UNIVERSITY OF Southampton

University of Southampton Research Repository

Copyright © and Moral Rights for this thesis and, where applicable, any accompanying data are retained by the author and/or other copyright owners. A copy can be downloaded for personal non-commercial research or study, without prior permission or charge. This thesis and the accompanying data cannot be reproduced or quoted extensively from without first obtaining permission in writing from the copyright holder/s. The content of the thesis and accompanying research data (where applicable) must not be changed in any way or sold commercially in any format or medium without the formal permission of the copyright holder/s.

When referring to this thesis and any accompanying data, full bibliographic details must be given, e.g.

Thesis: Author (Year of Submission) "Full thesis title", University of Southampton, name of the University Faculty or School or Department, PhD Thesis, pagination.

Data: Author (Year) Title. URI [dataset]

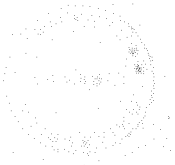
A NEW CONCEPT IN THE DESIGN OF FINITE AUTOMATA

by

F. W. M. STENTIFORD, M.A.

A thesis submitted for the degree of Doctor of Philosophy in the Faculty of Engineering and Applied Science, University of Southampton.

1974



FACULTY OF ENGINEERING AND APPLIED SCIENCE
ELECTRONICS

Doctor of Philosophy

A NEW CONCEPT IN THE DESIGN OF FINITE AUTOMATA

by Frederick Warwick Michael Stentiford

In recent years considerable effort has been brought to bear upon problems arising in the field of Computer Aided Design. Total mechanization of the design process has generally proved difficult and many successful projects have relied upon a certain measure of human interaction. This thesis proposes and discusses an approach to the automatic design problem in which computer searches are guided by an evolutionary process of random change followed by empirical evaluation.

In support of the theory evolutionary methods have been applied to two problems. Firstly a practical method for the reduction of very large finite-state machines is proposed and evaluated. Secondly an evolutionary search is used in the automatic design of an economical set of features for the recognition of OCR B printed characters. Detailed results are presented which give a clear indication of the novel performance of the system. Although the performance falls short of that associated with many commercial equipments, the results are shown to compare favourably with those achieved by an independently and intuitively designed set of features.

It is suggested that the work can be extended to Optical Character Recognition problems involving many fonts, and also to problems of fingerprint classification.

CONTENTS

APPENDICES

LIST OF FIGURES

ACKNOWLEDGEMENTS

The author would like to acknowledge the guidance and support given to him by his supervisor, Professor D. W. Lewin. He would also like to thank his colleagues both at Southampton University and within the Plessey Company Limited for many stimulating discussions.

The author is also indebted to the Plessey Company Limited for providing the OCR data which was used during this study.

The work was carried out with the financial support of the Plessey Company Limited and the Science Research Council.

Finally I must thank my wife, Pauline, for the encouragement and Janice Mackley who typed the final report.

1. INTRODUCTION

1.1 General

In recent years considerable effort has been brought to bear upon problems arising in the field of Computer Aided Design. Research has ranged from automated printed circuit design to the automatic selection of features for pattern recognition. These tasks can quite often be solved by human intuition, but serious difficulties are generally encountered as soon as attempts are made to mechanize the process. A common obstacle in many design problems is the absence of any formal theory which might lead directly to a practical solution. More often than not the automated approach involves an extensive search procedure which is kept within the bounds of practicability by the application of heuristics. An exhaustive search would theoretically produce all possible solutions, but in anything other than trivial problems such a method is not feasible. Of course, suitably chosen heuristics often lead to satisfactory solutions to particular problems. There is no guarantee, however, that heuristics chosen only by intuition and not supported by formal argument or empirical evidence, will always produce the desired results. Indeed the best solutions can be precluded entirely.

The research reported in this thesis has attempted to avoid the use of intuitive heuristics at the outset of the investigation of two design problems. Rather than introduce heuristics when first faced with an extensive search, this approach relies upon an evolutionary process consisting of a series of random steps each followed by an empirical evaluation. Only after a set of useful results have been produced are search heuristics extracted and used to enhance the original search process.

1.2 The Original Contribution

The intention in this thesis has been to demonstrate the effectiveness and exemplify the philosophy of evolutionary methods. Although the evolutionary concept is by no means original (1), to the author's knowledge, it has not been employed before in the manner of this thesis. The method is proposed as a search tool to be used in those instances in which mathematically justified exclusion rules do not reduce the volume of the search space to a practical size.

In support of this theory evolutionary methods have been applied for the first time in two problem areas. Firstly a new method for the reduction of finite-state machines is proposed (2,3) and evaluated. Secondly an evolutionary search is used in order to design a novel set of features for Optical Character Recognition. In addition an evolutionary approach to the concept of randomness has been put forward(4).

The basic philosophy behind this work rests on the avoidance of intuitive heuristics which are not supported by adequate empirical evidence to show that they are suited to the problem concerned.

1.3 Outline of the Thesis

The thesis begins with a discussion of some of the difficulties which frequently crop up in automated design problems. Particular emphasis is laid upon the difference between intuitive heuristics and rigorously supported rules in computer searches. The use of heuristics is outlined in the context of the reduction of finite-state machines and pattern recognition. The evolutionary method is introduced as a search tool enabling both results and useful search heuristics to be extracted. The next two chapters describe particular applications of the evolutionary approach. Firstly, chapter three describes algorithms for the reduction and decomposition of finite-state machines. These algorithms are

evolutionary in the sense that they rely on a source of random inputs and a process of empirical evaluation. Chapter four reports research in which an evolutionary search procedure has been used in the automatic design of recognition logics for an Optical Character Recognition system. The results are described and compared with those obtained by an intuitive design procedure. Finally chapter five draws conclusions from the various aspects of the work and proposes several areas for further study.

2. CURRENT LIMITATIONS IN THE DESIGN OF AUTOMATA

2.1 Computational Limitations

A modern digital computer functions in a logical manner by performing simple operations interpreted sequentially from a specified programme of instructions. The computer never deviates from behaving in this way, each step being perfectly predictable from the last according to the fixed instruction repertoire of the machine. It is natural therefore, that problems suitable for analysis by computer should be formalized and converted into a representation that is both unambiguous and amenable to computer implementation.

In the case of particular problems it might appear that a formal description in which all possible aspects of the task are detailed is ideally suited to the rapid processing capability of a digital computer. Such problems sometimes demand that computers search exhaustively through some large space of solution attempts. Certain small problems can be approached in this straightforward manner, but the majority of real practical problems meet with considerable difficulty when exhaustive techniques are employed. For instance, in the game of chess it is estimated that there are 10^{120} different continuations. Even the fastest and biggest computers available today would only be able to explore the minutest fraction of possible chess games. In a discussion on the possible reduction of search effort Minsky (5) has said, "the real problem is to find methods which significantly delay the apparently inevitable exponential growth of search trees".

Often similar computational difficulties arise from the chosen logical structure in which the problem is embedded. Michie (6) in an article on Artificial Intelligence comments, "the draw-back of the approach lies in the cumbersomeness of present day procedures for operating on logic statements, and the speed with which they tend to ramify into a forest of irrelevant deductions, particularly if the initial set of axioms is of more than trivial size". He reiterates this statement again in a theorem proving context (7) by saying, "mechanized proof procedures very easily stray into unprofitable inference paths through lack of any adequate formulations of the notion of relevance".

It appears therefore, that the computer scientist is faced with two alternatives, either he must obtain much more powerful computing machines or else relax the formal requirements that the desired results must satisfy. In the next section it will be shown that the first of these alternatives is not feasible.

2.2 Hardware Limitations

The computing power of modern machines is limited by their physical structure. A study of the power dissipation in logic circuits, based on extrapolation of present technology, indicates that the limit on speed lies only about an order of magnitude beyond the speed of the fastest contemporary circuits (8). The ILLIAC IV computer, currently under contruction, effectively plans to speed up computation by operating 64 processors in parallel. This, however, raises very serious problems on how to program the machine efficiently. "One of the most difficult tasks in organising an ILLIAC IV program is the allocation of storage.... the data should be stored in such a way that the user's algorithm keeps most of the processing elements active most of the time" (9).

Bremermann (10) has conjectured a theoretical limitation on any data processing system whether it be artificial or living, or parallel or sequential. He states that such a system can process no more than ($2x10^{47}$) bits/sec/gm. Even this number is small when compared with the number of possible patterns on a simple black and white 20 x 40 mosaic ($\sim 10^{340}$) or again the total number of chess games ($\sim 10^{120}$). It is not therefore feasible to expect imrpovements in hardware alone to overcome these huge computational problems. Michie (7) comes to the conclusion that the real difficulties lie with the software rather than the hardware. "At present limitations lie in inadequate understanding of mathematical-logical and programming principles rather than in hardware speeds or storage capacities".

In view of the impracticalities involved, emphasis in this research has shifted away from exhaustive methods and has sought to find techniques through which the results of incomplete analysis can be used to make searches more efficient (5, 10, 11, 12, 13, 14, 15).

2.3 Heuristics and Sieves

In an effort to reduce the sheer bulk of computation associated with many applied problems (for instance, switching theory and pattern recognition) it has been necessary to introduce 'heuristics' in order to direct the search away from unpromising possibilities and thereby reduce the processing requirements to practical levels.

Minsky (15) defines heuristics as "rules or principles which have not been shown to be universally correct but which often seem to be of help even if they may also often fail". It is common for any search aid to be called a heuristic. For instance, in the search for prime numbers it can be shown that all primes are of the form ($6n\pm1$). Some authors would call this fact (Sieve of Eratosthenes) a heuristic, but this thesis takes the view that such a rule only serves to define the search space. This attitude is taken because many of the problems

considered in this thesis are not subject to a complete analytical solution, and hence any heuristics that are employed would necessarily be of an intuitive nature. In the context of searches the distinction is made therefore, between a rigorously proved mathematical property or 'sieve', and the rather more frequently encountered heuristic. In view of these considerations the following definition of a heuristic will be employed:

> a heuristic is any algorithm or part of an algorithm not completely supported by rigorous analytical theory, which it is intuitively felt will aid the search for the solution to a particular problem.

In writing search programmes a priori intuitive knowledge of the problem that is known in advance is generally utilized by way of heuristics. Indeed in the more intricate searches heuristics are used to the extent that "no trial is made without a compelling reason, just as expensive experiments must be carefully designed in any research" (12). In this way, searches are carried out by the application of a set of rules some of which are intuitive and serve only to limit the processing.

The branch-and-bound algorithm has been used many times to embody heuristic information into a programmed search (16). Consider the problem of minimizing a function f in some discrete set X. Suppose that we have available a function g which computes a lower bound for f in a set A:

$$g(A) \leq f(x) \text{ for all } x \in A \text{ and } A \subset X \ldots\ldots\ldots [a]$$

Informally the branch-and-bound strategy partitions the set X into successively smaller portions, each time only further subdividing the most promising subset of X, as estimated by g.

Eventually a single point set is selected for subdivision thereby terminating the algorithm with the single point as the minimum. The actual lower bounding function depends on the particular problem and the a priori knowledge available and in this sense is a 'heuristic function'. In actual problems the possession of such a lower bounding function g is equivalent to the knowledge of an analytical fact about the problem domain. In terms of this thesis g does not therefore represent a heuristic but a universal mathematical property. In general an intuitive heuristic will not be supported by sufficiently strong evidence to satisfy inequality [a] and therefore there will be some uncertainty about the validity of the theorem and the minimality of the search.

Hart et al. (17) extend this technique to heuristic graph search and give as an example the problem of discovering a sequence of cities on the shortest route from a specified start to a specified goal city. They quote a suitable lower bound on the distance between cities as being the airline distance between them, and use this sieve to reduce the number of cities that need to be considered. This additional information can be logically deduced from normal Euclidean geometry and must lead to an optimal search. Sieves are extremely useful if they can be found. However, it is more common for information to be of an intuitive nature and not necessarily valid everywhere in the problem domain.

In short it can be said that the real payoff in using intuitive heuristics is a greatly reduced search and, therefore, practicality. Feigenbaum and Feldman (18) make the vital point, however, that "by drastic search limitations, sometimes the best solution (indeed, any or all solutions) may be overlooked". It is not possible to tell in advance how restrictive a set of heuristics

may turn out to be. Indeed the restrictions could quite easily go unnoticed thereby precluding many potential solutions which lie outside the bounds laid down by the heuristic rules themselves.

2.4 Finite State Machine Analysis

2.4.1 General

Algebraic machine theory is a branch of the theory of computation. In application it can be used to model the overall action of specific pieces of logical hardware and enables problems of design to be stated clearly and unambiguously. It does not, however, model the actions of electronic devices directly, but only through the movement of abstract 'states'. Clocked logical circuits normally operate by passing through a sequence of states, each state corresponding to a particular setting of the internal memory elements. Each 'next state' and output is determined solely on the next input and the current state. The two principal sequential machine models are the Moore machine and the Mealy machine. In both these models the output is a function of both past and present inputs, but whereas in the Moore machine the output is associated with the current state, the Mealy machine only gives outputs on state transitions (Appendix B.1). These models have been used extensively in research on the use of computer aids for logic circuit design. This section discusses the problems involved in reducing the number of internal states and the problem of implementing the machine in hardware by state assignment using preserved partitions.

2.4.2 State Reduction

In general it is felt desirable to remove redundant states from a machine in order to gain corresponding reductions in the final hardware. In particular cases, however, the process of state reduction itself can lead to more complex realizations in terms of hardware (19). Care should be taken to ensure that a simple internal structure does not exist before reduction takes place.

Most algorithmic state reduction techniques involve a search through all the maximal (or prime) compatible sets of states for a set which satisfies cover and closure requirements (20, 21, 22, 23, 24) (Appendix B.4). This approach has the advantage that the theory is sufficiently general to describe all possible reductions. That is, every possible valid reduction of a finite-state machine is describable in terms of compatible sets of states. An exhaustive search through all possible compatible sets would therefore guarantee an optimal result. However, the computational requirements can go up as the power of the number of machine states and it is usual for such techniques to employ sieves and heuristics in order to delay this exponential growth.

2.4.3 State Assignment

Having obtained a suitably reduced state table, the next step in the design procedure is the allocation of a binary code to every internal state so that input equations for the storage elements may be derived. The total number of distinct state assignments for a state table with n states using r state variables is (see (26))

$$\frac{(2^r - 1)!}{(2^r - n)r!}$$

In order to avoid an exhaustive search for an optimum assignment, several rules of thumb have been devised to assist the development of an economical circuit (26). The best realizations are normally obtained by utilizing structural information from each individual machine, rather than by using an external algorithmic procedure, (e.g. binary assignment). Such an assignment can generally be achieved from a knowledge of the preserved partitions on the state set (27), (Appendix B.2).

For example, suppose a preserved partition π on the state set $\{1,2,3,4,5,6,7\}$ is $\pi = \{\overline{1}; \overline{2,3}; \overline{4,6}; \overline{5,7}\}$. Then since 3 bits of storage are essential for 7 internal states, two bits can be used to identify the block and the remaining bit to distinguish the states within a block. This leads to an assignment in which the next state variables have a reduced dependence on the current state variables. This often means that fewer gates and connections will be required thereby leading to a more economical implementation. It has been shown (28) that most large machines (with certain bounds on the number of inputs) possess preserved partitions. This means that frequently there will be substantial hardware savings associated with the partitioning approach to state assignment.

Special machine structures can be recognized through their sets of preserved partitions and partition pairs. For instance, if it is possible to implement a machine as a shift register, then a set of partition pairs will exist which possess the comapping property (Appendix B.2). Shift register implementations cannot be forced onto an arbitrary machine; suitable structure must be present in the original machine if shift register decomposition is to be useful. However, it is quite possible that such an implementation could be achieved by first introducing appropriate redundant material into the machine.

2.4.4 Semigroup Approach

The internal structure of finite automata may also be described using semigroups (Appendix B.5). The machine may be represented by a set of mappings of the state set into itself with each mapping corresponding to an input. This set of mappings forms a semigroup and all the results of classical semigroup theory can be applied to such a set.

However, this formal theory makes two serious restrictions on the sort of machine which can be analysed. Firstly a sequence of inputs is considered to be equivalent to a single input without regard for the corresponding output strings. It does not seem sufficient to arrive at the right state when the output strings are different. It is not therefore anticipated that much application will be made to

real design problems (29). Secondly no account is taken of don't care conditions. In order to set up the semigroup the blank entries must be filled arbitrarily, Completing the machine in some useful way is still an unsolved problem (30).

Finally if this analysis is to be applied, the semigroup will probably need to be given by its multiplication table. This table will be substantially larger than the state transition table. For instance a machine with 10 states and 2 inputs will have a state transition table with 20 entries; on the other hand, the semigroup may have up to 10^{10} elements, and so the multiplication table may have up to 10^{20} entries. It is this size problem that virtually prohibits a direct use of the semigroup; it can only be used through its general properties at the conceptual level (30).

It can be seen that much of the work associated with the design of automata seems to be hampered by huge computational problems as soon as problem magnitudes increase (31). Whenever the work has been applied sufficient heuristics have generally been introduced to enable the practical results to be achieved.

2.5 Pattern Recognition

2.5.1 General

Although pattern recognition has been studied for a considerable length of time, it has not developed into a coherent discipline. Much of the work in pattern recognition has been concerned with

particular problems in particular circumstances and has contributed very little towards a general theory. Very often certain recognition tasks are successfully accomplished using so-called pattern recognition techniques, but nearly always the method of solution is very closely tailored to the particular problem with no hint of a general approach.

The only problem which falls under the heading of pattern recognition that has met with sufficient success to enable products to be marketed is that of Optical Character Recognition (OCR). The recognition of OCR B font characters is the central pattern recognition problem considered in this thesis, and will serve here as an illustration of some of the difficulties encountered in a real recognition task. To take a simplified example, it may be required to recognise the character 'A' when several different binary patterns are displayed on a 20x40 matrix. The 'foolproof' approach would be to record and label each of the 2^{800} different possible patterns with 'A' or 'not-A';then each time a decision is required, the test pattern could be compared with the library of measurements for an accurate recognition. Of course, this is not a practical solution and the usual problem of how to reduce the amount of data processing occurs again.

The principal reason for the independence of many workers in pattern recognition has been the necessity for carefully designed heuristics to make shortcuts through the enormous sequence of possible trials. These heuristics generally relate only to the a priori knowledge of the current task and are of little interest in other problem areas. The intuitive heuristic method is almost universal in pattern recognition: Nagy (32) states, "Heuristic methods for the discovery of measurementsfor pattern recognition

tasks are largely responsible for almost all of the pattern recognition devices which have been incorporated in practical systems to date". Kovalevsky (33) goes further by saying, "At the present time no one doubts any longer that a priori restrictions substantially narrowing the class of solutions must be present in every recognition problem".

For instance, in OCR instead of considering all possible pattern measurements, heuristics are used to select a small number of tests or 'features'. These features are chosen in such a way that the character to be recognised is characterized and can be identified by the presence of absence of the features, (e.g. 'two loops' is a feature of the character '8'). Once the features have been chosen many authors apply sophisticated statistical methods in order to minimize errors. However, "unless the features separate the patterns no amount of statistical analysis can untangle the errors that necessarily result when pattern images overlap" (34).

The enormity of the size of some of the search spaces sometimes itself has a effect on the course of pattern recognition research. This is best illustrated by the OCR example. It might have been decided to use binary features of the matrix matching type (see section 4) which gives binary responses depending upon whether or not they fit anywhere on the test character. If each element of the matrix can be black-seeking, white-seeking or don't care, the total number of possibilities for a 10x10 matrix and for a 16x30 matrix are $\sim 10^{50}$ and $\sim 10^{240}$, respectively. This argument is sometimes used against expanding the search from the 10x10 matrix

to the more complex, and therefore more numerous, 16 x 30 matrix measurements (32). There is no empirical evidence for making such a decision, for a fruitless search in a space of 10^{240} possibilities is hardly likely to be successful in one containing 10^{50}. Moreover a solution to the OCR problem through a small 10 x 10 window is intuitively a much more difficult task than if the whole character is visible through a larger window.

To summarize, it can be said that pattern recognition problems almost always involve a search through an inordinately large space of possibilities and restrictive heuristics are invoked in order to reach some sort of solution. Indeed there appears to be no rigorous theory which can be applied to reduce the enormous searches which occur in real world pattern recognition problems.

2.5.2 The Mesa Phenomenon

It is common for many search procedures to be formulated in such a way that the goal is achieved when the value of an evaluation function $E(\lambda_1, \lambda_2, \ldots, \lambda_n)$ is maximized by adjusting the parameters $\lambda_1, \lambda_2, \ldots, \lambda_n$. For instance, in pattern recognition the λ_i might represent the elements of a feature and E the evaluation function for that feature. The algebraic structure of the function E is not generally known and so 'hill-climing' methods are applied. This approach explores locally about a point and adjusts the λ_i in such a way that $E(\lambda_1, \lambda_2, \ldots, \lambda_n)$ increases most rapidly.

Friedberg (35) used hill-climbing to explore the space of computer programmes in order to discover those which performed certain simple computations. The machine was not successful as each programme took of the order of 1000 times longer to produce

than pure chance would expect. Minsky (5, 36) attributes this failure to what he has called the 'Mesa Phenomenon' in which a small change in a parameter usually leads to either no change in performance or to a large change in performance. The space is thus composed primarily of flat regions or 'mesas'. Minsky holds that the Mesa Phenomenon generally crops up in difficult search tasks where it is hard to find "any significant peak at all" in the evaluation function.

Bremermann (10) has conducted several hill-climbing experiments in order to solve systems of linear equations and linear inequalities. He found that almost invariably the process stagnated at various points depending on the size and nature of each incremental change. He states later (34) that, "the stagnation phenomenon is extraordinarily common and seems to be a basic property of almost any optimization process".

The hill-climbing approach is sometimes satisfactory, but two particular disadvantages can render the method unworkable depending upon the type of problem under study. Firstly the search requires the computation of the gradient of E and hence the local direction of most rapid improvement. This means that for a problem with n variables, n partial derivatives must be computed and hence E must be recomputed n times. Normally E is not known explicity in terms of the variables λ_i, and so the gradient computation in fact requires n complete problem evaluations. For large n each step in the search can therefore require considerable computation.

The second disadvantage shows itself in those problems in which E shows no tendency to vary continuously with the λ_i. Implicit in the hill-climbing method is the assumption that the 'hills' are fairly smooth and the gradient will indeed indicate the best direction for improvement. If this turns out not to be the case and small changes in λ_i cause quite dramatic changes in E, then the computation of the gradient will not always indicate the best directions of improvement. If there is no continuity in E at all then the hill-climbing method would have little to offer over a random search (using E as an evaluation function).

2.5.3 Evolutionary Search

Problems in Pattern Recognition are nearly always characterized by the presence of a very large space of possible solutions. As has already been stated, it is normal for the computational burden to be reduced by the use of intuitive heuristics. These almost inevitably cause some reduction in performance by precluding a possibly large number of solutions. Hill-climbing search techniques are used successfully in those instances in which it is known that the search space has a 'smooth' structure. Without this knowledge the hill-climbing heuristic can become an impediment in those problems in which this continuity is not present. It is possible to generalize at this point by saying that whatever search heuristic is chosen for a problem, whether it be hill-climbing or any other rule, unless there is real evidence to show that it is suited to the problem, then there is a danger that the results produced will not be satisfactory.

In the absence of any a priori knowledge of a Pattern Recognition problem, intuitive search rules cannot be used without running the risk of obtaining inferior results. This means that to be sure of an unbiased set of results the researcher must either use a random search or an exhaustive search. A random search is of course, impractical as it is probably even less efficient than the exhaustive method. However, solutions are unlikely to be scattered at random throughout the search space, and so a search procedure which made use of any 'similarity' of closely scoring solutions would be much more efficient. An evolutionary search consisting of a series of relatively small random changes makes use of the 'continuity' of successive solutions providing the sizes of the random changes are chosen appropriately. A suitable choice of size for the change would maximize the real-time rate of improvement and thereby reduce the overall computation necessary for the search. Evolutionary searches are still vulnerable to the Mesa Phenomenon and will fail in much the same way as the hill-climbers if there is not sufficient 'continuity' in the search space. In these circumstances the size of the random changes must be increased so that any continuity which <u>does</u> exist can be utilized to minimize computation.

As soon as such an evolutionary search has produced some useful results, it is possible that some intuitive heuristics can be extracted and used to expedite the original search. Such heuristics would then already have sufficient empirical justification to demonstrate that they would not detract the search process. It should be emphasised again that heuristics chosen by intuition without <u>any</u> knowledge of the search space would offer no such guarantee. Chapter 4 describes a series of experiments in which an evolutionary search has been employed

in the design of an economical set of features for the machine recognition of OCR B printed characters. The method has identified several heuristics which enhance the search process.

2.5.4 Previous Work

The evolutionary search is by no means a new concept; it was perhaps first suggested in a machine intelligence context by Turing (37) in 1950. The first fairly extensive experiments based on the idea were carried out by Fogel et al. (1). Small finite-state machines were evolved towards predicting some extremely simple pattern sequences. Considerable emphasis was placed upon mimicing biological phenomena ('growth', 'mating') without regard for the fundamental principles involved. For instance, mutations to the finite-state machines were of several different types* each occurring according to a certain predefined probability. It was inevitable therefore that the structure of the particular type of finite-state machine employed was reflected in the results which were achieved. In other words the step-by-step changes were influenced largely by the representation rather than by any desired final form which the solution might take. For this reason the author feels that it would not have been fruitful to make any direct extension of this work. Solomonoff (38) considers that "the method of Fogel et al. should not be regarded in its present state as being particularly good for working any but the simplest problems". In considering the work of Friedberg (35) and Fogel et al., Michie (7) states that the "approach is now considered naïve, and nature tends to be thought a poor model for cost-conscious designers".

---

* change a state, add a state, delete a state, change initial state, etc.

An evolutionary experiment is described by Mucciardi and Gose (39) in which 100 handprinted characters were classified into 5 classes. 64 intuitively chosen property detectors were applied to the characters before recognition was attempted. Since the recognition mechanism incorporated 64 parameters each of which could have been altered independently, very little weight could be assigned to the results. However, it was observed that the greater freedom of the evolutionary process led to better results than those arising from a more restricted experiment.

Several authors (Chow and Liu, 40; Kaufman, 41) have reported experiments on so-called evolutionary searches in which intuitive heuristics have been the main guiding force. For instance, Chow et al. state that heuristics are used to limit the search so that the proposed procedure is computationally feasible. Kaufman directs his search by assigning intuitive probabilities to each of the allowable sorts of changes.

Klopf and Gose (47) describe an evolutionary pattern recognition method which was applied to an artificial problem involving only 16 4-bit patterns.

