," *Science*, May 1983.

10. C.K. Levy, *Elements of Biology*, third edition, Addison-Wesley Publishing Company Inc., Reading, Massachusetts, 1982.

## Addendum

A completely random search (with replacement) will take roughly twice as long to find the optimum solution as an enumerative search (without replacement). To show this, consider the following two theorems:

**Theorem 1: If there are ß possible solutions and only one optimum solution, the expected number of trials that must be made before the optimum solution is found, using an enumerative search, assuming one trial is made at a time, is equal to (ß+1)/2.**

*Proof:* In an enumerative search, sampling is made without replacement. The probability, therefore, of discovering the optimum solution on any given trial is equal to the product of the probabilities of not discovering the optimum solution on any prior trial multiplied by the reciprical of the number of untried solutions. The expected number of trials that would have to be examined before finding the optimum solution would therefore be:

$$\begin{aligned}\Sigma\, x\cdot f(x) &= 1\cdot ß^{-1} + 2\cdot[(ß-1)/ß]\cdot(ß-1)^{-1} + 3\cdot[(ß-1)/ß]\cdot[(ß-2)/(ß-1)]\cdot(ß-2)^{-1} \\ &\quad + \cdots + (ß-1)\cdot[(ß-1)/ß]\cdots[2/3]\cdot[1/2] + ß\cdot[(ß-1)/ß]\cdots[2/3]\cdot[1/2]\cdot 1 \\ &= 1\cdot ß^{-1} + 2\cdot ß^{-1} + 3\cdot ß^{-1} + \cdots + (ß-1)\cdot ß^{-1} + ß\cdot ß^{-1} \\ &= ß^{-1}\cdot(1 + 2 + 3 + \cdots + (ß-1) + ß) \\ &= ß^{-1}\cdot[ß(ß+1)/2] \\ &= (ß+1)/2. \qquad \text{Q.E.D.}\end{aligned}$$

**Theorem 2: If there are B possible solutions and only one optimum solution, the expected number of trials that must be made before the optimum solution is found, in a completely random search, assuming one trial is made at a time, is equal to B.**

*Proof:* In a completely random search, sampling is made with replacement. The probability, therefore, of discovering the optimum solution on any given trial is equal to the product of the probabilities of not discovering the optimum solution on any previous trial multiplied by the reciprical of the total number of possible solutions. The expected number of trials that would have to be examined before finding the optimal solution would therefore be:

$$\begin{aligned}\Sigma\, x \cdot f(x) &= 1 \cdot B^{-1} + 2 \cdot [(B-1)/B] \cdot B^{-1} + 3 \cdot [(B-1)/B]^2 \cdot B^{-1} + \cdots \\ &= B^{-1} \cdot (1 + 2 \cdot [(B-1)/B] + 3 \cdot [(B-1)/B]^2 + \cdots) \\ &= B^{-1} \{1/(1-(B-1)/B))\}^2 \\ &= B. \end{aligned}$$

Q.E.D.