Lin (43) used an efficient search method in his solutions to various 'travelling salesman problems'. The search began with a random initial tour and evaluated all other tours which could be obtained by changing only 3 links. This continued until an improvement was found at which point the resulting tour was treated as the initial tour and the process repeated. In this way a succession of steadily improving solutions were evolved* until a locally optimum solution was obtained. It was important that the technique did not attempt to find the best improvement possible at each stage of the search, but rather

---

* Changes were not random but enumerative, however, their effect on the evaluation measure was not predicted in advance.

chose parameter values so that the probability of achieving an optimal solution within a certain time was maximised. Lin stated that in general the method of steepest descent tends to increase the computation time disproportionally and recommended that the domain of search be restricted, but only after sufficient information had been gathered so that good solutions were lost only with a very small probability.

In contrast Siklossy et al. (44) has been successful in applying exhaustive search methods to problems in theorem proving which had hitherto required a heuristic search for their solution. Their approach rested upon the construction of a search tree in which new nodes were generated using the rewrite rules in a strategic order. The various rewrite rules were categorized according to how quickly they increased the size of the search tree. During the search the rapidly expanding rules were only applied as a last resort after all else had failed. In this way the search was able to proceed for much longer before the size of the search tree became unmanageable.

Nilsson (45) describes and discusses several optimal search algorithms which can be used to search graphs or states spaces. These ideas are also extended to searches which carry out efficient problem reductions. Both sorts of search are founded upon the branch and bound concept of estimating functions which provide a bound for the cost of the various paths explored by the algorithm. Problems which can be formulated in a graphical form are probably best approached using such techniques providing also that sufficient heuristic (or analytic) information is available to prevent an undesirable number of nodes from being expanded.

Along with all other practical search methods, the evolutionary search relies upon a certain measure of 'continuity' in the search space and will stagnate if this continuity does not exist. In these circumstances the sizes of the evolutionary changes are increased until the search can utilize whatever continuity is present.

Previous work using evolutionary searches has not contributed significantly in the field of computer aided design. The criticisms of this work have been concerned with the simplicity of the problems attempted and the apparent inefficiency of the search. Work by Liu(43) has served to show that such searches can be made extremely efficient by using good heuristics.

2.6 Summary

This chapter has highlighted some of the computational difficulties facing the design engineer. Algorithms for the optimal reduction and decomposition of finite-state machines are frequently impracticable because of the sheer bulk of computation involved. Heuristics are therefore invoked in order that the methods might yield results without excessive processing. Pattern recognition tasks are almost always tackled using intuitive heuristics in an effort to reduce the computational requirements. Rarely is the application of rigorous theory sufficient to reach solutions to real world pattern recognition problems.

It was emphasised that there are dangers in the use of heuristics without prior evidence that they will not detract the search process. Heuristics restrict the number of trials required for a computation and thereby preclude solutions which lie outside the bounds laid down by the heuristic rules themselves. And further, it is not possible to tell how restrictive a set of heuristics may turn out to be unless the properties of the search space are known beforehand. The nature of solutions to certain pattern recognition problems for instance, can be completely unknown, and so the researcher often has no choice but to use possibly ill-founded heuristics.

An evolutionary search was proposed in order to postpone the choice of heuristics until after some results have been achieved. In this way the performance of any heuristic chosen can be broadly assessed before it is incorporated in the search.

3. EVOLUTIONARY DESIGN OF SEQUENTIAL MACHINES

3.1 Current State Reduction Algorithms

Several of the problems associated with the automatic design of finite-state machines were briefly described in 2.4. Although a perfectly sound formal framework exists for the theoretical treatment of such problems, the theory takes very little account of the serious computational difficulties which are encountered in practice. This chapter describes and evaluates a technique for the reduction of sequential machines and begins by surveying the methods currently available.

Grasselli and Luccio (21) describe a technique for the reduction of sequential machines in which an optimal reduction can be achieved without considering all possible compatible sets. In this method the maximal compatible sets are determined and then from these the prime compatible sets are formed. The desired reduction is obtained by selecting the minimal number of prime compatible sets which form a covering of the original states and which remain closed under all applicable inputs. It is shown that this selection problem can be equivalently formulated as an integer linear program. Such a program solves a set of inequalities containing as many variables as prime compatible sets with a cost funtion expressing that the number of selected classes be minimal. Grasselli and Luccio reduce the enormity of this selection problem by first drawing up the cover and closure implications in a covering and closure (CC) table. They then apply six sieve rules to this table by which various rows and columns can be eliminated. This leaves a substantially smaller CC table and hence a

relatively simpler integer linear program to reach the minimal reductions. It is worth noting that although the application of each rule leads to a valid reduction and does not preclude any optimal solutions, there appears to be no clearly defined order of application of the six rules in order to reach the smallest CC table.

House and Stevens (24) introduced a seventh sieve rule for the CC table and have achieved the reduction of a 22 state machine in 3 minutes of processing time. They stated however, that they were unable to give advice on the best order of application of their set of rules. The major disadvantage of reduction methods which set up cover-closure tables is the exponential rapidity with which the tables can increase in size as the machine size increases. For instance House and Stevens processed a 21 x 21 CC table for an 8 state machine and a 504 x 261 CC table for a 22 state machine.

Bennetts et al. (25) reduced the core store requirements by employing several intuitive heuristics and gave up the guarantee of an optimal reduction. Rather than selecting from the more numerous prime compatible sets, the method arrives at a reduction by choosing sets from only the maximal compatible sets and drastically reduces the possibilities which need to be explored in order to satisfy cover and closure requirements. It is stated however, that a considerable problem still remains in the construction of large maximal compatible sets. If such a set containing n states exists, then of the order $\frac{m!}{(m/2)!(m/2)!}$ compatible states containing m/2 states will require simultaneous storage at some point during the procedure. This quantity approximately doubles each time m increases by one.

Perryman (46) has investigated the problem of machine identification and has carried out a series of experiments aimed at the construction of finite-state machine models from a number of observations of an unknown prototype. Prototype machines of up to 40 states were studied and the work was extended to cases in which the start state was not known. The method used the concept of output consistent S.P. partitions in order to guide the choice of the model most likely to be correct. Experience gained during the construction of each model was carried forward to the next model. In order to check on the correctness of each model Perryman generated random input sequences and compared the outputs with those from the prototype. It was found that the identification procedure had most difficulty with prototype machines which possessed a large number of states having the same output, or two states which had very similar behaviour.

Existing state reduction algorithms generally suffer from huge computational burdens associated with the processing of compatible sets. The reduction technique described in the next section is based on a similar approach to the model testing method used by Perryman. It also relies upon the concept of state containment and does not make heavy demands upon core storage as machine sizes increase.

3.2 An Evolutionary State Reduction Algorithm*

The reduction algorithm described in this chapter avoids large core store requriements in two ways:

i) The employment of an evolutionary procedure.

ii) The restriction to reduction in which single states replace those which they contain (Appendix B.4) or are able to contain.

Restriction ii) means that an optimal reduction will not necessarily be achieved in those cases where it is possible to use a group of m states to do the work of another group of n states with n>m>1. Such a reduction is only describable in terms of overlapping compatible sets. It is suggested that cases in which ii) has a significant effect on the efficiency of the reduction are rare. For instance, of the examples in the literature (21, 23, 24, 47) only the machine described by Grasselli and Luccio (21) is prevented from being reduced to an optimal result (5-state reduced machine instead of 4) by restriction ii).

The evolutionary method proposed here detects probable state containment ('pseudoequivalence') by applying random input sequences to pairs of states and comparing the resultingpairs of output sequences. In this way very little more computer storage is needed than that required to store the given machine.

Suppose the given machine has the state set $(S_1, S_2, \ldots, S_n)$. Consider the ordered pair of states S_i and S_j and the p random input seuqences

$$X_{ijk} \quad (k=1, \ldots, p) \quad (p \leq 2^{n-1})$$

---

*The work described in this chapter has largely been published by the author (2, 3); see Appendix A.

all of length m. Each X_{ijk} is presented to the given machine, first with S_i as start state and then with S_j as start state. This results in two sets of p output sequences $\{Z_{ik} \mid k=1, \ldots, p\}$ and $\{Z_{jk} \mid k=1, \ldots, p\}$. If $\{Z_{ik}\} \not\equiv \{Z_{jk}\}$, S_i and S_j are said to be m-distinguishable as there exists an input sequence of length m which distinguishes the two states. If $\{Z_{ik}\} \equiv \{Z_{jk}\}$, (S_i, S_j) will be said to form a pseudo-equivalent (p-e) state pair.

The first state S_1 is selected and tested for pseudo-equivalence with all other states. If in this process a p-e pair (S_1, S_i) is discovered, S_i is replaced by S_1 wherever it occurs in the state table and the redundant state S_i is removed. After the discovery of all such p-e pairs (S_1, S_i), the procedure returns to select and compare the next unselected state with those states not already removed. In this way, any pseudo-equivalences which are discovered reduce the number of the remaining comparisons. It is found in the discussion on 'don't care' conditions that pseudo-equivalence is not a symmetric relation. This means that, in the worst case

$$(n-1)+(n-1)+ \ldots +(n-1) = n(n-1) = n^2-n$$

state comparisons might be made, and hence that the processing time does not increase more rapidly than the square of the number of states. It is observed that the computation associated with conventional reduction procedures which do not possess advance information on the machine structure, must increase at least as n^2 in order to detect all possible compatible state pairs. (Usually the computation is much greater than this; see 3.1).

The algorithm inputs each random sequence simultaneously to the pair of states under comparison. In this way, as soon as

different outputs are obtained, the process can be terminated with the unambiguous decision that the two states are nonequivalent. It is probable that any difference would appear early in the first input sequence, and so the algorithm makes nonequivalence decisions quite quickly. The bulk of the processing time occurs in the evaluation of the potential pseudo-equivalent states. If the states entered from each of the two states under comparison are ever simultaneously identical, then the current input sequence is terminated and the process continued with the next in the set of p input sequences. This is because a nonequivalence decision is impossible once the two paths in the state diagram have merged. At the end of the procedure a reduced machine is left which possesses no compatible states.

3.3 Incompletely Specified Machines

These ideas can now be applied to the much more difficult case where the machine is only partially specified. There are two sorts of 'don't care' conditions which can arise in a partially specified Moore or Mealy machine:

a) the next state transition is not specified

b) the output is not specified

These conditions are assigned values such that (S_i, S_j) would not be prevented from forming a p-e state pair, that is, entries are assigned in such a way that S_i is made to contain $S_j (S_i \supset S_j)$ where possible.

Let S_{i_n} and Z_{i_n} be the state and last output after the nth transition from S_i. The various actions and assignments in the algorithm are best summarized in a table:

	SPECIFIED	NOT SPECIFIED	
		ASSIGNMENT	ACTION
S_{i_n}	If $S_{i_n} \equiv S_{j_n}$ then t.c.s.	S_{j_n}	t.c.s.
S_{j_n}	else continue	not assigned	t.c.s.
Z_{i_n}	continue	Z_{j_n}	continue
Z_{j_n}	continue	not assigned	continue

t.c.s. = terminate current input sequence and continue the state pair comparison with the next in the set of p input sequences.

continue = continue with next input in current sequence.

TABLE 1

To illustrate the philosophy behind the treatment of don't care conditions, consider the following Mealy machine:

states	inputs a	b	c
A	C/0	E/1	B/0
B	D/0	D/1	B/0
C	A/1	C/1	E/1
D	A/-	-/-	D/1
E	A/0	B/1	E/-

outputs are shown alongside the state transitions

Suppose state B is being checked for possible state containment in state A. Then from the A and B rows in the table,

$A \supset B \Rightarrow C \supset D$ and $E \supset D$ (B clearly contains B)

continuing, $C \supset D \Rightarrow E \supset D$

To enable $C \supset D$ the entry 'A' in Da is written into Ca

To enable $E \supset D$ the entry '1' in Dc is written into Ec. Having modified the table in this way, $A \supset B$ and state B can therefore be replaced by A without affecting the function of the machine.

This same procedure is carried out by applying the input sequences aa, bca, to the state pair (A,B) and following the instructions in Table 1. The state pair (A,B) is a p-e pair and so state B is removed as described in 3.2.

Several reduced versions of the machine may exist; the reduction which is in fact carried out is dependent on the order in which state comparisons are made (Figure 1).

	1	2	3	4
A	G/1	-/-	-/0	-/-
B	-/-	B/-	E/-	-/-
C	-/-	-/1	H/-	E/-
D	-/-	-/0	-/-	A/-
E	G/0	E/-	B/1	-/-
F	-/-	I/-	-/-	B/0
G	D/-	D/-	-/0	E/-
H	-/-	C/-	-/-	D/-
I	F/1	C/0	-/0	-/-

	1	2	3	4
A	C/1	A/0	E/0	A/1
C	A/-	A/1	H/0	E/-
E	C/0	E/-	A/1	-/-
F	-/-	H/-	-/-	A/0
H	F/1	C/0	-/0	A/1

ordering: A,B,C,D,E,F,G,H,I
partition: $\{\overline{A,B,D};\ \overline{C,G};\ \overline{E};\ \overline{F};\ \overline{H,I}\}$

	1	2	3	4
A	G/1	H/1	H/0	B/0
B	G/0	B/0	B/1	A/1
G	B/-	B/-	-/0	B/-
H	A/1	A/0	-/0	B/1

ordering: A,C,B,D,E,F,G,H,I
partition: $\{\overline{A,C,F};\ \overline{B,D,E};\ \overline{G};\ \overline{H,I}\}$

	1	2	3	4
A	C/1	C/0	-/0	A/1
B	C/0	B/-	B/1	-/-
C	A/-	A/1	A/0	B/0

ordering: A,D,H,I,B,C,E,F,G
partition: $\{\overline{A,D,H,I};\ \overline{B,E};\ \overline{C,F,G}\}$

FIGURE 1 : Prototype and Three Reductions

3.4 Reliability of Reductions

Each step in the reduction algorithm is determined from the evaluation of a random input to the machine. In this way the process of reduction is an evolutionary procedure in which each change that is made to the machine has no watertight logical support, but is verified empirically by the evaluation of p input sequences (of length m). Absolute certainty of the validity of the reduced machine would be achieved if all possible input sequences ($p=i^{n=1}$; i=no. inputs) were used in the evaluation. However, if this is done, the technique becomes exhaustive and suffers from the same troubles as other reduction methods. At the outset of this research the author felt that substantially smaller values of p than i^{n-1} might only be necessary for quite high confidence in the results. Since the type-a don't care condition terminates an input sequence, p should certainly be larger than the number of type-a conditions.

It was stated (2) that any errors that were present in the final reduced version of the machine could be detected by a process of verification. The reduction was then recycled until all errors were eliminated. Naturally the amount of recycling would be reduced by increasing p, and hence p should be adjusted empirically until the overall processing time for a correct reduction is at a minimum. However, Bennetts (48) has said that "verifying the reduced state table obtained by pseudoequivlanet merging is not a trivial problem". He argued that the determination of the functional equivalence of two machines could only be achieved by an exhaustive procedure. This is certainly true in general, but in this particular case Bennetts failed to make use of other information provided by this method.

Each reduction which is correct is equivalent to the determination of a partition whose blocks are compatible sets of states (Figure 1), together with the merging of all the states in each into one. It follows that the reduction is verified by checking the compatibility of the states in each block of the partition and making certain that the merging process has been complete. This means that errors can be detected simply by ensuring that:

i) The partitions are output consistent.

ii) The partitions are preserved for all applicable inputs, and incomplete merges are eliminated by

iii) assigning don't cares in the final machine where necessary.

All three checks involve no more than a single scan over the state table. It is noted that i) and ii) are sufficient to demonstrate the validitiy of the partition, and iii) merely checks that the merging states are not only compatible with, but also contained by the merged state. For instance the incorrectly reduced machine described in Appendix A is based on the partition $\{\overline{A}; \overline{B,E}; \overline{C,G}; \overline{D}; \overline{F}; \overline{H,I}\}$ which is not preserved since $\overline{C,G} \rightarrow \overline{F,B}$ under input $\emptyset$.

3.5 Results

Some empirical evaluation of the reduction method was carried out at Southampton University by an M.Sc. student and a 3rd. year undergraduate and their results are summarized here.

Scott (49) carried out a series of experiments aimed at determining the relationship between the probability of a faulty reduction and the values of the parameters m, p and n.

It was observed that input sequences very seldom reached lengths of greater than n-1 during the reduction process. This meant that longer sequences would not effect the error probabilities to any very great extent. The maximum length of each input sequence (m) was therefore held at n-1 and not used as a variable parameter during the investigation.

A number of machines were reduced using values of p ranging from i^{n-1} (i=no. inputs) to unity. For each value of p the machines were reduced n(n+1)/2 times using each possible pair of states to initiate each reduction. The reductions were verified exhaustively and the probability of an error plotted against p for each machine. Three machines (A,B,C, in Table 2) of different internal structures were studied in this way. It was found that in all cases the error probability estimates decreased steadily to zero as p increased. The smallest values of $p(p_{min})$ for which no error occurred are given in Table 2. These values represent a 70% reduction on the number of all possible input sequences and which would be required in an exhaustive check.

Machine	A	B	C
No. initial states	5	5	6
No. final states	3	5	3
Average computation time (sec.)	7	8	12
p_{min}	51	64	41

TABLE 2

Error probabilities for larger machines were not computed in this project because the verification procedure required that all possible input sequeces be applied to the state pair (48) and hence demanded prohibitive amounts of computation.

Thomas (50) employed the more efficient verification procedure described in 3.4, but his experiments were again hampered by the limited availability of the computer and he was unable to obtain reliable results on larger machines.

Both Scott and Thomas concluded that the main advantage of the reduction method was its relatively low core store requirement. Both again agreed however, that there was reason to believe that this advantage would be greatly outweighed by the impracticable lengths of computation which would be required when larger machines were processed. In order to resolve this point the author carried out some experiments which determined the reliability of the reduction method when applied to the 4 input 22 state machine used by House et al. (24), (Figure 2).

The 126 incompatible state pairs in this machine were first determined manually using an implication chart. The 22 state machine together with the 126 state pairs were then supplied to a computer programme which attempted to recognize the incompatible states by applying p random input sequences of length 21 in the manner of the pseudo-equivalent reduction technique. If one of the 126 state pairs was not seen to be incompatible after the application of p input sequences, then this would have corresponded to an invalid reduction had the two states been merged. In this way the reliability of the

	a	b	c	d
1	1,-	2,2	-,1	8,2
2	-,-	3,2	5,1	9,-
3	-,-	4,-	6,1	10,2
4	-,-	-,-	-,-	11,2
5	1,-	-,2	-,-	-,-
6	1,-	3,2	12,1	-,-
7	-,-	4,2	-,1	-,-
8	-,-	2,-	-,-	14,2
9	7,2	-,-	-,-	15,2
10	7,-	-,-	6,1	16,2
11	1,2	4,1	5,-	17,2
12	1,2	2,1	-,-	-,-
13	-,-	3,1	-,-	-,-
14	-,-	-,-	13,2	18,2
15	7,1	-,-	12,-	-,-
16	7,-	2,1	-,2	-,-
17	1,1	3,1	-,-	-,-
18	1,-	4,-	-,2	19,-
19	-,1	-,1	5,-	20,2
20	-,2	-,1	5,-	21,2
21	-,2	-,2	5,-	22,2
22	-,1	-,2	5,-	14,2

FIGURE 2 : Reduction Example from House et al. (24)

pseudo-equivalent reduction method on this 22 state machine was studied for various values of p. The set of 126 incompatible states was processed several times for each value of p and the total number of errors recorded (Table 3). This enabled a probability of error to be computed and plotted against p (Figure 3). It was observed that all the errors attributed to values of $p \geq 1000$ arose from the single incompatible state pair (1,22). Upon investigation it was found that this incompatible state pair required the longest distinguishing sequence (of length 4). In order to be reasonably certain that this incompatible pair had been identified and hence that the reduction process would have yielded a valid result, 5000 or more random sequences had to be applied.

No. Sequences (p)	No. Runs (q)	No.Errors in q runs	Probability of error
1	111	11249	0.80
3	10	687	0.54
10	111	362	0.26
30	10	156	0.12
100	11	49	0.035
300	10	16	0.013
1000	42	28	0.0053
2000	50	11	0.0017
3000	10	1	0.0008
5000	10	0	-
10000	50	0	-
15000	50	0	-
20000	50	0	-

TABLE 3

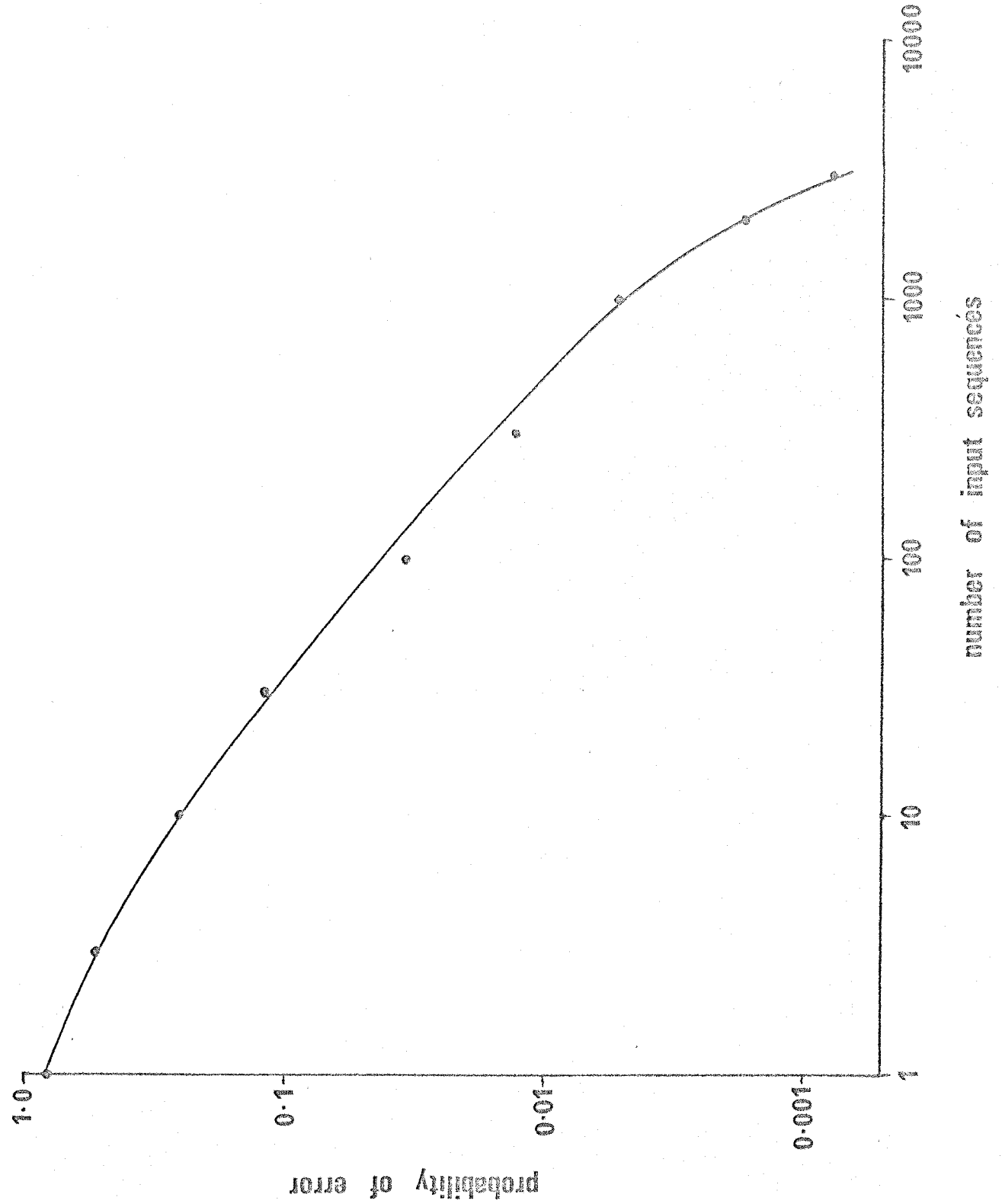


Figure 3. Probability of error in the reduction of a 22 state machine

3.6 The Pseudo-Equivalent Approach and Alternatives

The results described in the last section are sufficient to reveal a severe shortcoming in the pseudo-equivalent reduction process. The reliability of the method is seen to be clearly dependent upon the number of random sequences which are used in each state pair evaluation. However, this reliability is also dependent upon the structure of the machine being reduced. It was seen that invalid results can be very difficult to eliminate if the machine being reduced contains pairs of states which can only be distinguished by long sequences.

In the 22 state example it was discovered that only one pair of states required a sequence of length > 4 to distinguish them. This meant that it would have been sufficient to use all 256 possible input sequences of length 4 during the checks for incompatibility. This would have led to a much more efficient search. However, in general the longest minimal distinguishing sequence for all pairs of states is not known, and hence an accurate bound on the number of input sequences cannot be determined without further computing effort.

These results do not invalidate the merging process itself but point out the slow and inefficient (in terms of processing time) way in which it was applied. The method can be made exact by algorithmically compiling the list of all implied state containments and checking each one in turn for compatibility. This will generate a variable core store requirement which could increase as n^2, but only in those cases in which the machine was highly redundant.

As an example consider the prototype machine described in figure 1 using the state ordering A,D,H,I,B,C,E,F,G. The reduction is carried out as before except that each state pair is checked for

i) output compatibility and

ii) implied state containments.

The checking of implied state containments is carried out in the same way as an initial state pair and continued until no state containments are implied or an incompatibility has been found. This process can be outlined for the example as follows:

$A \supset D$ after merging with no implied containments.

$A \supset H$ after merging with no implied containments.

$A \supset I \Rightarrow G \supset F$ after merging.

$G \supset F \Rightarrow E \supset B$ after merging.

$E \supset B \Rightarrow B \supset E$ after merging.

$B \supset E$ after merging with no implied containments. This information has enabled states $\overline{A,D,H,I}$, $\overline{B,E}$, and $\overline{GF}$ to be merged. Only one state comparison remains.

$C \supset G$ after merging with no implied containments. This process gives rise to the third reduction in Figure 1.

This algorithm has not been programmed but nevertheless has been applied manually to most of the examples in the literature. Reduction of the largest of these (22 states) takes no more than a few minutes and a single sheet of paper.

3.7 Summary

Existing algorithms for the reduction of Finite-State Machines have been surveyed and a new approach has been proposed. The new method detects probable state containment by applying random input sequences to pairs of states and comparing the resulting pairs of output sequences. Any state which appears to be contained by another is merged into the containing state. Reduction is completed when no state can contain any other.

Experimental work on this problem has been described and has revealed a deficiency in the approach. It was found that incorrect reductions could only be avoided by employing impracticable amounts of computation. It was shown that machines which possessed pairs of states which could only be distinguished by long input sequences (of length k, say) could not be reduced accurately without allowing the volume of computation to depend exponentially upon k. This conclusion compared with Perryman's (46) findings in the problem of machine identification.

4. DESIGN OF LOGICAL CIRCUITS FOR OPTICAL CHARACTER RECOGNITION

4.1 The O.C.R. Problem

4.1.1 Matrix Matching

The requirement for a machine to read written material has existed ever since computers first began processing large quantities of data. The first commercial optical character recognition (O.C.R.) machine appeared over 17 years ago, but even so the market has not expanded as it had been expected (51). The most popular type of recognition to be described in the literature and to be used in commercial devices, is the 'matrix matching' scheme (52). It is probably chosen for its fast simulation speed on a digital computer and its simple implementation. Attention in this chapter is confined to the problems associated with the recognition of printed characters using such techniques, beginning first with a description of a typical system (53).

Five basic elements can be distinguished in the system illustrated in Figure 4. The print line on the document passes over an area illuminated by lamps; the reflected light is then collected by a lens and imaged on a vertical line array of photodiodes. The video-circuits amplify these and quantize them into black and white grid points. The matrix is a register where the incoming grid points are assembled to form an electrical image of the part of the line which has just been scanned. The extraction logic analyses the electrical image continuously and accumulates at the appropriate time a measure of fit between the character pattern in the matrix and each of the character classes to be recognised. Finally the decision

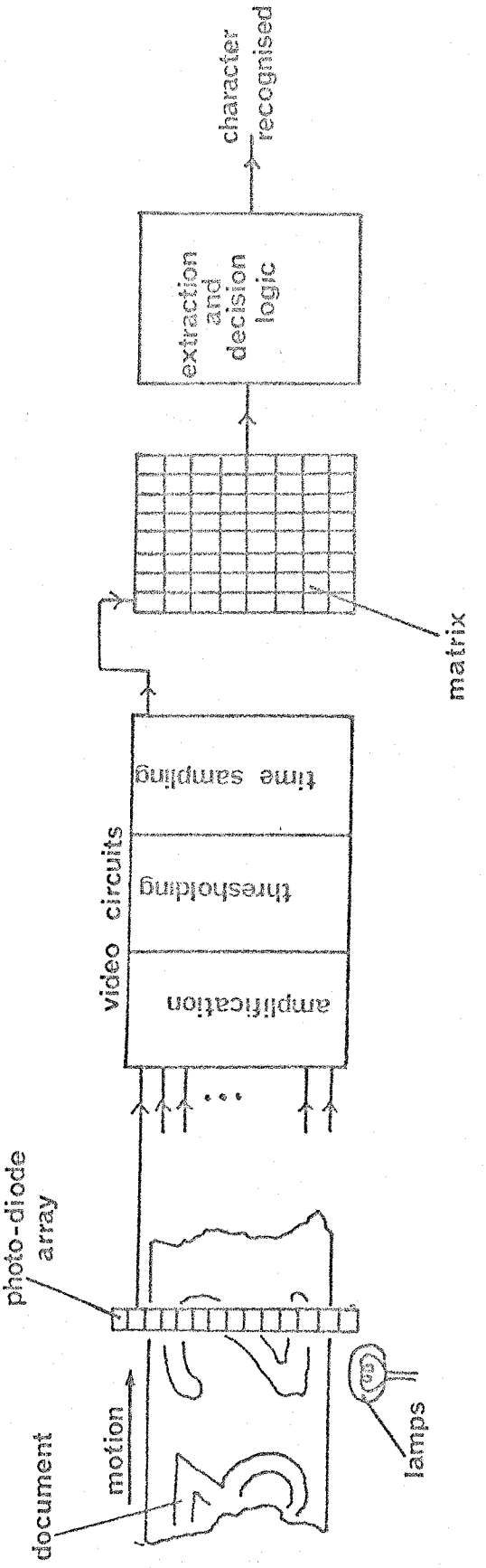


Figure 4. Block diagram of a read and recognition system.

logic inspects these measurements of fit and decides whether a particular class should be recognized or whether the pattern should be rejected.

The matrix is a long serial shift register which brings the binary patterns to be recognized into all possible translation positions. As the pattern moves across the matrix the extraction logic checks through a specified set of spatially related groups of points (called operators)for matches. Each point or element in the operator is either black-seeking or white-seeking, fitting black or white matrix points respectively. The whole operator matches only if all its elements fit simultaneously at least once somewhere on the matrix (Figure 5).

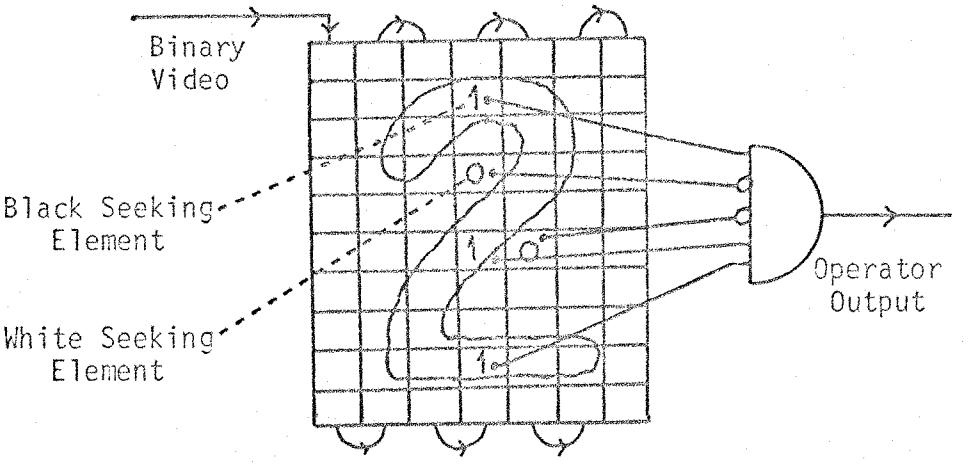


FIGURE 5 : Matrix shift register showing an operator which matches 2's

This information is passed to the decision logic enabling a character classification to be given. The major problem area lies in the design of operators which are best suited for the recognition of characters. Normally this task is carried out by a human designer who uses intuition to produce acceptable operator logic. This chapter examines the problem and later employs an evolutionary search procedure in the automatic design of operators.

4.1.2 Print Quality

The difficulty of operator design rests almost entirely upon the variety of print quality which the machine is expected to encounter. The performance of all O.C.R. machines is critically dependent upon the print quality, a factor which has proved to be very difficult to define (51). In order to clarify this point, print variability can be broken down into the following component parts:

1. Paper reflection.
2. Ink density.
3. Voids.
4. Spots.
5. Limb width.

Further parameters that add to the difficulties are positional in nature and can be identified as:

1. Vertical Position.
2. Horizontal Position.
3. Tilt.
4. Adjacent characters.

Finally there are two character variations, probably best described as:

1. Size.
2. Character font. eg. OCR A, OCR B.

Variations due to vertical position and horizontal position can be largely eliminated by an exhaustive scanning mechanism (4.1.1) which enables all possible character positions to be processed.

In theory the binarization or thresholding of the incoming analogue characters would be carried out perfectly if the characters were first recognized and then replaced by archetypes. It is the author's opinion therefore that in general, attempts to remove spots or fill voids (etc.)before recognition takes place can only destroy information and at best only maintain the overall recognition capability. Of course, this would not be true where specific types of degradation were known to be occurring. It might be concluded that good thresholding occurs where only a minimal amount of information is lost during the transition from analogue to digital signals.

The problem of separating adjacent characters has not been solved algorithmically (54). However, it can be approached in two other ways: either the scanning operation can be 'pulsed' so that 'read-out' occurs at certain times corresponding to the instants when characters are in the correct recognition position, or the difference between characters can be used in such a way that only when a character is in the correct recognition position does the output exceed some predetermined threshold (55). The first method is certainly appropriate in those cases in which the spacing of the characters is known (e.g. line printer outputs). In general this information is not available in advance, and the recognition logic must take the additional burden of distinguishing character contiguities from the characters themselves.

It can be seen that the design of operators for the recognition of even a single font can be extremely difficult if all aspects of print degradation are to be taken into account. The next section supplies some background information on the sort of performance currently obtained in OCR systems.

4.1.3 Current OCR System Performance

Explicit statements describing the performances of commercial OCR machines over all types of character degradation are not normally available. This is not because the manufacturer is unfamiliar with his equipment, but because there is no standard by which character degradation can be measured. This means that any performance figures which are issued could mislead the customer. Faced with an OCR requirement, it is generally safer for the customer to approach several manufacturers with the identical sets of data which are typical of his own recognition problem. The various performances can then be compared not only in terms of error and reject rates, but also in terms of capital cost and throughput speeds.

Nevertheless most manufacturuers of OCR equipment do give an indication of the performance of their machines in ideal conditions. Probably the best performance by an OCR document reading machine at the lower end of the price market is that achieved by the Plessey 4200B. This is a machine which reads OCR B numerics plus four symbols. It is capable of a reject rate of 1.5 in 10^6 and a substitution rate an order better than this. These figures were derived from 'quite good quality' characters obtained from several types of 'well maintained' line printers.

The IBM 1975 Optical Page Reader was designed to accept printed characters of over 200 fonts from quarterly employer forms. The report (56) indicated that the reject rate on a per character basis lay around 1%, but no indication of the character quality was given.

The most recent optical page reader to be announced in Britain is the Mullard X1300. The manufacturers state that the performance is almost without error providing the print has been obtained from recommended electric typewriters and ribbons and on recommended paper. Under these circumstances the user can expect no more than 1 in 10^5 reject and 1 in 10^6 substitution errors.

Error and reject rates when studied in isolation can be viewed in two ways; either they can be taken as measures of the performance of the recognition system, or as measures of the quality of the data which is recognized. An indication of the high character quality requirements of most OCR systems is seen in their failure to meet the specification laid down by the Post Office for automatic letter sorting. The Post Office state that as many as 20% of letters can be rejected as unreadable by the machine but no more than one error in 2000 characters is tolerable. In order to read the postcode the machine must accept about 20 classes of character of any font and of any degradation. Such a machine is not within the current state of the art in OCR.

4.2 Previous Automated Design Methods

A great deal of literature has been produced on many aspects of Pattern Recognition all of which could be said to be related to OCR. References in this section will only be selected from those papers which contribute directly to matrix matching aspects of OCR.

Bledsoe and Browning (57) were among the first to report a computer-automated scheme for the design of recognition logic. They employed a 10x15 photodiode array and selected a number of randomly generated sets of n points on the array. Each set of n points could take one of 2^n states depending on the pattern present in the 10x15 binary array. During training each state of each set of n points was labelled with the names of the characters which gave rise to it. Test characters were identified by passing a majority vote over the states activated by the test character. The method shows little hope of extension in view of the rather wasteful table-look-up nature of the process. However, the work carried out demonstrates that matrix matching methods do possess high discriminative power.

At about the same time Evey (58) described an intuitive technique which he used to solve a 14-class problem. This time the computer was only used as an aid in the manual design of operators which matched co-classed characters. Evey did not

favour automatic methods and made the following comment, "Most automatic procedures can be ruled out due to the astronomical number of possible logics (operators), but useful procedures have been developed by limiting the complexity of the conditions used in the statements. However, possibly because of this limitation, statements so produced have never been as successful in practice as those designed by people". (cf. Heuristics, 2.3).

Uhr and Vossler (59) were next to propose an automatic method in which 40 5x5 binary operators were scanned across the input matrix for matches. The positions of match enabled 4 "characteristics" per operator to be associated with each class of character. The recognition took place when these pattern characteristics were compared with a reference list. Apart from manual and random generation, operators were also derived heuristically according to the following rules:

a) The 5x5 matrix was extracted from a random position in an input character.

b) All 'zero' cells connected to 'one' cells were replaced by blanks.

c) Each of the remaining cells, both 'zeros' and 'ones' was then replaced by a blank with a probability $\frac{1}{2}$.

d) Tests were made to ensure that the operator did not have 'ones' in the same cells as any other currently used operator or any operator in a list of those recently rejected by the program.

These rules are intuitive heuristics and serve only to restrict the search space to what are thought to be more fruitful areas. Some elegant solutions are certainly precluded by b) (see 4.4).

The system was improved by adjusting various weights given to each operator; the operators themselves were left unaltered except by possible replacement. The sizes of the design and test sets (<100) in relation to the number of adjustable parameters (>160) and the number of operators, does not provide a convincing demonstration that the system would be economically viable if larger data sets were used.

Zobrist (60) later extended this work to a two layer scheme but contributed nothing new to the detailed design of the operators.

Kamentsky and Liu (61) also felt that in an automated method "there must be a way to restrict the number of switching functions that are to be considered in the design procedure. In effect, an efficient search procedure for logic must be found". They achieved this be generating random operators within certain heuristic spatial constraints. These constraints were intuitive and chosen to emphasise local features of the characters,but more importantly,they prevented a large number of potential solutions from even being considered. 3000 operators were generated and 75 were selected according to an information measure which favoured operators which matched one half of the set of characters. A Bayes' decision was used to obtain 0.3% reject and 0.15% error rates on a data set of 1300 samples

containing 26 classes. It was discovered that operators having around 5-7 elements commonly achieved the greatest discriminative power.

Later Liu (62) extended the work of Kamentsky et al. and achieved better results by applying an information measure to the actual choice of operator elements. Characters from the various classes were position normalized on the matrix and the information measure of each matrix point calculated. This enabled a list of matrix points of high information to be formulated from which sets of 5-7 elements were randomly chosen to compose operators. In other words, rather than use the spatial constraint of Kamantsky to restrict the number of possible operators, an information measure heuristic was used instead. One result on 2000 characters of one font consisted of 0.1% errors and 0.35% rejects using 96 operators in a 26 class problem. A later paper by Liu et al. (63) described a more extensive experimental investigation into the performance of a similar set of 96 operators, but again no clear indication of the print quality was given. The authors did state however, that "very little poor quality print was contained in the data set". It is admittedly difficult to describe print quality, but until some attempt is made, the various results cannot be compared even subjectively.

The research at IBM by Kamentsky and Liu was given a critical appraisal during the design of the IBM 1975 optical page reader (56). It was found that the performance deteriorated by an order of magnitude when realistic data were used. Moreover "because of the automated design procedures, there was no clear idea of what each component was supposed to do",

and hence it was difficult to modify and improve the machine. The project therefore turned to intuitive methods and enlisted the help of a system 360/model 40 with 256K bytes of memory interfaced to a complete reader. 96 operators and about 2000 character responses were built into the final machine. Operators were far more complex than those reported in earlier papers; an example described represented a lengthy 2-level boolean function of 92 matrix points. Each operator was designed for a particular task, often this was the resolution of a specific confusion pair (e.g. I,T). The authors concluded that "An automatic algorithm to replace the designer would have to be qualitatively different from the kinds that are presently available".

This philosophy has extended to the design of the IBM 1275 recognition system (53), and an interactive system for reading unformatted printed text (64). In both systems human intuition played a major part either in the design of the recognition logic or the recognition itself. Moreover recognition was carried out by straightforward methods of correlation with standard characters from each class.

In the design of operators for a Plessey optical reader Britt (65) was also sceptical about computer-automated approaches. "Unfortunately it is necessary to make a large number of simplifying assumptions in order to keep the computer time down to a reasonable level. Although computer simulation has been used during the project in the design of the features (operators), it has been concluded that simplifying assumptions cause the results to be of very limited value".

The first attempt to design a set of operators with some regard to the total structure of the operator set was perhaps by Coombs (66). He argued that operators should be selected only if their binary (fit/no fit) responses to the various character classes coincided with a column of a boolean orthogonal matrix (Figure 6).

Columns represent 11 operator responses.

```
1 1 1 1 1 1 1 1 1 1 1
0 1 0 1 1 1 0 0 0 1 0
0 0 1 0 1 1 1 0 0 0 1
1 0 0 1 0 1 1 1 0 0 0
0 1 0 0 1 0 1 1 1 0 0
0 0 1 0 0 1 0 1 1 1 0
0 0 0 1 0 0 1 0 1 1 1
1 0 0 0 1 0 0 1 0 1 1
1 1 0 0 0 1 0 0 1 0 1
1 1 1 0 0 0 1 0 0 1 0
0 1 1 1 0 0 0 1 0 0 1
1 0 1 1 1 0 0 0 1 0 0
```

rows represent operator response to each of 12 classes of character.

0 = no fit

1 = fit

FIGURE 6 : Boolean orthogonal Matrix

This meant that a complete set of operators which all satisfied this requirement would give rise to minimal confusion between classes. In fact the operators in such a set are optimal in the sense of their independence of each other; no redundant information is extracted nor are any measurements duplicated. Such a scheme does not need to be designed perfectly because its very construction ensures a large tolerance in the variability of operator responses. However, as Coombs points out, "we shall at a later stage require to find the n-tuples (operators) which divide the categories in the desired manner". The paper does not report a satisfactory solution to this problem.

Summary

The automatic generation of operators for character recognition has not met with sufficient success to meet customer requirements. In every case heuristics have been invoked to aid the search for operators. The search effort has been reduced for example, by applying spatial constraints, by using information measures, and by reducing the size of the operator.

Successful OCR projects have in most instances relied entirely on human intuition in the design of operators. In these cases operators have been individually designed to carry out tasks which can be assessed quickly and easily by the human designer.

It is interesting to observe that the number of operators employed in both automatic and intuitive schemes rarely exceeds 100. It might seem that by introducing a new operator and eliminating another source of confusion, the overall error rate will be reduced. However, each new operator necessarily gives rise to a small number of spurious matches, and this in turn will introduce substitution errors. If the number of additional errors exceeds the number removed, then the extension to the recognition logic becomes a disadvantage. In this way, if operators are intuitively designed to correct smaller and smaller sources of error, a point (around 100 operators) is reached at which no further improvements appear to be possible. It is suggested that operators which are designed to correlate with single classes or resolve a single confusion pair are particularly prone to this effect. That is, this sort of operator when working on noisy data, is very likely to fail in the single task for which it was designed; it can then do no other than damage the overall recognition performance. Only those operators which are capable of distinguishing many character classes from many others would be robust enough to provide information from poor quality data. The type of operators proposed by Coombs are eminently suitable in this respect although he gives no working method for finding them. Certainly it would be far too difficult to design them intuitively.

The next section proposes an evaluation function which together with an evolutionary search, enables a recognition system to be designed automatically. The search is not hampered by a priori heuristics and the evaluation routine ensures that the operators keep many of the advantages advocated by Coombs.

4.3. The Evolutionary Scheme

4.3.1 Flores and Grey Criterion

Flores and Grey (67) have described a theoretically optimum scheme against which any set of features for character recognition may be rated.

Consider a set of N features (operators).

A reference character C_i, from the i th class will give rise to a response vector $\underset{\sim}{F}_i$ where

$$\underset{\sim}{F}_i = \{f_{i1}, f_{i2}, \ldots, f_{in}\}$$

and where f_{ij} is the response of the j th feature to the i th reference.

Similarly the j th feature corresponds to a vector response $\underset{\sim}{f}_j$ from the K classes where

$$\underset{\sim}{f}_j = \{f_{1j}, f_{2j}, \ldots, f_{kj}\}$$

A test character T will correspond to the response vector

$$\underset{\sim}{F}_T = \{t_1, t_2, \ldots, t_N\} \ ,$$

and the decision as to which of K classes the unknown character belongs is made by examining K cross correlation functions

$$\phi_i = \frac{\underset{\sim}{F}_i \cdot \underset{\sim}{F}_T}{|\underset{\sim}{F}_i|} \qquad i = 1,2, \ldots, K.$$

and choosing the largest ϕ_i. It is necessary to normalize the reference vectors $\underset{\sim}{F}_i$ in order that decisions are not biased in favour of references with large moduli.

Let $\hat{\underset{\sim}{F}}_i = \frac{\underset{\sim}{F}_i}{|\underset{\sim}{F}_i|}$ be the unit vector in the direction of $\underset{\sim}{F}_i$.

The decision process can be restated as the discovery of the i which corresponds to the largest value of $|\underset{\sim}{F}_T| \cos \theta_i$ where θ_i is the angle that $\underset{\sim}{F}_T$ makes with the i th reference vector. T is therefore assigned the class corresponding to the reference vector which subtends the smallest

angle to $\underset{\sim}{F}_T$.

The effect of noise on the decision is now considered. In figure 7 there are two unit reference vectors $\hat{\underset{\sim}{F}}_i, \hat{\underset{\sim}{F}}_j, \theta_{ij}$ is the angle between them, and $\angle AOB = \angle COB$.

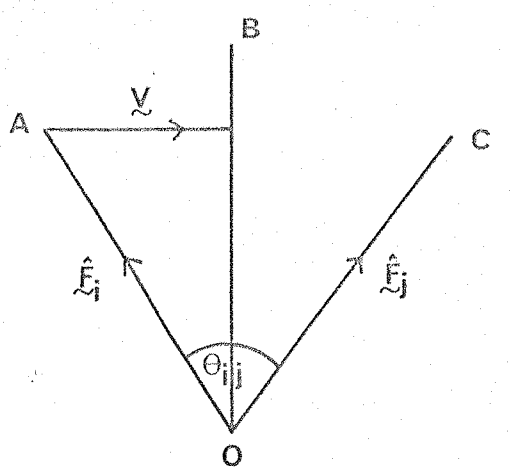


FIGURE 7 : Determination of minimum tolerable noise $\underset{\sim}{V}$.

Any vector from O in the plane and lying within the angle AOB is identified with the i th class; any vector lying in the angle BOC is identified with the j th class. The least noise which would corrupt $\hat{\underset{\sim}{F}}_i$ is $\underset{\sim}{V}$ with $|\underset{\sim}{V}| = \text{Sin } \theta_{ij}/2$, where $\underset{\sim}{V}$ is the noise vector whose components are independent random variables*. The minimum corrupting noise for the system thus depends upon the smallest angle which exists between any two vectors in the system. In this way, maximizing the smallest angle between any two vectors in the system of reference vectors constitutes an optimization of the system.

Flores et al. now showed that the largest angle α such that the angular distance between any pair of reference vectors is at least α is

$$\alpha = \cos^{-1}\left(\frac{-1}{K-1}\right) \quad \text{for } N \geq K-1$$

---

* It is assumed that each feature is affected by noise in the same way.

In other words the best discrimination and immunity to noise that can be achieved occurs when $\alpha = \cos^{-1}(-1/K-1)$. Moreover this maximum can be obtained when the unit reference vectors lie on the vertices of a regular K-1 dimensional simplex inscribed in the N-dimensional unit sphere.

More significant is the fact that α is independent of N. If the number of features, N, is increased beyond K-1, the optimum solution will not improve. However, it is true that an increase in N beyond K-1 will also increase the degrees of freedom for the choice of features which comprise an optimum or near optimum system. This makes it easier to comply with the constraints imposed by the geometry of the characters and still construct a near optimum system. It is worth noting at this point that Coombs (66) allows no degrees of freedom in his scheme. Flores et al. concluded that the most economical and optimal solution occurred when the number of features was equal to one less than the number of classes (N=K-1). They did not, however, attempt to set up rules or methods for constructing such a set of features.

It might be deduced from this work that if such a feature set is to be capable of any improvement then both the number of features and the number of classes must be increased. It is important to appreciate that increasing the number of classes and features does not significantly damage the optimum angular separation of the class reference vectors $\underset{\sim}{F}_i$. Indeed

$$\lim_{K\to\infty} \cos^{-1}(-1/K-1) = \frac{\pi}{2} .$$

However, if reference vectors are added without increasing the dimensionality of the space, then this angular separation decreases to zero. Extensions to the Flores et al. optimum system are considered in more detail in chapter 5.

4.3.2 Scoring Scheme

In this work an evaluation function is required which scores operators in such a way that the angular separation of the various reference vectors is maximized.

Consider a K-class character recognition problem. Without loss of generality let $f_{ij} = +1$ if the j th operator fits the i th reference, and $f_{ij} = -1$ if not. Then $\underset{\sim}{f}_1 = \{1,1,\ldots,1\}$ and $-\underset{\sim}{f}_1 = \{-1,-1,\ldots,-1\}$ represent the responses of the trivial operators which always fit, and never fit, respectively.

It has been decided that K-1 operators with responses $\underset{\sim}{f}_2, \underset{\sim}{f}_3, \ldots, \underset{\sim}{f}_K$ are sufficient for a near optimal system provided they can be found. The overall responses of such scheme can therefore be represented by a KxK response matrix.

$$\left[f_{ij}\right] \quad \begin{matrix} i = 1, \ldots, K \\ j = 1, \ldots, K \end{matrix}$$

The angular separation of the various normalised class reference vectors is maximized when the sum of all possible cross correlations $(\sum_{i,j} \hat{\underset{\sim}{F}}_i \cdot \hat{\underset{\sim}{F}}_j)$ between them is minimized. In the practical solutions described in this thesis it will be rare for all angular separations θ_{ij} to satisfy $\frac{\pi}{2} < \theta_{ij} \leqslant \cos^{-1} -1/35$. Therefore in this treatment it will be considered sufficient to minimise $\sum_{i,j} (\hat{\underset{\sim}{F}}_i \cdot \hat{\underset{\sim}{F}}_j)^2$ with the

risk that solutions in which $\theta_{ij} > \pi/2$ for all i,j will be overlooked.

First it is shown that the sum (C) of the squares of the cross correlations between each of the columns of $[f_{ij}]$ is equal to the sum of the squares of the cross correlations between each of the rows.

$$C = \sum_{i,j} \left\{ \sum_{k} f_{ki} f_{kj} \right\}^2$$

$$= \sum_{i,j,k,m} \left\{ f_{ki} f_{kj} f_{mi} f_{mj} \right\}$$

$$= \sum_{i,j,k,m} \left\{ f_{ki} f_{mi} f_{kj} f_{mj} \right\}$$

Interchanging dummy variables i,k and j,m gives

$$C = \sum_{i,j,k,m} \left\{ f_{ik} f_{jk} f_{im} f_{jm} \right\}$$

$$= \sum_{i,j} \left\{ \sum_{k} f_{ik} f_{jk} \right\}^2$$

Therefore minimizing the (cross correlation)^2 among the set of operator responses will almost always ensure that the cross correlation between character reference vectors is also minimized. The permanent inclusion of the trivial operator response $\underset{\sim}{f}_1$ means that the other K-1 operators are discouraged from a similar behaviour. This means that during the optimisation of the operators the cross correlation between character reference vectors is being reduced by usefully increasing the angular separation rather than by trivially reducing moduli.

These considerations provide a good evaluation measure for each operator. A candidate operator should receive a good score if its response is only slightly correlated with the responses of the operators already generated. In order to construct an increasing score which can be formulated in the fast integer arithmetic of a computer the following formula has been chosen.

$$E(\underset{\sim}{f}_0) = K\underset{\sim}{f}_0^2 - \sum_{i=1}^{K} \left(\frac{(\underset{\sim}{f}_0 \cdot \underset{\sim}{f}_i)^2}{\underset{\sim}{f}_i^2} \right)$$

where $\underset{\sim}{f}_0$ is the response of the candidate operator. Here $E(\underset{\sim}{f}_0)$ increases as the (correlation)^2 of $\underset{\sim}{f}_0$ with the $\underset{\sim}{f}_i$ decreases. This formula can be interpreted geometrically as the sum of the perpendicular distances from a point in K-space represented by $\underset{\sim}{f}_0$ onto the directions of each $\underset{\sim}{f}_i$. The new operator is included in the set if there is room, or if its performance is better than the worst member.

It is observed that this scoring scheme embraces the information measure of Kamentsky et al. (61) and Liu (62); the requirement for $\underset{\sim}{f}_0$ to be minimally correlated with $\underset{\sim}{f}_1$ necessarily implies that $\underset{\sim}{f}_0$ must partition the character classes into two halves.

4.3.3 Statement of the Problem and Its Approach

The OCR experimental work in this thesis was concerned with the discovery of a set of position invariant operators which lead to the recognition of poor quality OCR B binary alphanumeric characters*. The patterns were presented on a 40x20 array, although in fact each individual character could normally be placed entirely within a 30x16 array.

It was decided at the outset that the operators should be of the matrix matching type and should be restricted in the following ways:

a) Size should not be greater than 30x16.

b) Elements should be either white-seeking(ø), black-seeking (1) or don't cares.

c) Number of 0 and 1 elements should be bounded.

All three limitations were chosen so that results would be compatible with current equipments.

Reference characters were chosen to be the three thicknesses of the theoretical archetypes for OCR B alphanumerics. Operator responses were obtained from this set of 108 characters.

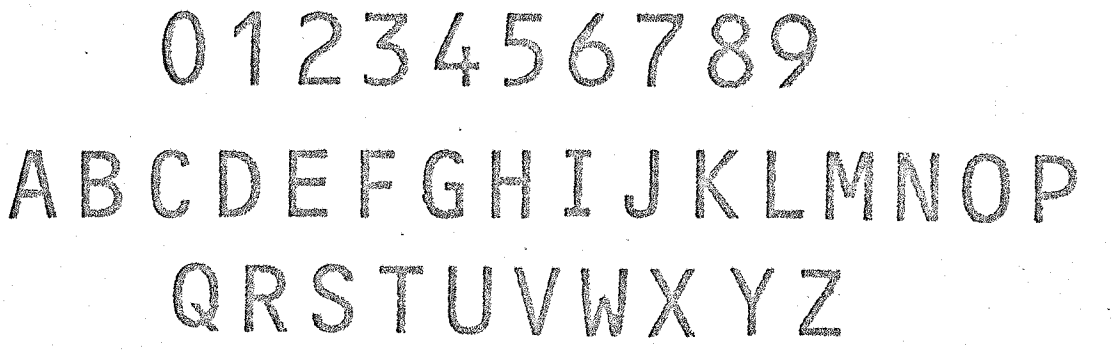


FIGURE 8 : The OCR B Upper Case Font

* OCR B is a stylized font specially designed to satisfy requirements of both human and machine readers (Figure 8) (68).

Having chosen a suitable evaluation function, the search for good operators was carried out using an evolutionary procedure. Initially a random operator (ζ) was generated having a bounded number of elements, F .

$$\zeta = \left\{ \underset{\sim}{x}_1, p_1; \underset{\sim}{x}_2, p_2; \ldots; \underset{\sim}{x}_F, p_F \right\} \qquad F \leq F_o$$

where $\underset{\sim}{x}_i = (x_{i_1}, x_{i_2})$ are the co-ordinates of the i th element and $p_i = 0$ or 1 is the parity of the i the element. The operator was evaluated and the score retained. A single random change was then made to an $\underset{\sim}{x}_i$ or a p_i. If the performance did not improve, the alteration was removed, otherwise it was left in place. After a succession of such steps, useful structure was 'trained' into the operator. After M_o (~50) successive alterations had brought no improvement the evolution was terminated and the operator checked for redundant elements. Elements were removed if their omission either had no effect on the score, or caused an improvement. The operator was then evaluated for inclusion in the set (library) of retained operators. It was discarded and the process repeated, if there was no room in the library and the operator was no better than any of the operators already present.

The next section describes the course of experimental work which was carried out using the evolutionary procedure as a tool in the search for useful operators.

4.4 Experimentation and Results

4.4.1. 10x10 Single Class Operators

The most natural and direct approach was initially thought to be the identification of features which characterized each individual class of pattern. It was also felt worthwhile to explore

the possibility of using small 10x10 operators because these might be 'easier to find' than larger versions.

The scoring scheme employed was based on the following formula

$$E = 36g_{\alpha} - \sum_{i=1}^{36} g_i \quad (g_i = 0,1,2, \text{ or } 3).$$

where g_i is the number of matched archetypes in the i th class and α is the class which the operator is expected to identify.

Operators were given a fixed number of random elements and allowed to evolve in the manner explained in the last section(4.3.3). In the very first experiments it was discovered that operators possessing 12 or more elements did not evolve. Their initial structure was so unrelated to the characters that matches rarely occurred even after a considerable number of single alterations has been tried. In this way, it was decided that operators should be given 11 elements. It was also found very early in the work that once the evolution has progressed a small distance, changes to the parity of single operator elements rarely caused an improvement. As soon as the operator took up a particular structure, the parity change almost invariably was too traumatic an alteration to allow the operator to continue functioning. The gains provided by the added 'freedom' of the parity change, were not therefore considered sufficient to justify the increased length of time between operator improvements, and so it was abandoned.

Although operators all started with 11 elements, the software allowed changes to be made which placed elements on 'top of each other', thereby effectively reducing their number. At the end of the evolution of an operator, redundant elements were removed enabling the useful size of the operator to be seen. A total of 489 operators were evolved according to the single class evaluation formula with α ranging through

the classes A-Q. The proportions of operators possessing various numbers of elements are illustrated in table 4.

No. Elements	5	6	7	8	9	10	11
No. Operators	1	17	58	176	154	59	24

TABLE 4

It is seen that operators possessing 8 elements are favoured for the task of matching one class out of 36 classes. It becomes increasingly difficult to construct good operators with fewer than 8 elements because the discrimination must rely much more upon ingenious spatial relationships between the elements in order to achieve the match/no match ratio of 1/36. As far as evolution was concerned, it was important to initialize operators with one or two more elements than the optimum number. The presence of one or two redundant elements gave the search a measure of flexibility and enabled the operator to be modified without damaging its basic structure. (See also 4.4.3).

The composition of the 8 element operators is illustrated in Table 5 where it is seen that equal numbers of black and white seeking elements are preferred.

	Bl. | Wt.	Bl. | Wt.	Bl. | Wt.	Bl. | Wt.	Bl. | Wt.	Bl. | Wt.
No. Elements	7 - 1	6 - 2	5 - 3	4 - 4	3 - 5	2 - 6
No. Operators	1	12	69	82	12	0

TABLE 5

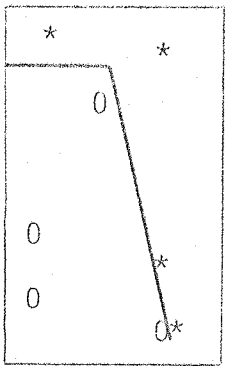
There seems to be some justification, again in the interests of program efficiency, not to generate operators possessing 1 or fewer elements of either parity. A large number of black seeking elements probably does not evolve successfully, whereas a small number of black seeking elements will not extract enough information from the characters.

Each operator was scanned over all 108 characters in about 5 seconds and a new operator was produced after about 100-200 iterations or 10-20 minutes of processing time. Many of the results described in this thesis were obtained during night periods when the computer would not otherwise have been used. Details of the software relevant to this work are given in Appendix D.

Four examples are shown in Figure 9. Alongside each operator is a histogram of the responses over all 108 archetype OCR B characters. The particular aspects highlighted by the operators have beem superimposed on each illustration. These are instances of features which occur only in single classes of characters. It is noted that especial use is made of black/white adjacent elements or edge detectors, factors which were specifically excluded by Uhr and Vossler (see 4.2.).

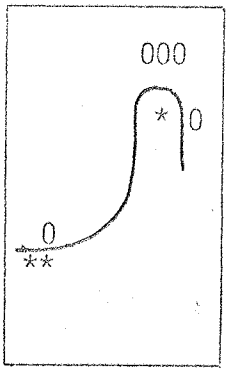
An analysis of the operator responses enabled the characters A-Q. to be ordered according to the percentage of operators scoring $\geqslant 104$. This ordering is shown in Table 6.

a)



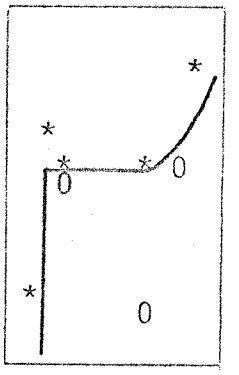
```
      *
      *
      *
0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ
```

b)



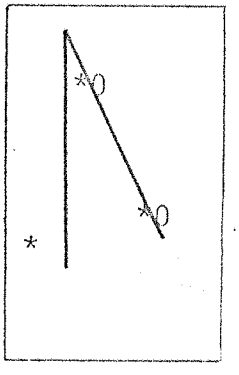
```
      *
      *
      *
0123456789ABCDEFGHIJKLMNOPQRSTUVWYXZ
```

c)



```
                  *
                  *
                  *
0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ
```

d)



```
            *
            *
            *
0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ
```

FIGURE 9 : 10 x 10 single class operators and responses

Character	Total No. Operators	No. scoring ⩾104	%
H	32	32	100
K	32	32	100
I	32	29	91
A	28	23	82
E	32	26	81
J	32	25	78
F	32	23	72
N	32	23	72
P	32	21	66
G	32	20	63
M	19	12	63
Q	10	6	60
O	32	19	59
D	32	18	56
C	27	14	52
B	22	8	36
L	32	6	19

TABLE 6

Several of the easily confused characters (Q,O,D) are seen to be quite low in the list. However, the characters 'L' and 'B'were probably not at the bottom because of any intrinsic difficulty of recognition, but because most of the distinguishing features were too large to be represented within a 10 x 10 window.

Several of the best operators corresponding to each class were grouped together so that some real character examples could be classified using a majority decision. Some preliminary results indicated that the operators would only classify characters whose quality was equivalent to that of the archetype.

Single class operators are probably best designed intuitively because the human designer can build his own 'experience' into the operator without having to test it out on all possible character degradations. However, intuitive heuristics can be misleading and a human designer would certainly have difficulties if he was confined to a 10 x 10 window. Nevertheless single class operators only distinguish one class of character from the rest and are bound to be inefficient because of their poor information content (see summary to 4.2). This is illustrated in Table 7 in which ideal operators designed for three different tasks are rated according to the number of pairs of classes that are distinguished and the total number of classes in the problem.

Total No. Classes	2	10	36	100	K
Specific separation of 2 classes	1	1	1	1	1
Single class recognition	1	9	35	99	K-1
Binary partition of classes	1	25	324	2500	$(K/2)^2$

TABLE 7 : Numbers of pairs of classes distinguished by 3 types of operator

In a 36 class problem it is seen that operators which match half of the character classes can be almost ten times as powerful as those

which match only single classes. Moreover this advantage increases as the number of classes increases*. However, the problem of designing a whole set of operators with the necessary interdependencies is a task far beyond human intuition.

In the next section operators are evolved not to fit single classes, but to satisfy the criterion of angular separation set out in 4.3.2.

4.4.2 10 x 10 Orthogonal Operators**

10 x 10 operators were again evolved but this time the following evaluation function was used:

$$E(\underset{\sim}{f}_o) = 33\underset{\sim}{f}_o^{\,2} - \sum_{i=1}^{33} \frac{(\underset{\sim}{f}_o \cdot \underset{\sim}{f}_i)^2}{\underset{\sim}{f}_i^{\,2}}$$

as derived in section 4.3.2. In this practical application each $\underset{\sim}{f}_i = \{f_{1i}, f_{2i}, \ldots, f_{36i}\}$ was a position normalized version of the true response vector $\underset{\sim}{g}_i = \{g_{1i}, g_{2i}, \ldots, g_{36i}\}$ over the 108 archetype character set, where

$$f_{ij} = g_{ij} - 3/2 \quad (g_{ij} = 0, 1, 2, \text{or } 3)$$

This transformation was carried out in order to allow angles to be assessed outside the positive 'quadrant' and to avoid zero denominators in the evaluation formula. Initially the library was empty and the stored responses were set to zero. This meant that at first,

---

* This implies that the number of operators also increases (See 4.3.1).

** The term orthogonal is used because ideally there is a right angle between any pair of response vectors.

$f_{ij} = -3/2$, $i \neq 1$ with $f_{1j} = +3/2$ (as defined earlier in 4.3.2.).
It followed that the first operator to be generated was very strongly encouraged to respond at right angles to the trivial response $\underset{\sim}{f}_1$. The first operator was satisfactory therefore if it merely matched half of the character classes and was stored with response $\underset{\sim}{f}_2$. The second operator was still encouraged to match half of character classes, but at the same time its response had to be perpendicular to $\underset{\sim}{f}_2$. As the library filled, more $\underset{\sim}{f}_i$ became non-trivial and the requirement to be perpendicular to $\underset{\sim}{f}_1$ decreased. Eventually a library of 32 orthogonal operators with responses $\underset{\sim}{f}_2, \underset{\sim}{f}_3, \ldots, \underset{\sim}{f}_{33}$ was produced. However, there were always several very poor operators in the library which added very little information to the recognition capability and had a correspondingly low score. And so the evolutionary process was continued allowing the weakest operators in the library to be replaced by better ones as and when they were generated. In this way a steadily improving orthogonal set of operators was obtained. Continuing the process in this manner meant also that the search was not impeded by a poor choice for the first few operators; indeed any operator was discarded as soon as its response did not appear to be contributing to the 'overall' performance.

Over several experiments a total of 76 operators were generated all initially possessing 11 elements and all having this number possibly reduced by the elimination of redundancy. The distribution of operators possessing various numbers of elements is illustrated in Table 8.

No. Elements	5	6	7	8	9	10	11
No. Operators	6	32	26	8	2	1	1

TABLE 8

This time the most popular number of elements was 6 (cf. Liu who chooses 5-7 in (62)) as compared with 8 for the single class operators. This reduction occurs because the orthogonal operators are expected to fit 18 classes of character instead of one. It follows therefore that in general, orthogonal operators not only extract more information, but also they have a simpler internal structure than the single class operators.

The composition of the 6-element operators is displayed in Table 9 where again an equal split in the elements is favoured.

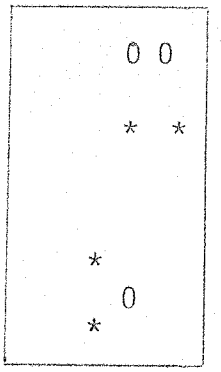
	Black/White	Black/White	Black/White	Black/White
No. elements	5 - 1	4 - 2	3 - 3	2 - 4
No. operators	1	10	19	2

TABLE 9

Some examples of orthogonal operators produced by the procedure are shown in Figure 10. Often the intention of the operator could be seen by inspection. For instance, operator a) detects top left hand corners, and operater d) detects a specific type of concavity. The author feels that although it is easy to interpret operators in this way the reverse process of obtaining an operator from a response requirement is extremely difficult and may not even be possible.

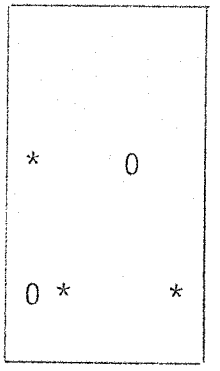
An orthogonal set of 32 operators was evolved and evaluated on some real OCR B alphanumeric data. The decision scheme was based upon the cross correlation of the response vectors with 36 reference vectors derived from the archetypes (4.3.1). The data was obtained from line printer outputs and was separated subjectively into 6 grades of print quality. A detailed account of the grading is not given here, but it is sufficient

a)



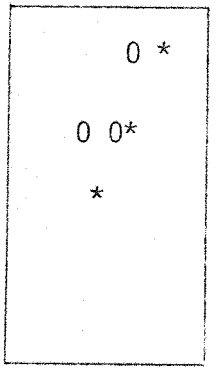
```
*     *   **********        ******
*     *   **********        ******
**    *   **********        ******
0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ
```

b)



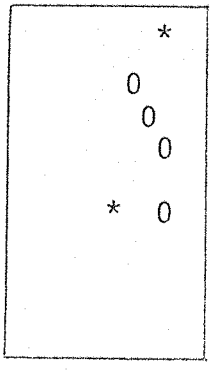
```
*       * **  *   *  *     * * * *****
*       * **  *   *  *  *** * * *****
*       * **  *   *  *  *** * * *****
0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ
```

c)



```
 *** ***** * *       *    ** ** *
**** ***** * *       *    ** ** *
**** ***** ***       *    ** ** *
0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ
```

d)



```
**   *  * *** ** * ** * ***
**   *  * ****** * ** * ***
*** * ** ****** * ** * ***
0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ
```

FIGURE 10 : 10 x 10 Orthogonal Operators

to state that the quality was much higher than the data used in the next section, where in fact better error rates were achieved. The results are summarized in Table 10

	Total No. Characters	No. Correct	% Error
Grades 1 & 2	524	484	8
Grades 3 & 4	636	449	35
Grades 5 & 6	223	126	44

TABLE 10

The error rates were obtained for a forced decision with no option to reject the character. It was observed that particular character confusions could be blamed for a large number of the errors. Among the worst offenders were Ø - O, E - F and B - P. Most of the difficulty could be attributed to the fact that distinguishing features for these character pairs all involved a dimension which was greater than 10 units long (Figure 11). This meant that a 'P' contained all the 10x10 features that a 'B' contained, and similarly an 'F' contained all the features in an 'E'. A 'bottom left hand corner' detector would in theory make the required distinctions. However, it was found that

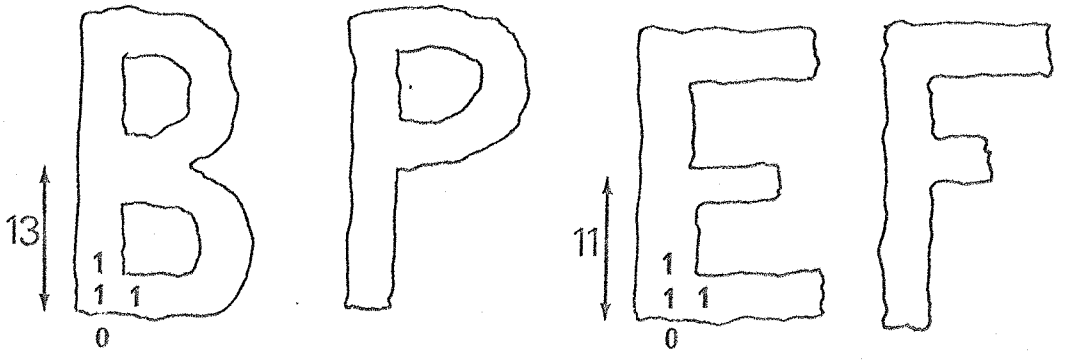


FIGURE 11 Two Confusion Pairs

varying character thicknesses and edge irregularities made this intuitive operator of very little value. In any case, such an operator designed for a specific confusion pair is not in keeping with the strategy of orthogonal operators.

In view of these difficulties it was decided to allow an operator to span an entire character. This would of course increase the size of the search space, but at the same time it would remove severe restrictions in the potential recognition capability of the system.

4.4.3. 30 x 16 Orthogonal Operators

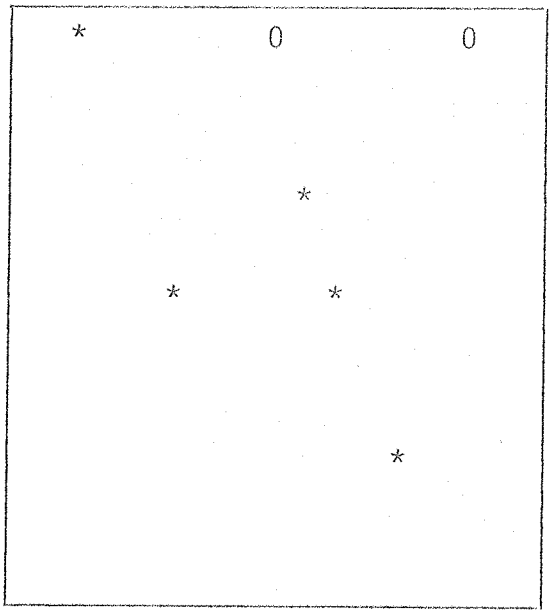
In this experiment operators were allowed to evolve within a 30 x 16 frame. Earlier empirical results suggested the following bounds for the generation of operators:

a) All operators were initialized with 7 elements.

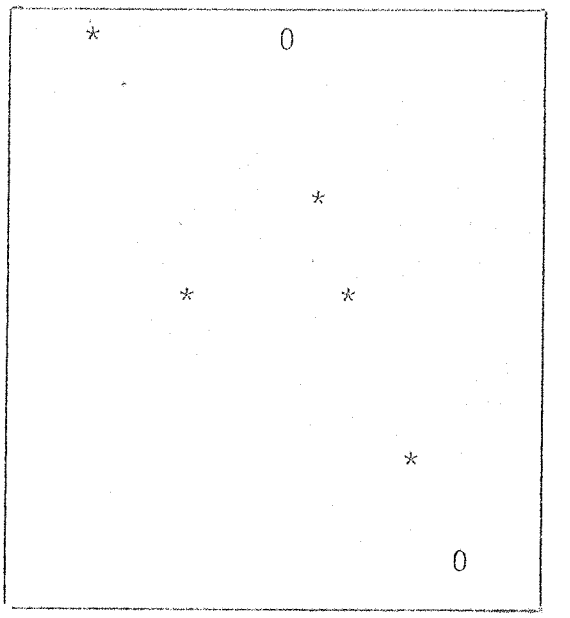
b) No operator was generated with less than 2 black seeking elements.

The orthogonal operator evaluation function was again employed (4.3.2), this time with K = 65. The sequence of evolutionary changes during the production of a typical operator is illustrated in Figures 12 and 13. Initially the operator has five black seeking points and two white seeking points located in random positions within the 30 x 16 frame. Amongst a total of 41 random alterations there were 11 improvements. The successive response vectors are shown in Figure 14. Although the evolution is incomplete, it is still possible to identify the final operator (No.12) with the detection of 'bottom right hand corners'.

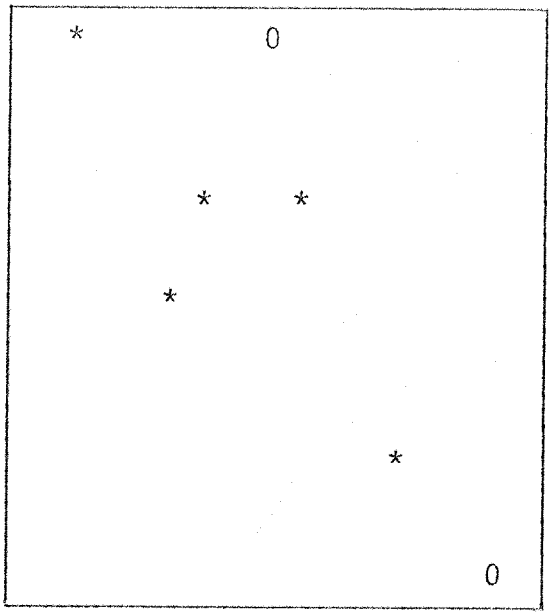
It was thought that in practice operator responses would not all be mutually orthogonal and therefore that many more operators than the optimal 35 could be accommodated in the recognition system. It was found



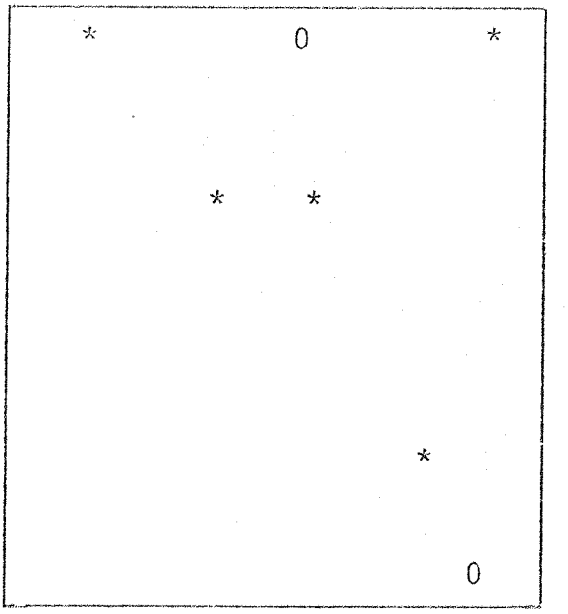
1.



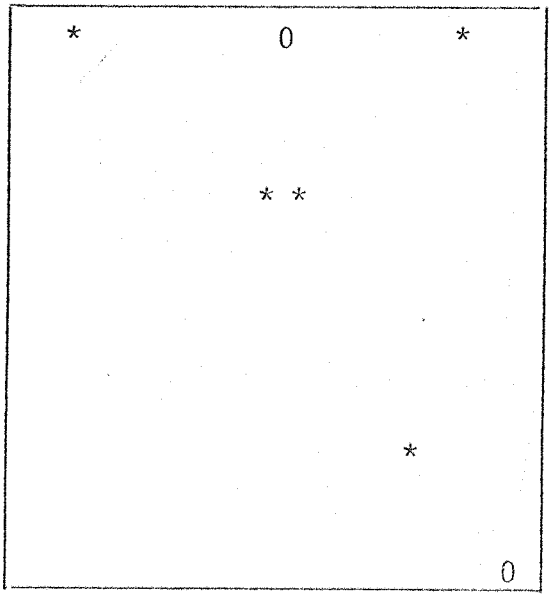
2.



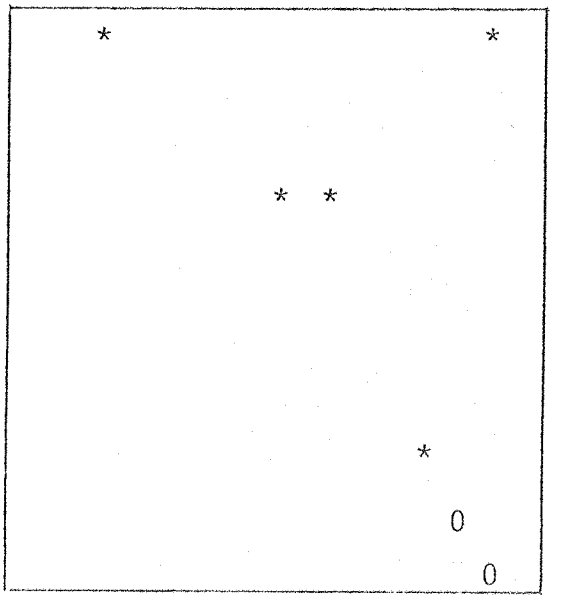
3.



4.

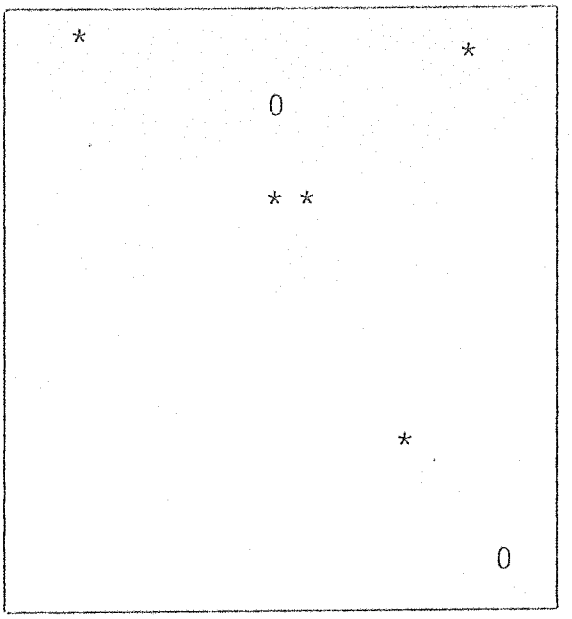


5.

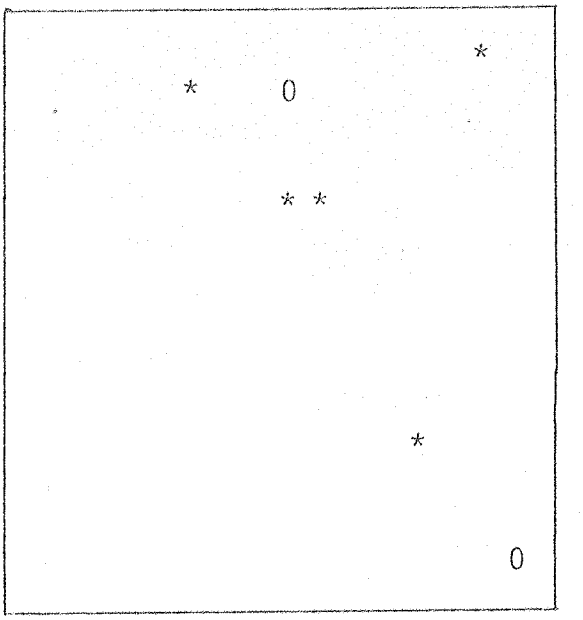


6.

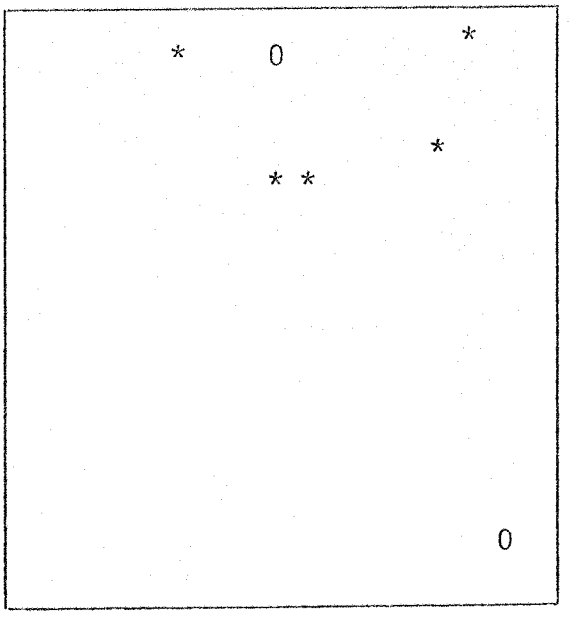
FIGURE 12



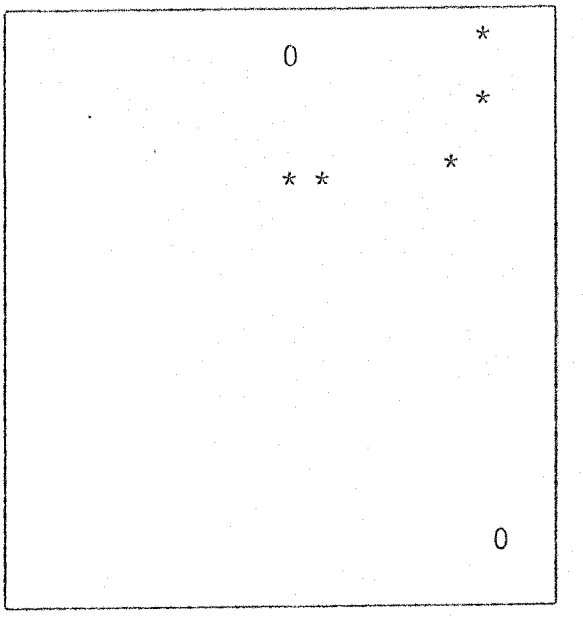
7.



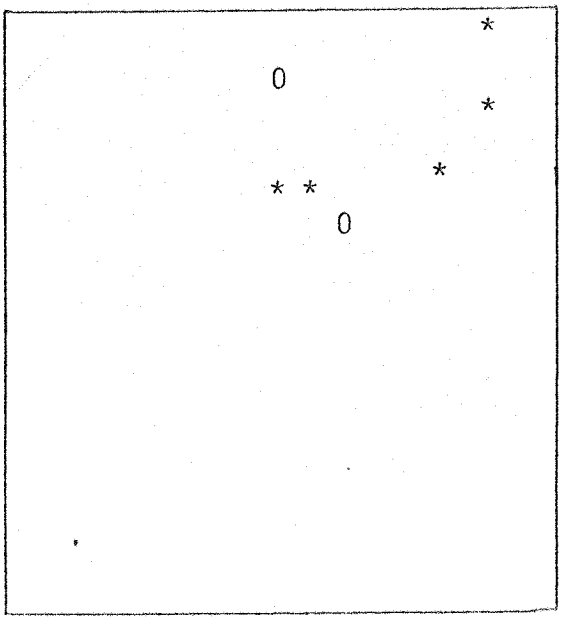
8.



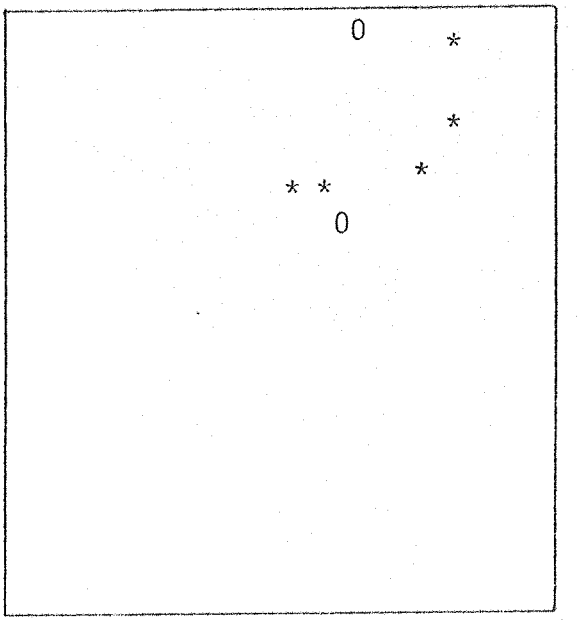
9.



10.



11.



12.

FIGURE 13

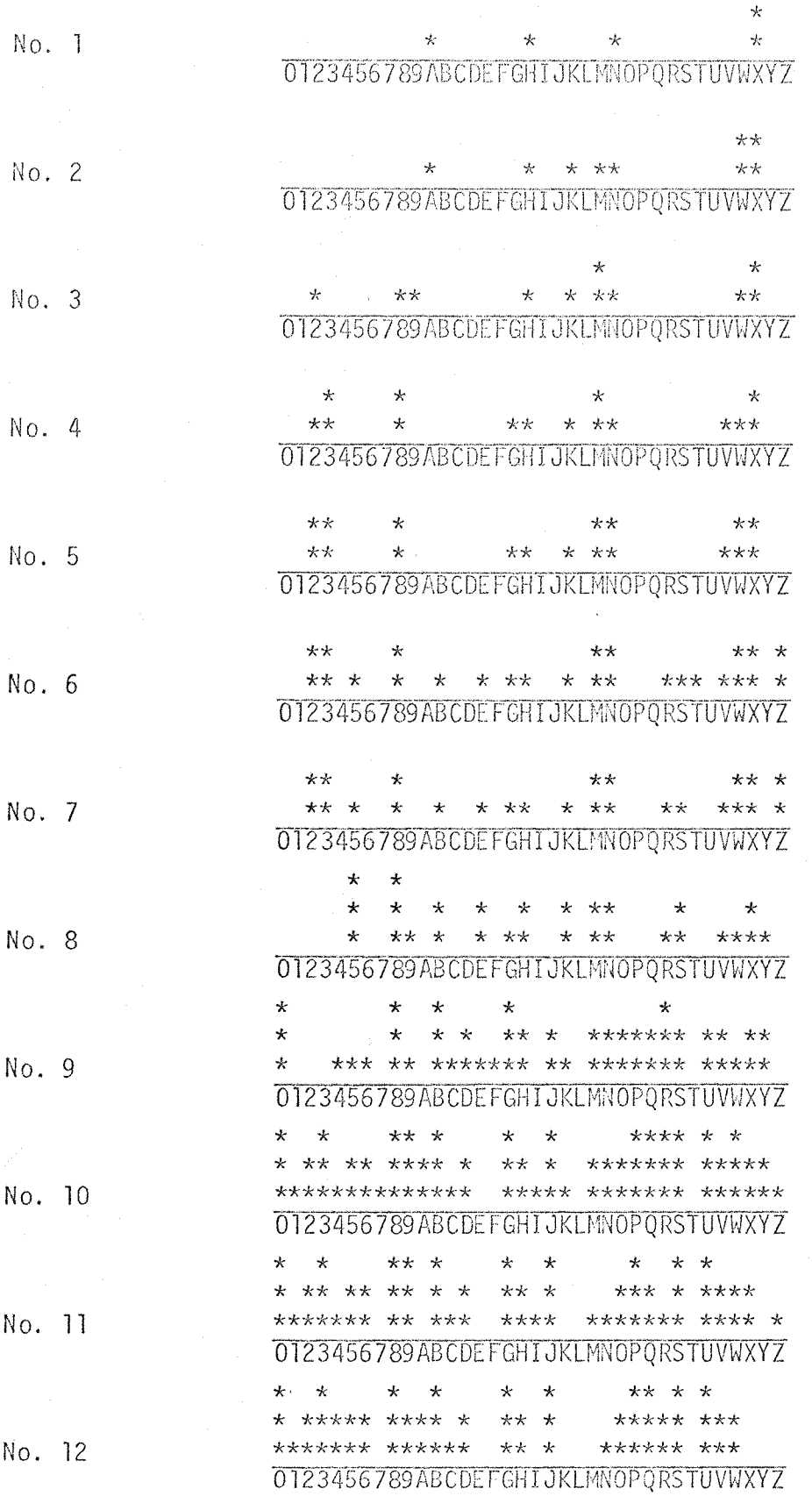

No. 1
No. 2
No. 3
No. 4
No. 5
No. 6
No. 7
No. 8
No. 9
No. 10
No. 11
No. 12
0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ

FIGURE 14

that the rate at which new operators were added to the library dropped considerably after about 45 operators had already been generated. The procedure 'preferred' to overwrite poor operators with better ones rather than increase the total number. This meant that as the number of operators in the library was increased, certain operators became weaker and at least one began to score less than the trivial operator. Operators weakened in this way simply because their responses were surrounded by several others which were not at a satisfactory angular separation. Here then is some experimental validation for the theory expounded by Flores et al. in 4.3.1 where it was stated that no improvement could be made to a recognition system by merely increasing the number of features without limit.

Another set of orthogonal operators was now generated but this time it was limited in size to 40 operators in order to avoid the stagnation phenomenon described above. The set of operators was evaluated over real data using a simple nearest neighbour decision scheme:

Let g_{ij} be the number (0, 1, 2, or 3) of archetype characters matched by the j the operator in the set. Suppose the test character T gave rise to the response vector $F_T = t_1, t_2, \ldots, t_K$ where K is the number of operators and $t_j = 3$ or 0 for a fit or no fit, respectively. Then T was assigned to the class corresponding to the smallest of the 36 sums S_i, where

$$S_i = \sum_{j=1}^{K} |g_{ij} - t_j| \qquad i = 1, 2, \ldots, 36.$$

If, however, the two smallest sums S_a, S_b, differed by less than or equal to 4, the character was rejected and no classification was given.

The data used in this section were 2864 OCR B alphanumeric line printer characters. The quality of print ranged from the totally unrecognizable to the very highest quality. The characters were subjectively graded by document and so it was to be expected that many individual characters were incorrectly graded. There were 32 grades numbered 1-32, the larger numbers being associated with the poorer qualities.

The odd grades corresponded to the thinner sorts of character, and the even grades corresponded to the thick characters. Table 11 lists the numbers of characters which were put into various pairs of grades. Figures 15 and 16 depict 8 character samples taken from 8 of the grades. The top row of 4 in each block of 8 are typical characters in the grade; the lower row consists of 2 of the worst followed by 2 of the best characters.

GRADES	1- 3	5- 7	9-11	13-15	17-19	21-23	25-27	29-31
No. CHARACTERS	69	160	157	238	202	319	162	114

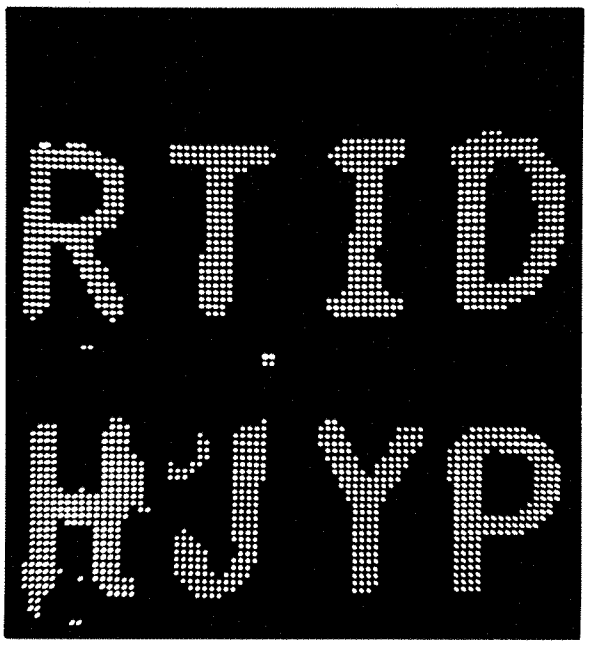
Odd Grades : Total = 1421

GRADES	2- 4	6- 8	10-12	14-16	18-20	22-24	26-28	30-32
No. CHARACTERS	81	207	225	157	138	341	221	73

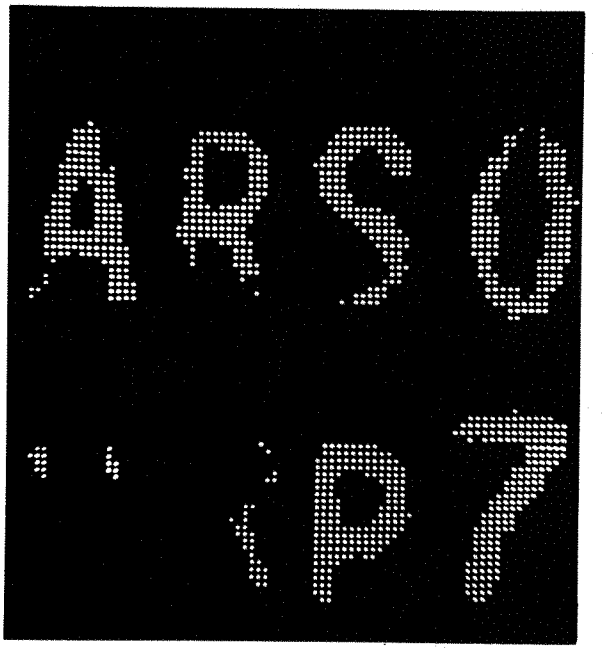
Even Grades : Total = 1443

TABLE 11

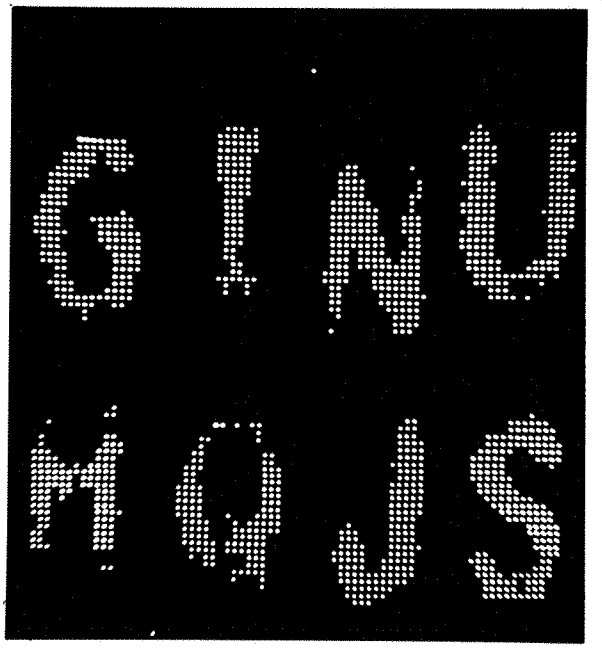
The performance of the 40 30 x 16 operators was decidedly better than that achieved previously. However, a few difficult characters



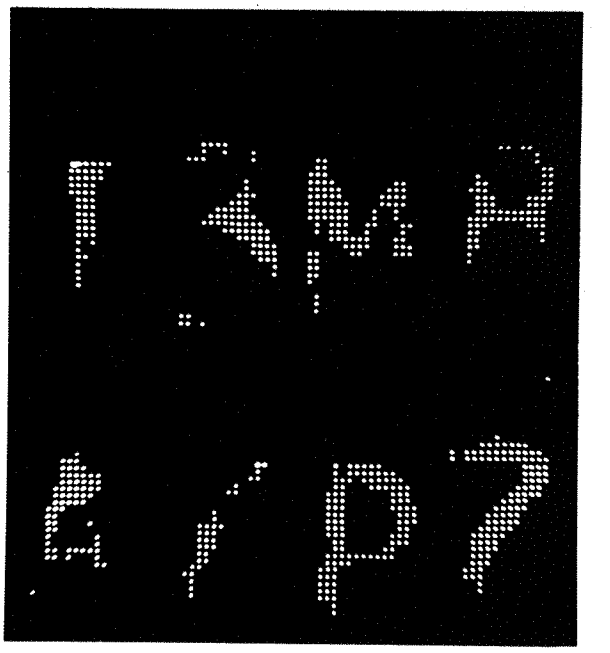
Grade 3



Grade 13

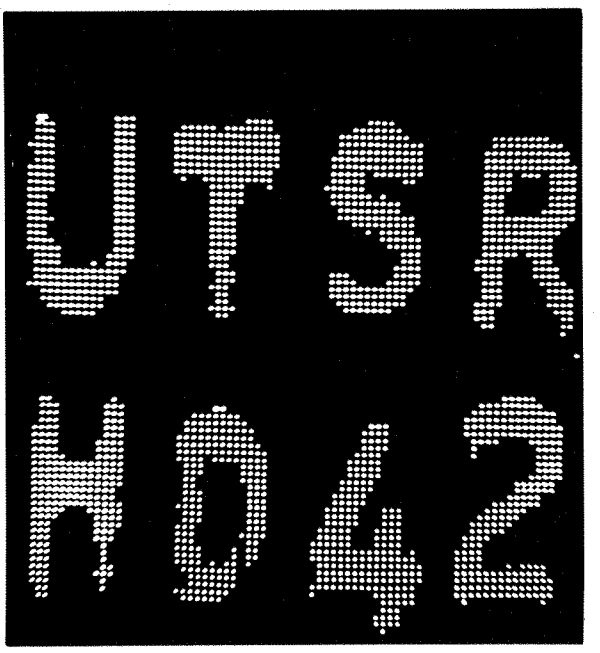


Grade 21

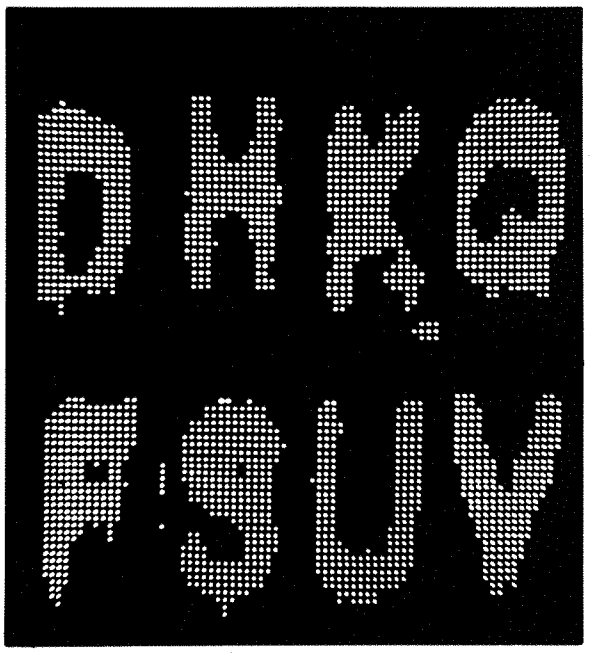


Grade 31

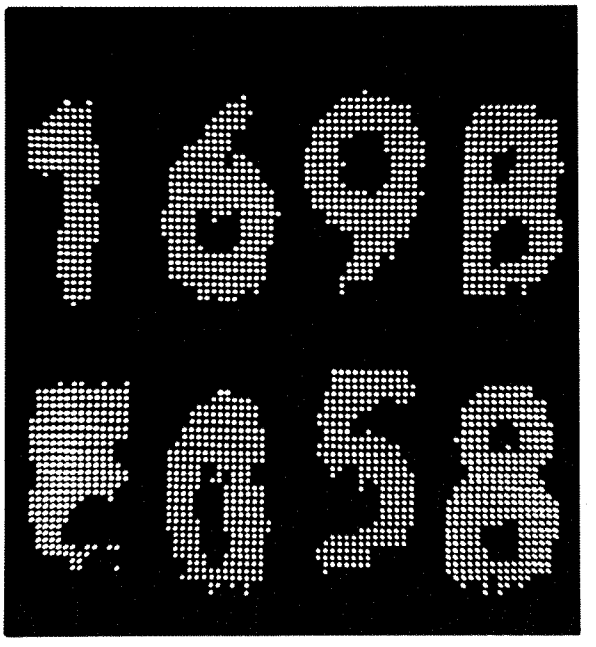
Figure 15. Typical Odd Grade Characters.



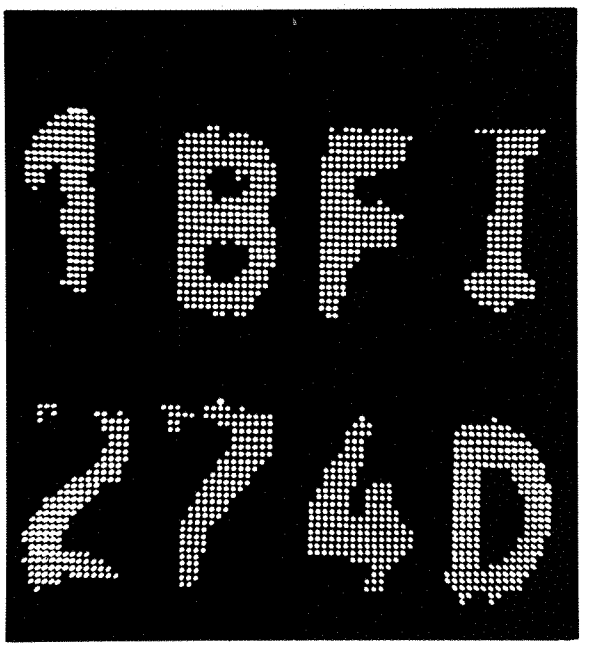
Grade 4



Grade 12



Grade 22



Grade 32

Figure 16 . Typical Even Grade Characters.

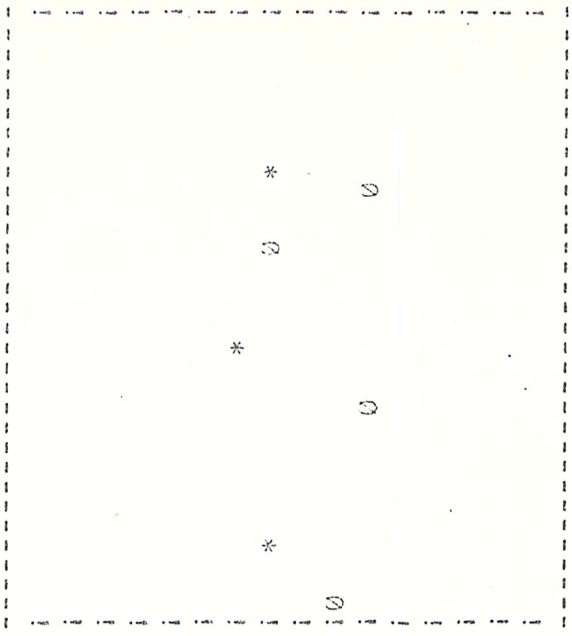
(e.g. Ø, O, D) again gave rise to the major proportion of the total error and reject rates. Two steps were taken to offset this problem.

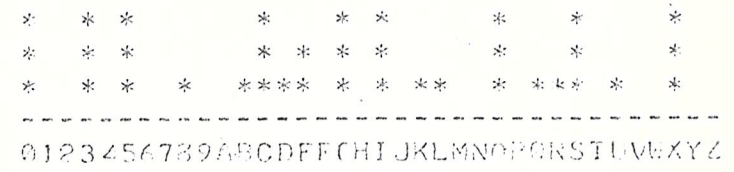
i) 24 more operators were chosen from earlier operatorsets which distinguished the more difficult confusions. This brought the total number of operators in the set up to 64.

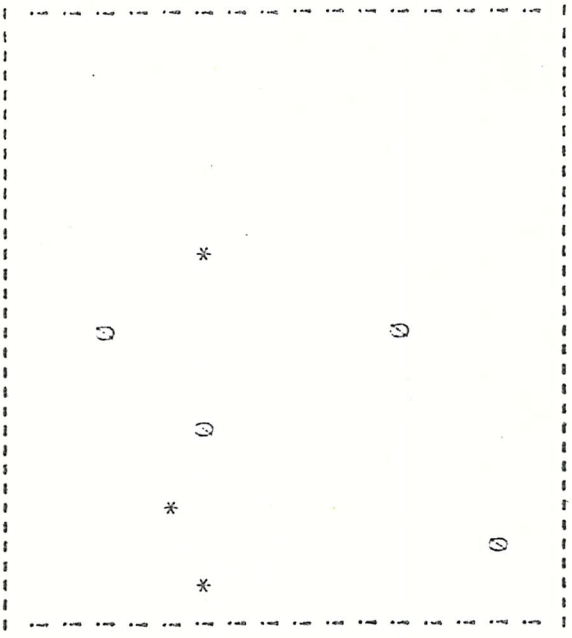
ii) The 36 character references $g_{i1}, g_{i2}, \ldots, g_{i64}$ were regenerated from the average operator responses to real data derived from grades 1 - 10.

Step i) was unsatisfactory in the sense that the extra 24 operators did not extract as much information from the characters as they might have done. However, the number of errors was reduced by this action. Step ii) effectively moved the 36 character reference vectors to the centres of gravity of the 36 clusters of data points in feature space. This was carried out because the 108 theoretical archetypes did not represent some of the real characters.

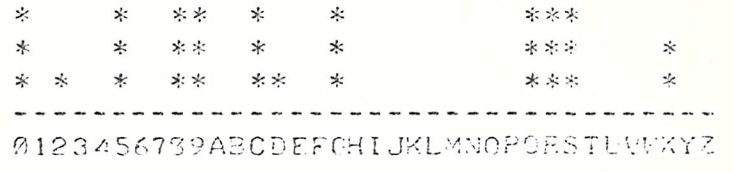
Three examples of operators together with their average responses are illustrated in Figure 17. The complete set of 64 operators is listed in Appendix E. This set of operators was evaluated over the OCR B data described above, and the error rates and reject rates are summarized in Table 12 and Figure 18.

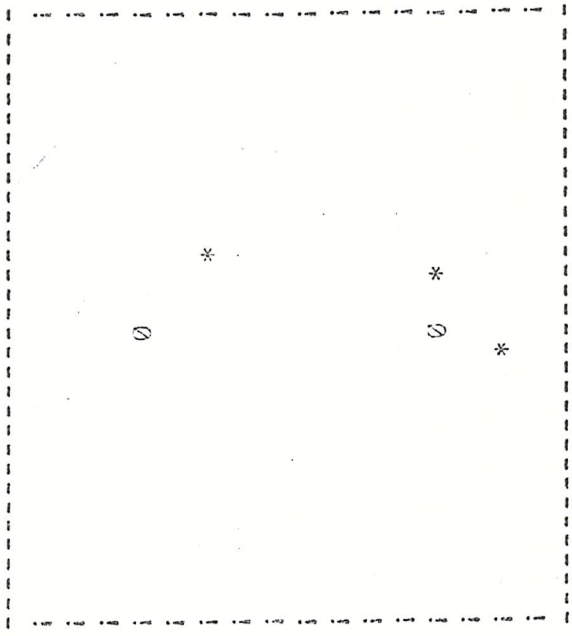






NO. 34





NO. 43

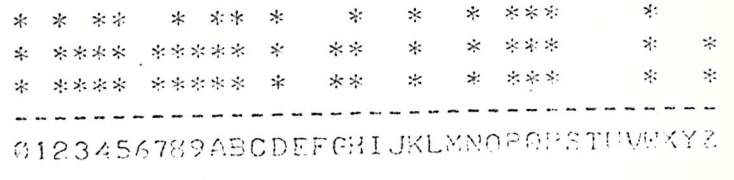
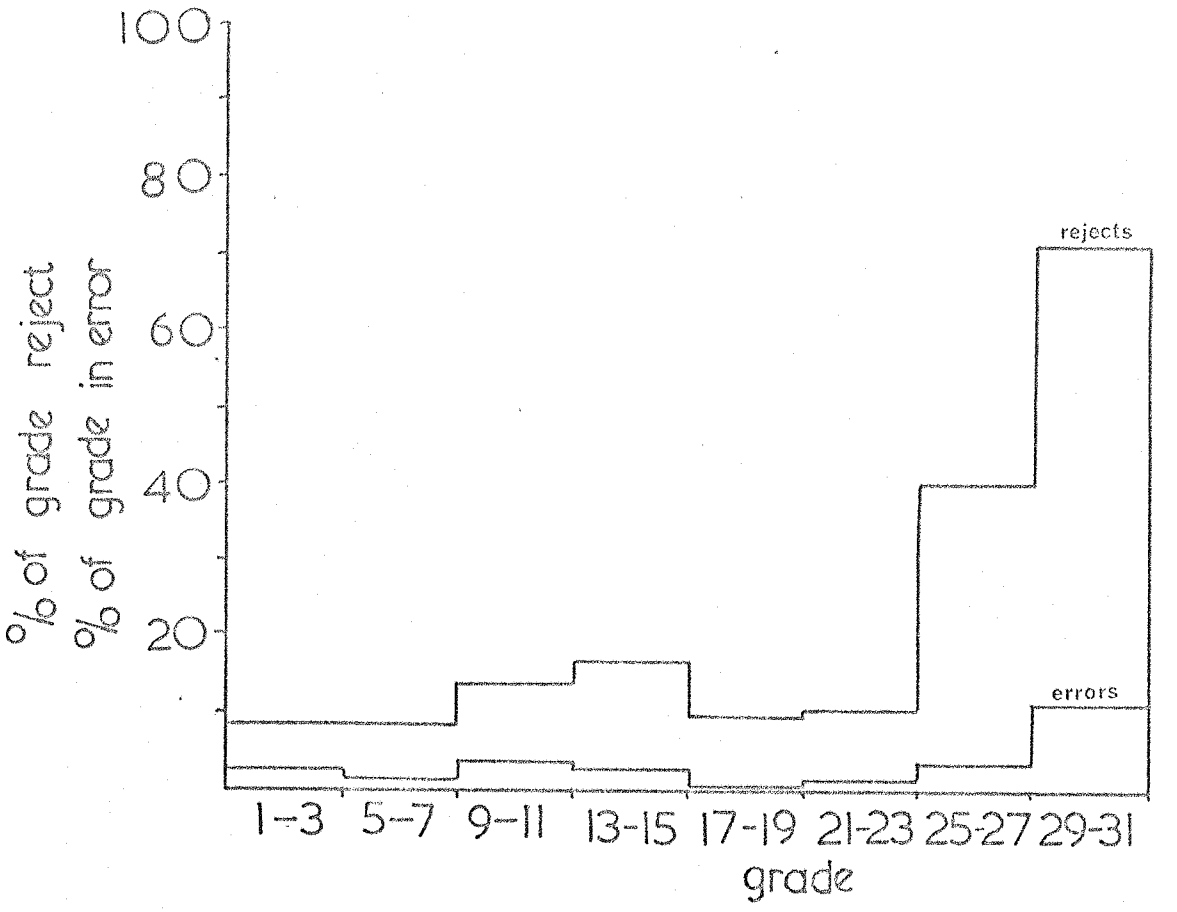


Figure 17. Three 30 x 16 orthogonal operators.



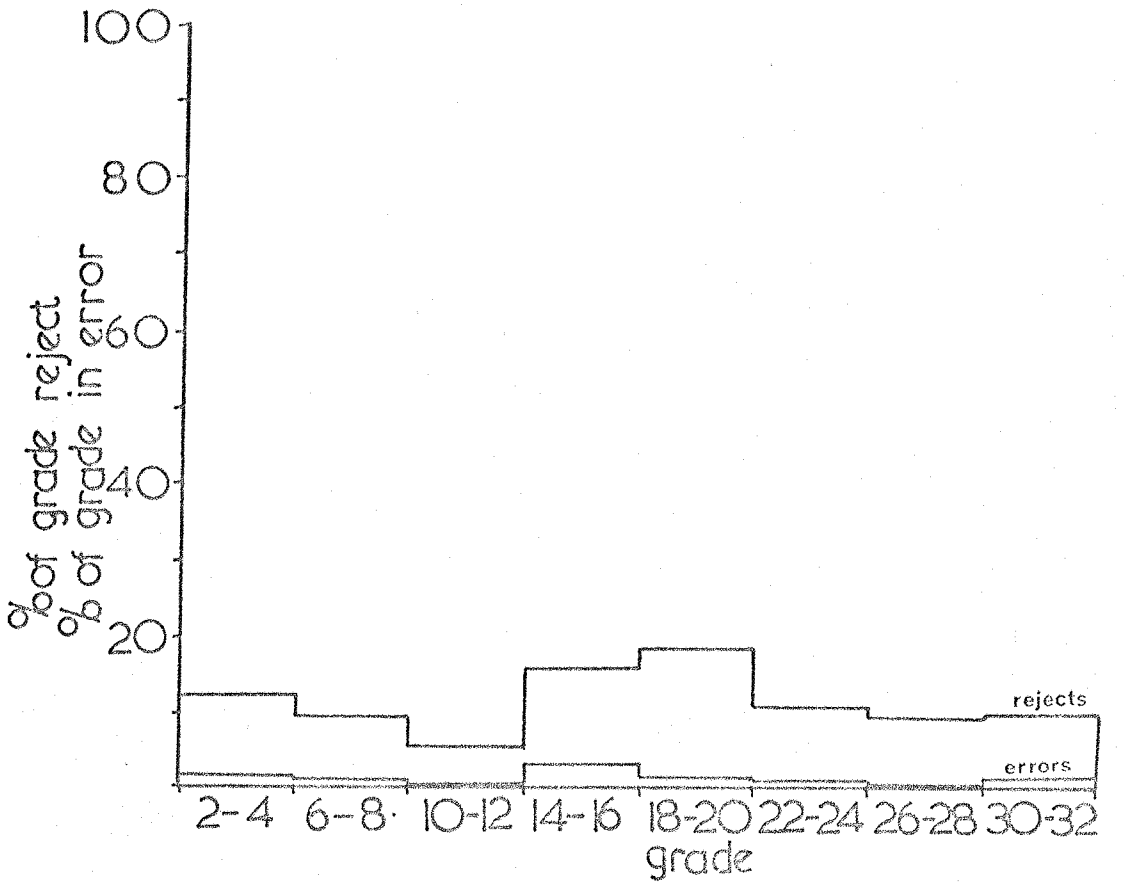


Figure 18 . Error rates and reject rates .

True Classification

Machine Classification

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
0	36	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0
1	0	76	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	4	0	0	0	0	0	0
2	0	0	52	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	65	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	54	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	62	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	52	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	2	1	0	0	0	68	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	48	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	60	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A	0	0	0	0	0	0	0	0	0	0	80	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0
B	0	0	0	0	0	0	0	0	0	0	0	56	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C	0	0	0	0	0	0	0	0	0	0	0	0	79	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
D	0	0	0	0	0	0	0	0	0	0	0	0	0	57	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
E	0	0	0	0	0	0	0	0	0	0	0	0	0	0	128	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
F	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	63	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
G	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0	0	37	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
H	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	81	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FIGURE 19A : Confusion matrix (continued in Figure 19B)

True Classification

Machine Classification

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
I	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	99	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
J	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	47	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0
K	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	58	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
L	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	85	0	0	0	0	0	0	0	0	0	0	0	0	0	0
M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	52	0	0	0	0	0	0	0	0	0	0	0	0	0
N	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	87	0	0	0	0	0	0	0	0	0	0	0	0
O	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	72	0	2	0	0	0	0	0	0	0	0	0
P	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	91	0	7	0	0	0	0	0	0	0	0
Q	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	32	0	0	0	0	0	0	0	0	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	95	0	0	0	0	0	0	0	0
S	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	92	0	0	0	0	0	0	0
T	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	80	0	0	0	0	0	0
U	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	57	0	0	0	0	0
V	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	40	0	0	0	0
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	39	0	0	0
X	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	53	0	0
Y	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	63	0
Z	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	36

FIGURE 19B : Confusion matrix (continued from Figure 19A)

GRADE	1- 3	5- 7	9-11	13-15	17-19	21-23	25-27	29-31
% Reject	8.7	8.7	14.0	16.8	9.9	10.4	40.1	71.1
% Error	2.9	1.9	3.8	2.9	0.5	1.3	3.7	11.4

Odd Grades

GRADE	2- 4	6- 8	10-12	14-16	18-20	22-24	26-28	30-32
% Reject	12.3	9.7	5.8	16.6	18.1	10.9	9.5	9.6
% Error	1.2	1.0	0.4	3.2	1.4	1.2	0.5	1.4

Even Grades

TABLE 12

Out of 2849* characters which were actually classified by the recognition system, 59 were substitution errors and 446 were rejected. The true classification and the machine classification of the characters displayed in Figures 15 and 16, are given in Table 13. The complete confusion matrix is displayed in Figure 19. It is seen that major confusions occur between Ø and O, and E and F. A study of the characters themselves reveals that only a small amount of distortion is required to transform Ø→O and O→Ø; it is also common for the lower limb of E's to be completely missing.

---

* This number is different to that given in table 11 because several characters were subsequently rejected owing to paper tape imperfections.

GRADE 3	R	T	I	D	H	J	Y	P
MACHINE CLASSIFICATION	R	T	I	D	¶	¶	Y	P
GRADE 13	A	R	S	Ø	Ø	R	P	7
MACHINE CLASSIFICATION	A	R	S	¶	¶	¶	P	7
GRADE 21	G	I	N	U	M	Q	J	S
MACHINE CLASSIFICATION	G	I	N	U	M	Q	J	S
GRADE 31	I	3	M	B	A	7	P	7
MACHINE CLASSIFICATION	¶	¶	¶	¶	¶	¶	P	7
GRADE 4	U	T	S	R	H	D	4	2
MACHINE CLASSIFICATION	U	T	S	R	H	D	4	2
GRADE 12	D	H	K	Q	F	S	U	V
MACHINE CLASSIFICATION	D	H	K	Q	F	8	U	V
GRADE 22	1	6	9	B	F	G	5	8
MACHINE CLASSIFICATION	1	6	9	B	¶	¶	5	8
GRADE 32	1	B	F	I	2	7	4	D
MACHINE CLASSIFICATION	1	B	F	I	2	7	4	D

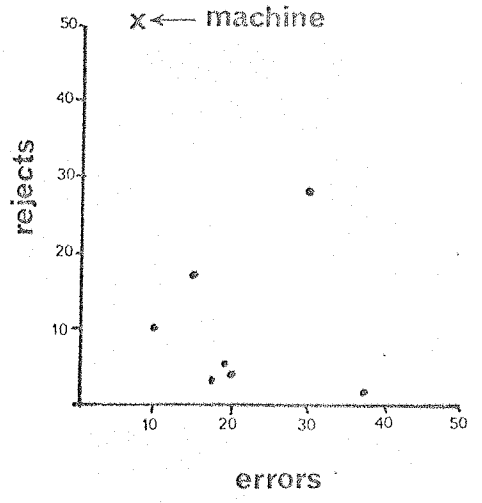
¶ = reject

TABLE 13

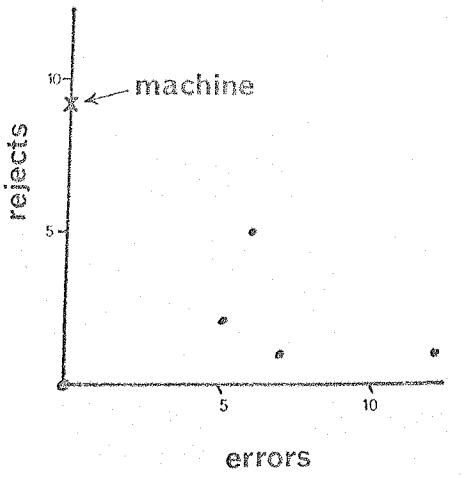
The problem of distinguishing certain difficult pairs of characters was given to a small group of human recognisers so that the machine performance could be compared. All 2's and Z's, Ø's and O's, and 8's and B's were extracted from the data and presented in a random order. The human recogniser could either decide between the two classes or reject the character. The reject rates and error rates for each person over the three problems are given in Figure 20. The machine performance was obtained by forcing a decision between the two classes in question and rejecting the character if the sums S_1, S_2, corresponding to these classes, differed by less than 4. It is seen that the humans rejected fewer characters and made more mistakes than the machine. A fairer comparison might have been made if the machine had been compelled to reduce its reject rate. Nevertheless the experiment demonstrates that some of the poor quality characters probably could never be classified with certainty.

The question of contiguous characters must arise in any O.C.R. system. So far only isolated single characters had been considered, and it was felt important that the reaction of the recognition system to contiguous characters should be tested. Of course, if character segmentation is achieved outside the recognition system, this problem will only arise during a faulty scan (4.1.2).

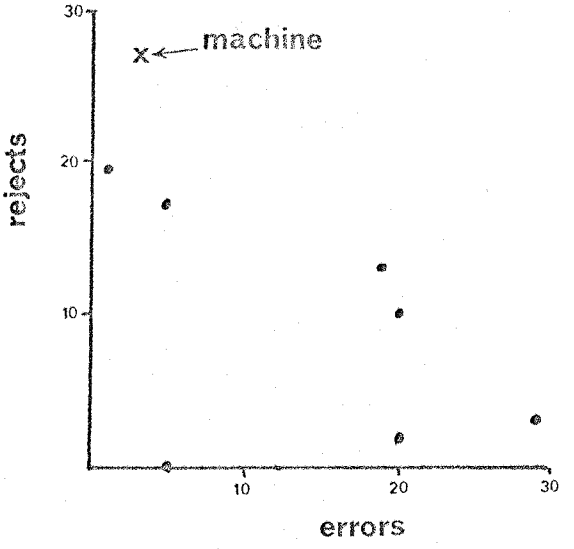
In this experiment 136 character 'contiguities' were constructed by abutting the final half of one character with the first half of another and recording the machine classifications. It was found that the machine gave false classifications to 15 of the shapes but rejected the remainder. The experiment demonstrated that the system possessed a clear ability to reject alien structures. This suggested that such a recognition scheme was feasible for inclusion in a continuous system (4.1.2) which would be



0-O confusion



Z-2 confusion



B-8 confusion

Figure 20. Human recognition performance.

independent of the horizontal separations of characters. A decision would be made only after several successive identical classifications had been given. Each character would be 'seen' by the machine to be segmented from the rest by a period of reject classifications.

4.4.4 Improvements to the Evolutionary Search

The evolutionary search has been successful in discovering operators for several different OCR problems. The minimal use of intuitive search heuristics was in keeping with the philosophy of this thesis. In hindsight, however, there were several instances where avoidable limitations had been imposed on the search.

A restrictive heuristic was the decision to keep the number of operator elements rigidly at 7 throughout the evolution of an operator. A more adaptable scheme would have allowed new elements to be created and old ones to be removed. This could have been achieved simply by providing a pool to which an old operator element could be moved, or from which a new element could be taken.

Another restriction which also hampered the search was the 30 x 16 operator frame itself. All too often the evolving operator moved up against the operator frame and was thereby prevented from any further variation in that direction. Upon reflection there was very little need to have any boundary for the evolving operator at all. Characters could always be located within a 30 x 16 matrix and therefore a useful operator could never have its black seeking elements extending across a matrix larger than 30 x 16. Some white seeking elements could probably extend over a greater area than 30 x 16, but it is likely that they would be redundant unless they were also in the vicinity of the character. It follows therefore that an unbounded evolving operator will automatically maintain

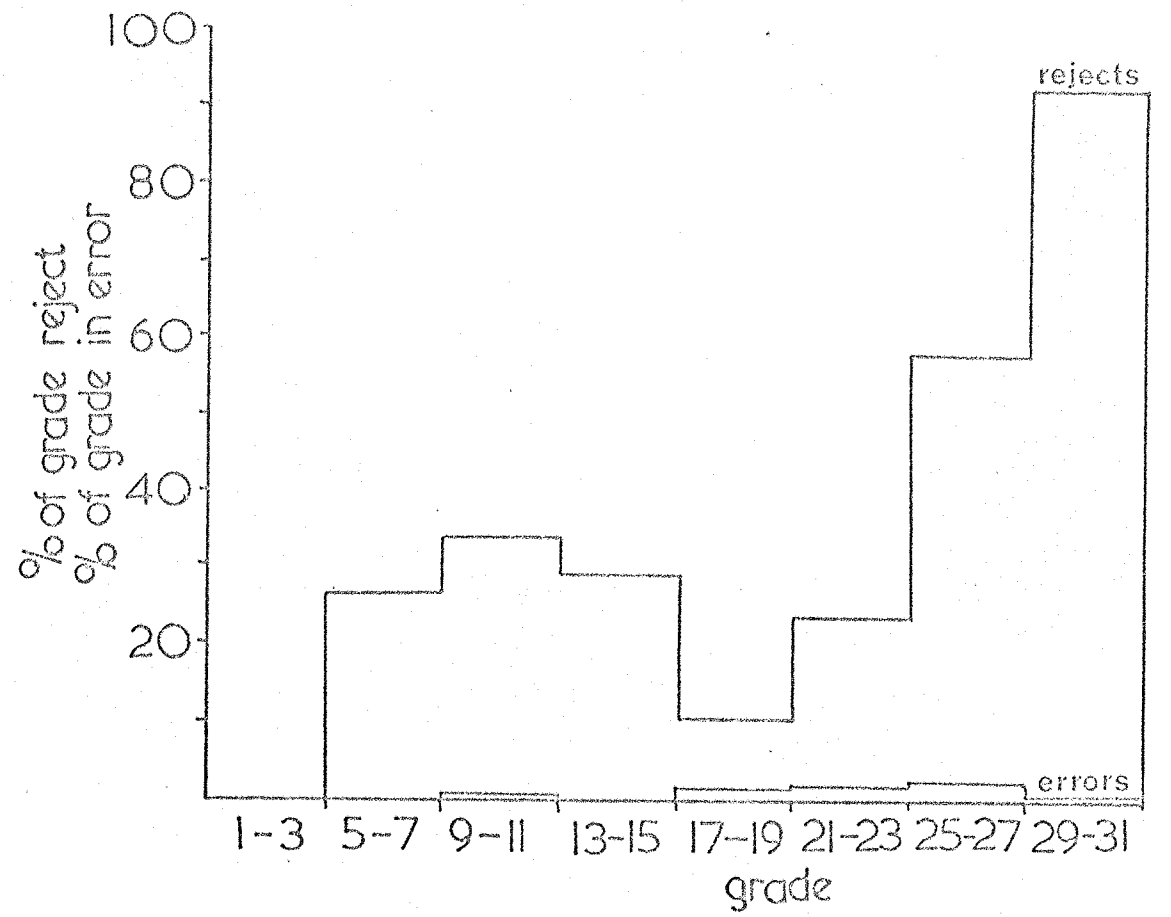
itself within the approximate dimensions of the characters being recognised. Only occasionally will the final operator have to be rejected on the grounds that its white seeking elements lie outside the terms of the problem. Such an arrangement could be simulated by employing a larger frame, say 30 x 20, and centralizing the operator within the frame after each improvement.

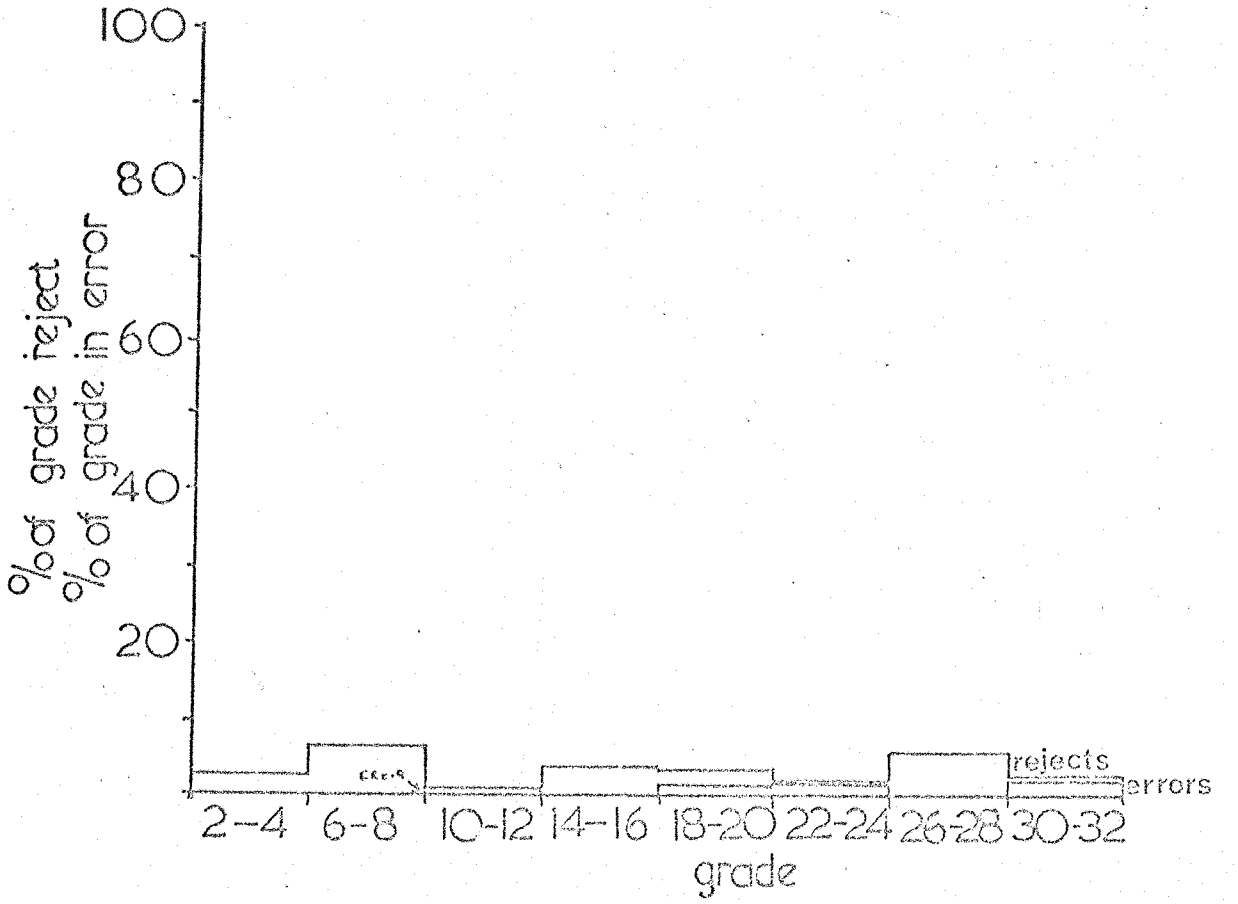
Finally the structure of the pseudo-random number generator restricted the search. If the number sequence provided by the generator coincided in some profound way with the problem structure, then very poor results could be expected. However, it is felt that the chances of this occurring in practice are remote.

4.4.5 Comparison with Intuitive Design Methods

Parallel but independent work was also carried out on the intuitive design of operators. This work involved the manual selection of each element in a set of 108 operators. The design engineer called upon several years of experience in this field and produced a set of operators for the OCR B font in about six months. The recognition system which was designed in this way was given the identical character set employed in 4.4.3. The performance is summarized in Table 14 and Figure 21.

The exact details of the intuitively designed operators and the recognition scheme are not available because of commercial secrecy.





. Figure 21 . Error and reject rates — intuitive design

GRADE	1- 3	5- 7	9-11	13-15	17-19	21-23	25-27	29-31
% REJECT	0.0	26.9	33.8	29.0	10.9	23.5	57.5	91.9
% ERROR	0.0	0.0	0.6	0.0	1.5	1.9	2.5	0.9

Thin Grades

GRADE	2- 4	6- 8	10-12	14-16	18-20	22-24	26-28	30-32
% REJECT	2.5	6.3	0.9	3.8	3.6	2.1	5.4	2.7
% ERROR	1.2	1.0	0.4	3.2	1.4	1.2	0.5	1.4

Thick Grades

TABLE 14

Out of 2849 characters 550 were rejected and 27 were in error. This compares with 440 rejected and 69 errors with the automatically designed system. It is seen from Figures 18 and 21, and Tables 12 and 14 that the intuitively designed system never reached an error rate greater than 3.2% but this was achieved at the expense of a large rejection of the thinner grades of character.

The total number of elements used in the 108 operators was 849 and this compares with the 352 used in the automatically designed set of 64.

	No. ERRORS	No. REJECT	No. ELEMENTS	No. OPERATORS	DESIGN EFFORT
INTUITIVE	27	550	849	108	0.5 man year
AUTOMATIC	69	440	352	64	30 hours computation

TABLE 15

The intuitive design method required ½ man year of design effort and this must be compared with the 30 hours of computation on a DDP 516 necessary for the automatic method. At current rates the manual techniques were by far the more expensive.

4.5 Summary

This chapter has examined the problem of the design of operators for Optical Character Recognition. Current methods rely heavily upon a combination of intuitive heuristics and manual intervention. It has been shown that the design of the more efficient type of orthogonal operator is not amenable to an intuitive approach, nor has a suitable search heuristic been produced in the literature.

The evolutionary search procedure was used in this work and enabled most of the effort to be concentrated upon the study of different evaluations rather than the effects of different search techniques. This meant that the properties of each type of operator could be assessed without the burden of first designing a set of heuristic search rules. Several useful search heuristics became apparent after some results were obtained and were used at later stages to expedite the original search.

A set of othogonal operators was constructed automatically and evaluated over real data to give encouraging results. These results

were especially promising in view of the poor print quality, the simplicity of the operators, and most important, the extensibility of the recognition system (see 4.3.1 and 5.2.1). Finally a comparative experiment was described which used intuitive design methods and demonstrated that a comparable performance was achieved by the automatic method at a fraction of the design costs.

5. Conclusions and Future Work

5.1 Conclusions

5.1.1 The Evolutionary Method

The intention in this thesis has been to deomonstrate the philosophy and use of evolutionary search processes. During the study it became apparent that the evolutionary method possessed several advantages over other approaches when applied to a certain class of problem. On the other hand it was shown that when employed on a totally different type of problem the method was extremely inefficient.

The non-exhaustive nature of evolutionary searches makes them eminently suitable for problems involving very large search spaces and little or no a priori information. Their application to such problems can enable useful heuristics to be inferred from the results and used to expedite the original search process (chapter 4). Conventional graph traversing techniques are only successful in large problems if powerful heuristics are known which prevent an unmanageable number of nodes from being expanded. Hill climbers and to a certain extent the evolutionary searches, require advance information about the 'smoothness' of the hills in the search space; such information is often lacking.

Useful heuristics or analytic properties of the search space are often known before the solutions to certain problems are attempted. In these instances a purely evolutionary approach is likely to be inefficient, indeed if sufficient structural information is available,no guesswork or intuition should be necessary and the goal can be reached directly (3.6).

5.1.2 Reduction of Finite-State Machines

Evolutionary search concepts were applied to the problem of the reduction of finite-state machines. It was hoped that some of the large computational requirements which are commonly encountered in this area might be reduced by using such an approach.

A set of rules was defined for merging pairs of states after they had been functionally compared using sets of random input sequences. The results indicated that in order to obtain a valid reduction, an excessive amount of computation was necessary. It was found that the amount of processing was in fact dependent upon the internal structure of the machine being reduced, more precisely upon the length of the longest minimal distinguishing sequence required to demonstrate that a pair of states was incompatible. This meant that a machine which was functionally equivalent to a counter for example, would probably never be reduced accurately because of the length of the distinguishing sequences. In conclusion it can be said that the technique as originally described was not feasible as a general purpose tool for the reduction of finite-state machines. However, the results did not invalidate the rules for merging states and these remain as a useful deterministic reduction method (3.6).

A similar criticism can be levelled at the method for the determination of preserved partitions described in Appendix C, and a deterministic version of the algorithm is again to be preferred.

5.1.3 Optical Character Recognition

The most significant demonstration of the evolutionary method was in the design of operators in the problem of Optical Character Recognition. This was a practical problem which demanded the automatic recognition of over 2800 OCR B characters over a complete range of print qualities.

It was found that the success of the evolutionary search was not affected by the complexity of the evaluation criteria. In other words the precise details of the search mechanism did not need to be altered every time a change was made to the evaluation. This meant that operators could be designed according to a variety of requirements without the burden of designing different sets of heuristic search rules. Effort was therefore concentrated upon the study of the operators arising from each evaluation rather than the effects of different search techniques.

Several results were obtained using this search method and some initial observations were made:

a) Although operators which matched single classes of character were often straightforward to design (intuitively and automatically), they were shown to be insufficient in their extraction of class information from the test characters. This meant that such operators were unsuitable for the recognition of very poor quality print.

b) It was found that a restriction on the size of the operator led to recognition weaknesses which were directly attributable to this size limitation.

In view of these considerations operators were allowed to span the entire character area, and an improved evaluation criterion was chosen. This criterion gave rise to sets of operators with 'orthogonal' response histograms, or equivalently, to sets of operators which were optimal in an information theoretic sense. Efficient recognition could still be achieved therefore, even if a percentage of the operators failed to function in the way the engineer intended. Moreover such a scheme was economic not only in the number of operators required for a particular problem, but also in the number of elements required for each operator*.

Many operators having various performances were obtained using the evolutionary search. A closer study of the search paths leading to the more useful operators yielded several imprecise properties or heuristics some of which were used to add efficiency to the original search:

1. The first search heuristic which was employed limited the number of elements to be ≤ 11 and allowed changes to be made only to single elements.
2. Changes were limited to alterations in position of the elements. Parity changes were found to be unproductive.
3. Operators which matched single classes were most likely to possess 8 elements (4 black seeking and 4 white seeking).

---

* The total number of black and white seeking points in the 64 operator set described in Appendix E was 352 or an average of 5-6 points per operator.

4. Orthogonal operators were most likely to possess between 5 and 6 elements, at least 2 of which were black seeking. Operators were therefore initialized with 7 elements at least 2 of which were black seeking.

5. Improved performances were obtained by employing a window size which spanned whole characters.

A set of 64 operators was produced and evaluated over real data to give results which are summarized in Figure 18. It is seen that the reject rates and error rates increase as the quality of the thinner (odd grades) type of character decreases. However, this does not occur with the thick (even grades) characters and probably indicates that a large proportion of the errors and rejects are due to an insufficient number of operators fitting the characters, rather than a large number of operators all fitting the wrong characters.

It is also significant to point out that the shape of the reject histograms follows the shape of the error histograms quite closely. This tends to confirm that the quantity of data employed was sufficient to give a realistic indication of the performance of the system. If insufficient data had been used then one might not have observed such stable trends in the error/reject histograms.

A detailed study of the errors and rejects indicated that the majority were associated with a few character confusions, e.g. O-Ø, E-F, P-R. It was clear that the overall performance would be substantially improved if these confusions could be resolved.

The difficulty of some of the character classifications was demonstrated by comparing the machine performance with that of a small group of human recognizers. It was found that the machine made fewer errors but rejected more characters than the human recognizers.

A much more extensive experiment was carried out which compared the performances of operators derived by evolutionary methods and those derived using human intuition. The manually designed operators were applied to the same data set and the two recognition performances compared. In a data set of 2849 characters it was found that the automatically designed operator set made 42 more substitution errors but rejected 110 fewer characters than the intuitively designed system. The automatically designed logic was also found to be half as costly in terms of operator complexity (numbers of gates) and considerably cheaper in terms of design effort (Table 15).

This study of the orthogonal type of operator has shown that it is not necessary to process vast quantities of data in order to design useful operators. Instead it is sufficient to employ only a relatively small set of theoretical archetype characters in order to determine good positions for operator elements. Indeed it is perhaps easier to imagine that there is no real need to search beyond the basic construction of the characters for additional information as to their classification*.

---

* A study of specific degradations would supply more information provided that the degradations themselves could be recognized.

This conclusion relieves the enormous data handling problem which is seen especially in the work of Kamentski and Liu (61) and Liu (62).

Probably the greatest advantage of the orthogonal operator set lies in its extensibility. The same evolutionary method of construction can be used to generate more operators which will resolve difficult character dichotomies, and at the same time maintain a useful capability over the rest of the classes (5.2.1). In this way errors arising from common character confusions are removed but not at the expense of other spurious errors.

The performance of a pattern recognition system can never be predicted with any certainty over unseen data. It is only possible to demonstrate performance over a limited quantity of data which is hopefully representative of that expected in practice. Indeed a doubt which is commonly expressed about pattern recognition systems in general, concerns their ability to generalize into unseen data. It is a question which can always be asked and never be satisfactorily answered. However, the author feels that this criticism has no real foundation for it can only be resolved by a straightforward evaluation of the unseen data. The data is then no longer 'unseen' in the sense that no information is gained about the expected performance on the next batch of unseen data. It is much more important to determine precisely <u>how</u> a pattern recognition system can be improved if, perhaps at a later stage, the characteristics of the data move outside the limits which can be tolerated by the system. In other words the pattern recognition system itself must be capable of accepting beneficial modifications in the light of results obtained long after the initial training process. Such a procedure for the orthogonal operator OCR system is described in the next section.

5.2 Areas for Further Study

5.2.1 Optical Character Recognition

The work described in this thesis has revealed an important avenue of study for further research in Optical Character Recognition. The practical OCR problem tackled in this thesis is by no means solved, and a considerable amount of research effort is needed before the results could give worthwhile economic gains over conventional techniques. However, the work on Optical Character Recognition comes closest to providing an economically viable method. In order to achieve this it is proposed that the recognition system be extended to cater for more stringent error rates and reject rates.

As has been described in the thesis (4.3.2 and 4.3.3) evolutionary methods exist for the generation of K-1 orthogonal operators in a K-class problem. However, it is found in such a system that the majority of errors and rejects can be attributed to just a few difficult character classes. These errors can be studied and representative characters chosen (e.g. some Ø's) to act as new archetype characters for the K+1 th class*. A new operator is then evolved thereby producing a (K+1) x (K+1) response matrix and maintaining the orthogonality between the operator responses themselves (4.3.1). During recognition, of course, the K+1 th class will be identified with the original class (e.g. the Ø-class). This technique can be applied each time it is felt that the recognition of certain classes is unsatisfactory. As the system expands in this way, it would be quite in order for an earlier operator to be overwritten by a newly evolved operator which has greater capability over the extended set of classes.

---

* These archetypes might, for instance, reflect particular forms of degradation.

The multifont problem can be approached in much the same way; the number of classes would simply be extended each time a new character variant was found to be causing an excessive number of errors or rejects. For instance, one would almost certainly require a lower case 'A' class (a). In this way the system could be improved towards either the recognition of poorer quality characters, or the recognition of a greater number of character fonts, or both.

5.2.2 Finite-State Machine Studies

The evolutionary search has been shown to be inefficient when applied to the reduction of finite-state machines. Nevertheless the rules which have been set down for merging pairs of states can still be applied algorithmically with a guarantee of accurate results.

As described earlier the merging technique can yield several different reductions of the same machine by beginning the process with a different ordering of states. Further study is required, therefore in order to remove this apparently arbitrary starting point for the reduction. It seems likely that the method will achieve its best performance when the reduction has available as many alternative paths as possible at every stage in the process. In other words rather than first merging the rows containing mainly don't care entries and leaving an unreducible set of non-redundant rows, it would be more efficient to attempt to merge the non-redundant rows first. This means that a good ordering would be one which put the rows with the fewest don't care entries at the top, and the greatest number at the bottom.

It was emphasised that the reduction of finite-state machines only has real value if simple structure is not already

present,and moreover that a knowledge of the preserved partitions of a machine directs attention towards this structure. It is only a small step further to propose that redundant structure can be added in order to simplify machine implementations, perhaps to achieve a shift register configuration. This is the 'state splitting' problem (29), as yet unsolved in a practical sense.

5.2.3 Fingerprint Classification

Fingerprint processing and file searching are becoming increasingly severe information bottlenecks owing to the volume of prints handled. This problem is aggravated by the decreasing number of fingerprint analysts.

A second major problem is the need for an effective single print classification technique, that is, one which generates a large number of classes. The currently accepted Henry system and also the Battley system are both inadequate (69) for positive identifications from single prints. However, just as with OCR, the chief obstacle to automatic fingerprint processing is print quality. Prints can be distorted in a variety of different ways, each of which presents considerable difficulties at the encoding stage.

Grasselli (70) encodes prints by recording the local slopes of the fingerprint ridges and then smoothing the resulting values. The transformation is completed by combining these elemental slopes. Recognition is carried out by detailed ridge following algorithms in the vicinity of the core of the encoded print. Hankley and Tou (69) employ a scheme which systematically searches along ridges and detects topological features. The topological coding generates a 'sentence' which describes the structure of the print. It also enables the print to be classified according to some ordering of the set of sentence codes.

A serious disadvantage of both techniques is the critical dependence upon the very first step in the encoding process. Providing the important information is not lost at this point, correct classifications can always be obtained. However, this criticality throws considerable doubt upon the repeatability of subsequent encodings of the same print; it is an essential requirement that encodings of the same print taken at different times should only vary within clearly defined bounds.

It is suggested that the approach taken in this thesis towards the OCR problem could also be employed in the classification of fingerprints. Features would again correspond to binary matrices and each reference print would be characterized by a binary vector which indicated which features matched the print. Scene of crime prints would then be scanned by the same set of features and the resulting vector compared in a nearest neighbour sense with the reference vectors. Ideally each reference vector would be orthogonal to the others so that the system could tolerate quite noisy prints before all the usefull recognition capability was lost. In fact as prints decreased in quality, fewer features would match, and this in turn would result in a smaller search reduction ratio. The fingerprint problem differs fundamentally from OCR in that the print classes are not necessarily predetermined. In practice this would mean that the classes would be formed by the set of features employed and would be satisfactory only if each class contained a sufficient number of prints.

In short this approach has some clear advantages over current automatic methods of fingerprint classification:

i) Once the feature set has been evolved, all fingerprint encoding and classification could be automatic.

ii) The effect of noise on the system reduces the search reduction ratio rather than creating classification errors.

iii) The system can be extended to cater for more prints by increasing the number of features.

Of course the bulk of the work lies in the determination of the orthogonal features themselves and it would be an extremely interesting project to discover precisely what form these features would take.

5.2.4 Fuzzy Sets

The concept of a fuzzy set was introduced by Zadeh (71) in order to formalize what is meant by a set of objects with a continuum of grades of membership. Zadeh (72) extended the concept to fuzzy algorithms in an attempt to describe imprecise operations which are commonly encountered in real life, e.g. instructions for parking a car. These ideas bear a close relationship to the work carried out in this thesis and are now discussed in this context.

A fuzzy set is defined as follows:

> A fuzzy set A in X is characterized by a membership function $f_A(x)$ which associates with each point in X a real member in the interval [0, 1], with the value of $f_A(x)$ at x representing the 'grade of membership' of x in A.

SOUTHAMPTON UNIVERSITY LIBRARY

Zadeh described algorithms as fuzzy if they contained names of fuzzy sets. In this way precise meanings were assigned to fuzzy instructions through the use of the membership functions of the fuzzy sets which entered into such intructions. However, as Zadeh pointed out, "the assignment of a precise meaning to a fuzzy instruction does not in itself resolve the ambiguity of how it should be executed". Certainly if the various membership functions are interpreted as probability measures, then the execution of the fuzzy instruction presented no difficulty. Zadeh mentioned other methods of execution, but emphasized that he had not provided a definitive answer to this question.

Probably the most attractive feature of fuzzy algorithms lies in their indefinite nature. The success of a fuzzy algorithm should not depend on the knowledge of the "precise specification of the membership functions of the fuzzy sets entering into such instructions (72) It is in this sense that fuzzy algorithms and the evolutionary algorithms described in this thesis overlap in their general intent. It is important to observe that many of the fuzzy sets which occur in Pattern Recognition do not have a priori membership functions. In this thesis for example, the probability of error for an evolved set of OCR operators can only be obtained by carrying out a series of real evaluations. This is because the evolutionary method does not necessarily rely upon a priori heuristics during a search; it is only after the search has been completed that a meaningful measure can be attached to the result.

Zadeh's original paper (71) has stimulated considerable interest in the theory of fuzzy sets, mainly because it provides a way of actually expressing many of the practical problems now facing computer scientists. This discussion has highlighted the 'fuzzy' nature of evolutionary algorithms, and it is suggested that future work might be concerned with the relation between the 'quantity of evaluation' and the accuracy or 'fuzziness' of evolutionary results.

5.3 Philosophy of the Thesis

This thesis has discussed the basic thinking behind evolutionary concepts and has reported its application in two problem areas. It has always been emphasised that an evolutionary search should not be directed at the outset by preconceived heuristic rules. The advantages of this approach can never be demonstrated rigorously, but it can be shown that particular deductive searches frequently get 'trapped' before a satisfactory solution has been obtained. The final test is, of course, an empirical evaluation carried out on a practical problem.

It is not feasible to reject all heuristics during a search, for this is tantamount to carrying out a totally random procedure. The research has shown however, that the use of a suitably chosen evolutionary 'small change' heuristic (2.5.3) was sufficient to generate encouraging results. Indeed the philosophy throughout the work has been to eliminate unjustified restrictions from the searches, and to allow the evolution to proceed in as unconstrained a manner as was possible within the framework of the various problems. Only after a set of useful results have been produced are search heuristics extracted and used to enhance the original search process.

Currently there appears to be a detectable move away from the more traditional and formalistic methods of computer science. Minsky (73) goes as far as saying that "an excessive preoccupation with formalism is impeding the development of computer science". Two examples of this trend away from conventional methods were reported recently. The first is concerned with the problem of fast digital image registration (74), and the second with digital fault detection (75). Both techniques incorporate a random element and compared with earlier methods, they make considerably reduced demands on computation. The author has conducted several preliminary experiments which investigated the feasibility of a non-deterministic definition of randomness (4). This definition is based upon the concept of an evolutionary procedure and bears some relation to the problem of 'structure detection' in pattern recognition. Whether the non-analytic technique becomes popular surely depends upon its success in a wide variety of problems. Over the next few years one might expect an increasing volume of research to be devoted to an evaluation of such methods; only then will it be possible to attach realistic and meaningful values to each approach.

6. REFERENCES

1	Fogel, L.J., Owens, A.J., & Walsh, M.H.	"Artificial Intelligence Through Simulated Evolution", John Wiley: New York, 1966.
2	Stentiford, F.W.M. & Lewin, D.W.	"Heuristic Procedure for the Reduction of Finite-State Machines", Electron. Lett., Vol.7, No.23, pages 700-702, Nov. 1971.
3	Stentiford, F.W.M.	"Reduction of Internal States in a Sequential Machine using an Evolutionary Procedure", British Computer Society, Logic Design Meeting, 11th. Oct., 1972.
4	Stentiford, F.W.M. & Lewin, D.W.	"An Evolutionary Approach to the Concept of Randomness", British Computer Journal, Vol.16, No.2, May, 1973.
5	Minsky, M.	"Steps Towards Artificial Intelligence", Proc. IRE, 49, pages 8-30, 1961.
6	Michie, D.	"The Intelligent Machine", Science Journal, pages 50-54, Oct. 1970.
7	Michie, D.	"Future for Integrated Cognitive Systems", Nature, Vol.228, pages 717-722, Nov.21, 1970.
8	Keyes, R.W.	"Physical Problems and Limits in Computer Logic", IEEE Spectrum, pages 36-45, May 1969.
9	Kuck, D.J.	"ILLIAC IV Software and Application Programming", IEEE Trans.on Computers, Vol. C-15, No.8, pages 758-770, Aug. 1968.
10	Bremermann, H.J.	"Optimization Through Evolution and Recombination", Self-Organizing Systems 1962, Spartan Books: Washington, 1962.
11	George, F.H.	"Heuristic Programming a Survey", Int.J.Systems Sci., Vol.3, No.1, pages 93-112, 1972.
12	Minsky, M.	"Descriptive Languages and Problem Solving", in "Semantic Information Processing", ed. M.Minsky, MIT Press: Cambridge Mass., 1968.
13	Michie, D.	"Heuristic Search", The Computer Journal, Vol.14, No.1, pages 96-102, 1971.
14	Feigenbaum, E.A.	"Artificial Intelligence: Themes in the Second Decade", Edinburgh IFIP, J10-J14, 1968.
15	Minsky, M.	"Some Methods of Artificial Intelligence and Heuristic Programming", NPL Symposium No.10, "Mechanization of Thought Processes", 24th.-27th.Nov. 1958, pages 5-27, HMSO: London 1959.

16	Hall, P.A.V.	"Branch-and-Bound and Beyond", Proc IJCAI 2,1971.
17	Hart, P.E., Nilsson, N.J. & Raphael, B.	"A Formal Basis for the Heuristic Determination of Minimum Cost Paths", IEEE Syst. Sc. and Cybernetics, SSC-4,2, pages 100-107, July 1968.
18	Feigenbaum, E.A. & Feldman, J.	"Computers and Thought", McGraw-Hill: London 1963.
19	Hartmanis, J. & Stearns, R.E.	"Some dangers in State Reduction of Sequential Machines", Information and Control, 5, pages 252-260, 1962.
20	Paull, M.C. & Unger, S.	"Minimizing the Number of States in Incompletely Specified Sequential Switching Functions", IRE Trans, EC-8, pages 356-367, 1959.
21	Grasselli, A. & Luccio, F.	"A Method for Minimizing the Number of Internal States in Incompletely Specified Sequential Networks", IEEE Trans., EC-14, pages 350-359, 1965.
22	De Sarkar, S., Basu, A.K. & Choudhury, A.K.	"Simplification of Incompletely Specified Flow Tables with the help of Prime Closed Sets", IEEE Trans.,C-18, pages 953- 956, 1969.
23	Kella, J.	"State Minimization of Incompletely Specified Sequential Machines, IEEE Trans., C-19, pages 343-348, 1970.
24	House, R.W. & Stevens, D.W.	"A New Rule for Reducing CC Tables", IEEE Trans, pages 1108-1111, Nov. 1970.
25	Bennetts, R.G., Washington, J.L. & Lewin, D.W.	"A Computer Algorithm for State Table Reduction", IEEE Eurocon 71 Conf., Switzerland, Oct. 1971.
26	Lewin, D.W.	"Logical Design of Switching Circuits", Nelson: London, 1968.
27	Hartmanis, J.	"On the State Assignment Problem for Sequential Machines I", IRE Trans on Electronic Computers, Vol.EC-10, No.4, pages 157-165,Jun'61.
28	Muller, D.E. & Putzolu, G.R.	"Frequency of Decomposability Among Machines with a Large Number of States", J. of Computer and System Sciences, 2, pages 219-227, 1968.
29	Hartmanis, J. & Stearns, R.E.	"Algebraic Structure Theory of Sequential Machines", Prentice-Hall, 1966.

30	Assmus, E.F. Jr. & Florentin, J.J.	"Algebraic Machine Theory and Logical Design", from "Algebraic Theory of Machines, languages and semigroups", ed. Arbib M.A. Academic Press: New York, 1968.
31	Walker, B.S.	"A Node Logic System", British ComputerSociety Logic Design Meeting, 14th. June, 1972.
32	Nagy, G.	"State of the Art in Pattern Recognition", Proc. IEEE, Vol.56, No.5, pages 836-862, May 1968.
33	Kovalevsky V.A.	"The Current State of the Problem of Pattern Recognition", Trans in Cybernetics, New York: Consultants Bureau, 1971.
34	Bremermann, H.J.	"What Mathematics Can and Cannot do for Pattern Recognition", Pattern Recognition in Biological Systems, Proc. 4th. Cong. of Deutsche Gesellschaft für Kybernetik, Berlin Technical University, pages 31-44, April 6-9'70.
35	Friedberg, R.M.	"A Learning Machine, part 1", IBM J.Res. and Dev., Vol 2, pages 2-13, January 1958.
36	Minsky, M. & Selfridge, O.G.	"Learning in Random Nets", 4th. London Symp., on Information Theory, ed. C.Cherry, London: Butterworths, 1961.
37	Turing, A.M.	"Can A Machine Think", Mind, Vol.59, No.236, pages 433-460, October 1950.
38	Solomonoff, R.J.	"Some Recent Work in Artificial Intelligence" Proc. IEEE, Vol.54, No.12, pages 1687-1697, December 1966.
39	Mucciardi, A.N. & Gose, E.E.	"Evolutionary Pattern Recognition in Incomplete Nonlinear Multithreshold Networks", IEEE Trans. on Electronic Computers, pages 257-261, April 1966.
40	Chow, C.K. & Liu, C.N.	"An Approach to Structure Adaption in Pattern Recognition", IEEE Trans. on System Science and Cybernetics, Vol.SSC-2, No.2, pages 73-80, 1966.
41	Kaufman, H.	"An Experimental Investigation of Process Identification by Competitive Evolution", IEEE Trans. on System Science and Cybernetics, Vol.SSC-3, No.1, pages 11-16, June 1967.
42	Klopf, A.H. & Gose, E.	"An Evolutionary Pattern Recognition Network", IEEE Trans. Syst. Sc. and Cyb. pages 247-250, July 1969.

43	Lin, S.	"Computer Solutions of the Travelling Salesman Problem", Bell System Technical Journal, Vol.44, No.10, December 1965.
44	Siklossy, L. & Marinov, V.	"Heuristic Search vs. Exhaustive Search", Proc. 2nd. Int. Conf. on A.I., 1-3rd.Sept'71, pages 601-607.
45	Nilsson, N.J.	"Problem Solving Methods in Artificial Intelligence", McGraw-Hill: New York 1971.
46	Perryman, G.	"Discovering the Structure of an Automaton from Partial Information", M.Sc. thesis, Edinburgh University, 1970.
47	Unger, S.H.	"Flow Table Simplification - Some Useful Aids", IEEE Trans. on Computers, Pages 472-475, June 1965.
48	Bennetts, R.G.	"Verification of Reduced-State Tables Based on the Pseudoequivalent Merging Technique", Elect. Lett., Vol.8, No.4, pages 84-86, Feb.1972.
49	Scott, R.V.	"Reduction of Internal States in a Sequential Machine Using Heuristic Procedures", M.Sc. Thesis, Southampton University, 1972.
50	Thomas, W.	"The Pseudo-Equivalent Merging Technique for State Reduction", B.Sc. Report, Southampton University, May 1973.
51	Bell, H.A.	"OCR : What happened to those Market Predictions" from "The New Technologies", Infotech Information Limited, Maidenhead, pages 351-365, 1971.
52	Andersson, P.L.	"Optical Character Recognition - A Survey", Datamation, pages 43-48, July, 1969.
53	Van Steenis, H.	"The IBM 1275 Recognition System and Its Development", Pattern Recognition in Biological and Technical Systems, Proc. 4th. Cong. of Deutsche Gesellschaft für Kybernetik, Berlin Technical University, April 6-9, 1970.
54	Hoffman, R.L. & McCullough, J.W.	"Segmentation Methods for Recognition of Machine Printed Characters", IBM J. Research Development, pages 153-165, March, 1971.
55	Turner, L.F.	"A System for the Automatic Recognition of Moving Patterns", IEEE Trans. on Information Theory, Vol.IT-12, No.2, pages 195-205,April 1966.
56	Andrews, D.R., Atrubin, A.J. & Hu, K.C.	"The IBM 1975 Optical Page Reader, Part III: Recognition Logic Development", IBM J. Res. Development, 12, No.5, pages 364-371, 1968.

57	Bledsoe, W.W. & Browning, I.	"Pattern Recognition and Reading by Machine", Proc. Eastern Joint Comp. Conference, pages 225-232, 1959.
58	Evey, R.J.	"Use of a Computer to Design Character Recognition Logic", Proc. EJCC, pages 205-211, 1959.
59	Uhr, L. & Vossler, C.	"A Pattern Recognition Program that Generates, Evaluates and Adjusts its Own Operators", Proc.WJCC, pages 555-561, 1961.
60	Zobrist, A.L.	"The Organization of Extracted Features for Pattern Recognition", Pattern Recognition, Vol. 3, pages 23-30, 1971.
61	Kamentsky, L.A. & Liu, C.N.	"Computer-Automated Design of Multifont Print Recognition Logic", IBM J. Res. Development, Vol.7, pages 2-13, January, 1963.
62	Liu, C.N.	"A Programmed Algorithm for Designing Multi-font Character Recognition Logics", IEEE Trans. on Electronic Computers, Vol.EC-13, pages 586-593, October 1964.
63	Liu, C.N. & Shelton Jr., G.L.	"An Experimental Investigation of a Mixed-Font Print Recognition System", IEEE Trans. on Electronic Computers, Vol. EC-15, No.6, December 1966.
64	Archer, R.N., Koppleman, G.M., Miller, M.J., Nagy, G. & Shelton, Jr. G.L.	"An Interactive System For Reading Unformatted Printed Text", Vol.C-20, No.12, pages 1527-1543, December 1971.
65	Britt, R.H.	"A Practical Technique for Feature Extraction", IEE Conf. on Pattern Recognition, 29-31 July, NPL publication No.42, pages 347-354, 1968.
66	Coombs, A.W.M.	"On the Construction of an Efficient Feature Space for Optical Character Recognition", Machine Intelligence 4, Edinburgh University Press, pages 385-401, 1969.
67	Flores, I. & Grey, L.	"Optimization of Reference Signals for Character Recognition Systems", IRE Trans. on Electronic Computers, EC-9, No.1, pages 54-61, March 1960.
68		"Standard ECMA-11 for the Alphanumeric Character Set OCR B for Optical Recognition", ECMA European Computer Manufacturer's Assoc., Geneva, November 1965.

69 Hankley, W.J. & Tou, J.T. "Automatic Fingerprint Interpretation and Classification Via Contextual Analysis and Topological Coding", in Pictorial Pattern Recognition, Thompson Book Co., Washington,1968.

70 Grasselli, A. "On the Automatic Classification of Fingerprints - Some Considerations on the Linguistic Interpretation of Pictures", in Methodologies of Pattern Recognition, ed. S.Watanabe, Academic Press, New York, 1969.

71 Zadeh, L.A. "Fuzzy Sets", Information and Control, Volume 8, No.3, pages 338-353, June 1965.

72 Zadeh, L.A. "Fuzzy Algorithms", Information and Control, Volume 12, pages 94-102, 1968.

73 Minsky, M. "Form and Content", JACM, Vol.17, No.2, pages 197-215, April 1970.

74 Barnea, D.I. & Silverman, H.F. "A Class of Algorithms for Fast Digital Image Registration", IEEE Trans. on Computers, Vol.C-21, No.2, pages 179-186, Feb. 1972.

75 Breuer, M.A. "A Random and an Algorithmic Technique for Fault Detection Test Generation for Sequential Circuits, IEEE Trans. on Computers, Vol.C-20, No. 11, pages 1364-1370, November 1971.

76 Stentiford, F.W.M. Internal Plessey Report, Plessey Radar Research Centre, 1973.

77 Hartmanis, J. "Loop-Free Structure of Sequential Machines", Inf. and Control, 5, pages 25-43, 1962.

The following published papers were included in the bound thesis. These have not been digitised due to copyright restrictions, but the links are provided.

http://dx.doi.org/10.1049/el:19710479

APPENDIX B SEQUENTIAL MACHINES : DEFINITIONS

B.1 Machines

The two classic finite machine models are defined (29):

A Moore type sequential machine is a quintuple

$$M = (Q, I, Z, \delta, \lambda)$$

where i) Q is a finite nonempty set of states;

ii) I is a finite nonempty set of inputs;

iii) Z is a finite nonempty set of outputs;

iv) δ: $Q \times I \rightarrow Q$ is called the transition (or next state) function;

v) λ: $Q \rightarrow Z$ is called the output function.

A Mealy type sequential machine is a quintuple

$$M = (Q, I, Z, \delta, \lambda)$$

where i) Q is a finite nonempty set of states;

ii) I is a finite nonempty set of inputs;

iii) Z is a finite nonempty set of outputs;

iv) δ: $Q \times I \longrightarrow Q$ is called the transition function;

v) λ: $Q \times I \longrightarrow Z$ is called the output function.

B.2 Partitions

A partition π on a set Q is a collection of disjoint subsets of Q whose set union is Q. The sets of π are called blocks of π and $s \equiv t$ (π) iff s and t are contained in the same block of π.

A partition on the set of states of the machine

$$M = (Q, I, Z, \delta, \lambda)$$

is called a preserved partition* iff $s \equiv t\ (\pi)$ implies that

$$\delta(s,x) \equiv \delta(t,x)(\pi) \text{ for all } x \in I.$$

Addition(+) and multiplication(·) can be defined on the partitions of a set Q. If π_1 and π_2 are partitions on Q, then:

i) $\pi_1 \cdot \pi_2$ is the partition on Q such that
$s \equiv t(\pi_1 \cdot \pi_2)$ iff $s \equiv t(\pi_1)$ and $s \equiv t(\pi_2)$

ii) $\pi_1 + \pi_2$ is the partition on Q such that
$s \equiv t(\pi_1 \cdot \pi_2)$ iff there exists a sequence in Q $s = s_0, s_1, \ldots, s_n = t$ for which either $s_i = s_{i+1}(\pi_1)$ or $s_i = s_{i+1}(\pi_2)$ for $0 \leqslant i \leqslant n-1$.

π_2 is larger than or equal to π_1 iff every block of π_1, is contained in a block of π_2 and write $\pi_2 \geqslant \pi_1$. In this case π_1 is a refinement of partition π_2.

It can be shown that if π_1 and π_2 are preserved partitions on the set of states of a sequential machine M, then so are the partitions $\pi_1 \cdot \pi_2$ and $\pi_1 + \pi_2$.

Partition pairs find applications in the state assignment problem and in feedback decomposition (29).

A partition pair (π, π') on the machine

$$M = (Q, I, Z, \delta, \lambda)$$

is an ordered pair of partitions on Q such that $s \equiv t(\pi)$ implies $\delta(s,x) \equiv \delta(t,x)\ (\pi')$ for all $x \in I$.

---

* Hartmanis and Stearns (29) call these partitions substitution property (S.P.) partitions.

In order to cater for the don't cares in the original machine this definition is weakened:

If π and τ are partitions on Q, (π,τ) is a weak partition pair iff

$s \equiv t(\pi)$ implies $\delta(s,x) \equiv \delta(t,x)$ (τ) for all $x \in I$ such that $\delta(s,x)$ and $\delta(t,x)$ are specified.

If (π,τ) is a weak partition pair, then π has the comapping property with τ iff every block of π maps under I into at most one block of τ and every block of τ is mapped into by at most one block of π.

B.3 Covers

Quite frequently the blocks of partitions are mapped by the next-state mapping (δ) onto overlapping subsets of Q. When this occurs, partitions of Q can no longer be used to obtain information about the machine. The notion of a cover is introduced to overcome these difficulties.

Let Q be the state set of a machine M.

A set $\psi = \{C_i\}$ is a cover* if

a) the union of all $C_i \in \psi$ is Q,

b) $C_i \subseteq C_j$ iff $i = j$.

The C_i are called blocks of the cover.

For every subset C of Q the image of C under δ with input $x \in I$ is defined to be

$$\delta(x,C) = \{q_a \mid q_a = \delta(x,q_j) \text{ for } q_j \in C \text{ and } \delta(x,q_j) \text{ defined}\}$$

A cover $\psi = \{C_1, C_2, \ldots, C_r\}$ is called a preserved cover iff there exists one k such that $\delta(x,C_j) \subseteq C_k$ for every $x \in I$. If $\delta(x,C_j) = \emptyset$, then $\delta(x,C_j) \subseteq C_k$ for all $C_k \in \psi$.

---

* Hartmanis and Stearns (29) define a set system in a similar way.

Most of the results concerning partitions can be extended to equivalent results with covers. However, the advantages gained by the greater generality are lost to the impracticability of the algorithms. This is because in finding the set of all preserved partitions for an n-state machine approximately n(n-1)/2 calculations are needed, whereas $(2^n - n - 1)$ calculations are needed to find all of the preserved covers. Some authors conclude that although the cover concept is helpful in formulating and thinking about machine problems, and although some specialized applications can be developed, it will very seldom be a practical tool for exhaustive analysis.

B.4 State Reduction

States q_i, q_j are said to be equivalent iff every possible input sequence causes the machine to produce the same output sequence when the initial state is q_i as it does when the initial state is q_j.

A partition π on the machine M is called output consistent iff $s \equiv t(\pi)$ implies $\lambda(s) = \lambda(t)$ for all inputs x.

When machines are not defined for all possible inputs the concepts of state equivalence as well as preserved partitions are no longer applicable. The treatment of those machines requires further definitions.

A sequence that can be applied to a partially specified machine in initial state q without encountering any undefined transitions (i.e. $\delta(q_i, x)$ is always defined) is said to be applicable to state q.

State q_1 contains state q_2 ($q_1 \supseteq q_2$) iff every input sequence that is applicable to q_2 is also applicable to q_1, and results in the same output sequence regardless of whether it is applied to initial state q_2 or to initial state q_1.

It might seem that the state containment relation would be sufficient to find the minimal number of states to represent a given machine. It turns out, however, that this relation is too strong in that it is often possible to use a group of m states to do the work of another group of n states with $n>m>1$. State containment is merely the particular case of m=1. To account for this situation further definitions must be introduced.

The states are said to be compatible iff every input sequence which is applicable to both q_1 and q_2 gives rise to identical output sequences.

A set B of states is called a compatible set if every pair of states contained in B are compatible. B is a maximal compatible set if B is not a proper subset of any compatible set. The image P of a compatible set C is called the condition class for C.

A compatible set C_i is non-prime, if there exists another compatible set C_j such that

i) $C_i \subseteq C_j$

and ii) $P_j \subseteq P_i$

Otherwise C_i is a prime compatible set.

The general state reduction problem amounts to the discovery of a preserved cover with a minimal number of blocks each of which is a compatible set. (It is in fact only necessary to consider prime compatible sets).

B.5 Semigroups

A semigroup is a set with a rule of combination defined on the members of the set having

i) the closure property.

ii) the associative property.

The semigroup of a state machine $M = (Q, I, \delta)$ is the semigroup generated by the inputs regarded as mappings of Q into Q where the rule of combination is the concatenation of the input sequences.

APPENDIX C AN EVOLUTIONARY ALGORITHM FOR THE DETERMINATION OF PRESERVED PARTITIONS

An essential step in the implementation of a finite-state machine is the assignment of binary state variables (2.4.3). A knowledge of the internal structure of the machine is imperative if an elegant realization is to be obtained. The preserved partitions of the machine not only provide probably the best insight into the functioning of the internal states, but also direct attention towards the most economic assignment.

Chapter three has been concerned with an algorithm for the state reduction of machines, or equivalently, the discovery of maximal partitions whose blocks consist of compatible states. These partitions are output consistent and preserved, but only for certain input sequences that are applicable to the machine states. Preserved partitions are normally considered only when the machine is fully specified, in which case they can be usefully manipulated in accordance with supporting theory. Conventional programs for the production of preserved partitions involve a tree search and a storage requirement which goes up as the square of the number of states. As with state reduction such a technique becomes unwieldy when a very large number of states is encountered.

In the first instance completely specified machines are considered. The algorithm begins with the trivial preserved partition $\pi_0 = \{\overline{1};\overline{2};\ \ldots;\overline{n}\}$. Two initial states (q_{i_1}, q_{j_1}) are chosen and a random sequence of length m is input simultaneously to both. This result in control passing through a succession of m pairs of states:

$$(q_{i_1}, q_{j_1}), (q_{i_2}, q_{j_2}), \ldots, (q_{i_m}, q_{j_m}).$$

Each pair of states will belong to the same block of a preserved partition which also has q_{i_1} and q_{j_1} in the same block. Input sequences are terminated if a state pair is composed of identical states. Otherwise all inputs of the sequence are accepted. The state blocks in the initial partition π_0 are grouped together transitively according to the

equivalence defined by the list of state pairs, (q_{i_k}, q_{j_k}). This produces a partition π_1 with $\pi_1 > \pi_0$. Beginning with the same state pair, the grouping of state blocks in π_1 is continued with another random input sequence producing π_2 with $\pi_2 \geq \pi_1 > \pi_0$. After p such sequences we have

$$\pi_p \geq \pi_{p-1} \geq \cdots \geq \pi_1 > \pi_0.$$

If $\pi_p \quad \pi_I = \{\overline{1,2, \ldots, n}\}$ then π_p is output as a pseudo-preserved partition. If π_q becomes equal to π_I, perhaps for small q, the process can be terminated with the unambiguous decision that there is no non-trivial preserved partition with the current initial state pair in the same block. When this procedure has been applied to all n(n-1)/2 possible start-state pairs, a possibly empty set of distinct non-trivial pseudo-preserved partitions will have been generated.

It is possible that the partitions produced by this technique are not actually preserved. Greater certainty in the results is obtained by increasing the parameters m,p. However, the errors which occur in pseudo-preserved partitions are only of a particular type as is illustrated by the following theorem.

Theorem

If a non-trivial preserved partition π of a completely specified machine exists, then a non-trivial pseudo-preserved partition $\pi_p \leq \pi$ is always found.

Proof

$\pi_p > \pi_o$ since the generation of a pseudo-preserved partition requires that at least two states are grouped in the same block of π_p.

Consider a pair of states q_a, q_b in the same block of π. Suppose that $\pi_p \not\leq \pi$ using (q_a, q_b) as an initial start state pair. Then there exists a pair of states $(q_{a'}, q_{b'})$ which are in the same block of π_p but not in the same block of π. But the identification of $(q_{a'}, q_{b'})$ in the same block of π_p was inferred from the identification of (q_a, q_b).

Therefore (q_a, q_b) are not in the same block of π. Contradiction.

$\therefore \quad \pi_o < \pi_p \leq \pi$.

Ideally don't care conditions should be assigned so as to minimize the complexity of the resulting hardware. The algorithm described here fills the don't care conditions in such a manner that any preserved partitions that are produced are as small as possible. In this way assignments are made so that each binary variable describing the new state depends on as few variables of the old state as possible.

If on the kth input a don't care condition is reached from one of the initial states, q_i, then the state q_{j_k} is substituted in its place. Similarly if the don't care is reached from q_j, q_{i_k} is substituted. The input sequence is then terminated and the procedure continued as for completely specified machines. There may be many such assignments for the don't care conditions each giving rise to a different set of preserved partitions. The partitions which are in fact produced are dependent on the order in which the pairs of states are compared.

Preserved partitions can be used to decompose a composite machine into a set of interconnected smaller machines. In effect this means that assignments can be given to the component machines without any dependence upon the internal states of the other component machines. It can be shown (77) that there is a loop-free realization of a sequential

machine from n smaller machines if there corresponds a set of n preserved partitions whose product is π_0. It is observed that although this method often enables a machine to be constructed from simpler machines, it makes no contribution towards simplifying the implementation of the output mapping.

An assignment carried out on the basis of a pseudo-preserved partition might not be an accurate representation of the original machine. However, only if the machine was made to function in a way which was not explored during the derivation of the original pseudo-preserved partition, would the implementation go astray. In the same way as with state reduction, malfunctions can be made arbitrarily rare by increasing the computation, and thus the reliability of the pseudo-preserved partitions. An absolute check on the pseudo-preserved partitions can be carried out again in much the same way as with state reductions by ensuring that:

i) The partitions are preserved and
ii) all next state don't care conditions are assigned.

At this stage any errors which are found can be corrected quite simply by merging appropriate blocks of the partition π_p thereby producing the required partition π (see Theorem). However, in the light of results obtained in Chapter three, considerable doubt must be placed on the reliability of pseudo-preserved partitions unless very many input sequences are employed. As with state reduction it would be more efficient to apply the don't care assignment rules in an exhaustive fashion.

Finally an initial study of the preserved partitions of a machine will lessen the danger of simple internal structure being destroyed by state reduction(2.4.2). In this way straightforward implementations will be unlikely to go unnoticed.

APPENDIX D OCR SYSTEM SOFTWARE

D.1 General

The programmes and subroutines were written in FORTRAN and DAP-16 for the Honeywell DDP 516 16K computer. Two on-line teletypes, a high-speed reader, a high-speed punch and a storage scope are employed at various points in the system. The project was influenced to a considerable extent by the absence of any backing store, and it was this limitation which probably made the orthogonal type of operator so attractive. A pattern recognition problem is normally not thought feasible unless large quantities of data can be accessed by the computer. In spite of this, the methods employed in this thesis enabled a recognition system to be designed and evaluated entirely within 16K of core store.

In parallel with the evolutionary search programmes, the corresponding manual versions were also written. It was felt that a manual interface was essential for gaining insight into some of the problems. It turned out however, that manual methods were tedious and did not look promising in view of the limited effort available. These programmes were little used and will not be described here (76).

The work was naturally centred around two main programmes, the evolutionary operator generator and the character classifier programme.

D.2 Programme Descriptions

D.2.1 Evolutionary Operator Generator

This programme provided a software framework which hypothesized, evaluated and modified OCR operators according to the evolutionary principles set down in 4.3.3. Operators which were produced were stored in a library which could hold up to 64 operators. The programme occupied the area 100_8 - 20746_8.

Command Instructions

When initialized or interrupted the programme asked for one of four numeric commands:

1. Programme was to function without output.
2. After each modification the following was to be punched out in a teletype-listable form:
   a) Three parameters associated with the running of the programme:
      Total no. modifications.
      No. good modifications.
      Best current operator score so far.
   b) Response histogram of the modified operator.
   c) Pictorial representation of the operator.
   (for b) and c) see Figure 17).
3. As 2. but only c) was output.
4. As 1. but only after an improvement had occurred.

Additional Features of the Programme

A. Hypothesized (random) operators were rejected if they did not fit any character in the archetype set.

B. Before an operator was stored in the library each element was checked for redundancy and removed if it did not contribute, or contributed negatively, to the overall operator score.

C. An operator which was produced by the programme was stored in the operator library only if its performance exceeded at least one of those already stored. In this case, the worst operator, in terms of the current evaluation function, was overwritten.

D. This programme did not call the FORTRAN I/O library for storage reasons.

E. The Flowchart for this programme is illustrated in Figure 22.

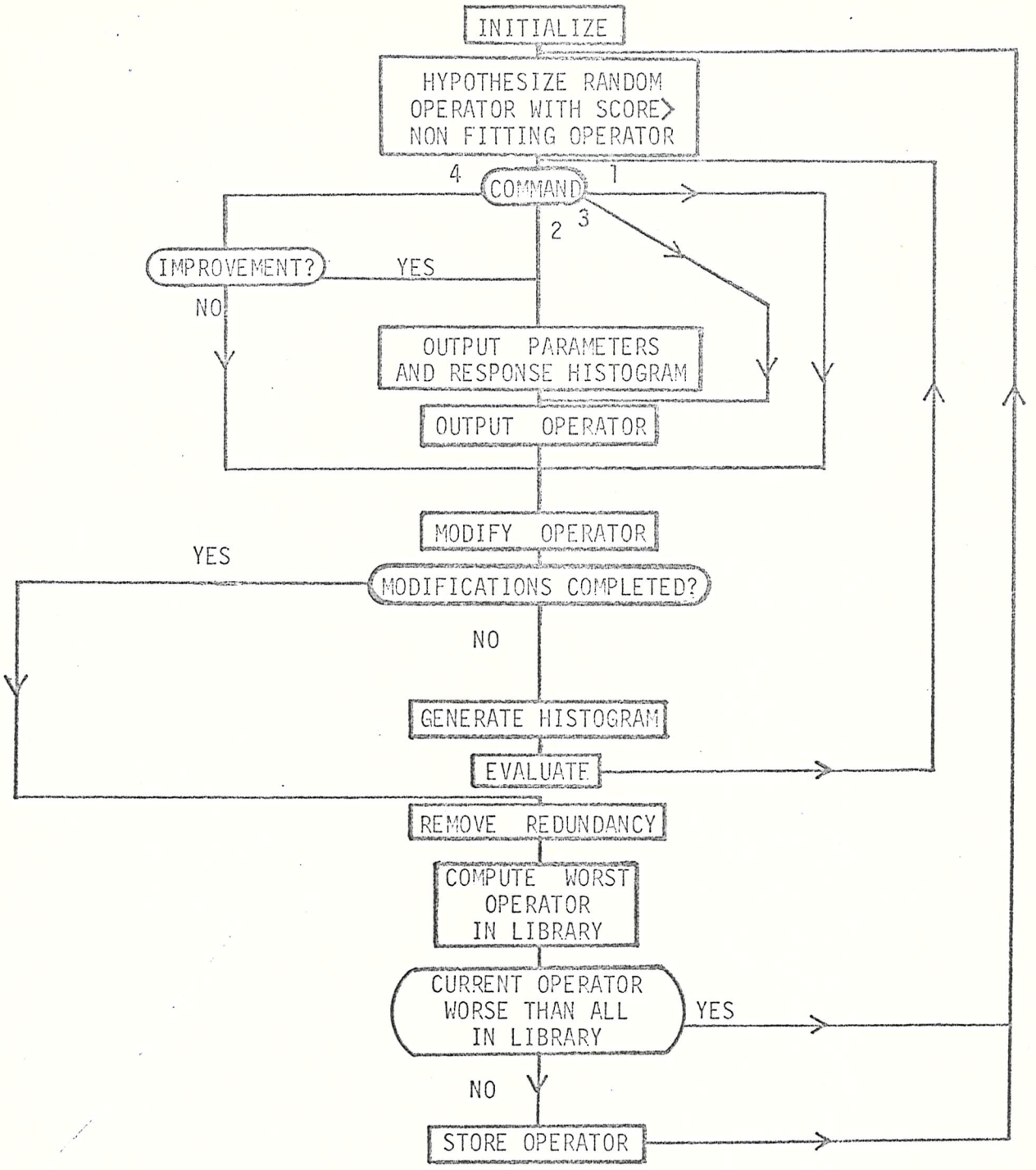


FIGURE 22 : Flowchart for Evolutionary Programme

D.2.2 OCR Classifier Programme

This programme classified and displayed the OCR B data using the nearest neighbour classifier described in 4.4.3. An operator library produced by the evolutionary programme was required for the running of this programme. The programme occupied the area 100_8 - 30000_8.

Command Instructions

When initialized or interrupted the computer asked for one of seven numeric commands:

1. Complete printout plus display. In this mode each character was displayed on the storage scope and the classification data was printed out on the second teletype.
2. Complete printout, display and hard copy. This option was as 1., but a hard copy of each character was punched out in a form which was directly listable on an offline teletype.
3. Error Reject Printout plus display. This option was as 1., but only for characters which were either errors or rejects.
4. Error Reject Printout, display and hard copy. This option was as 3 but a hard copy of each displayed character was also produced in the manner of 2.
5. Confusion Matrix. This option caused the computer to punch out a teletype-listable copy of the current confusion matrix.
6. Threshold Parameters. The computer requested two reject thresholds. The first determined the largest difference between the 1st. and 2nd. classification distances at which characters were rejected. The second determined the smallest distance for a first choice classification at which characters were rejected.
7. Pause.

D.3 Data Areas

D.3.1 Common Areas

There were three principal fixed areas in core which had a universally common usage throughout the system.

NAME	EXTENT (octal)	SIZE	FUNCTION
TRCH	23266-33451	4212	Storage space for archetypes
MIFH	33452-35651	1152	Response histogram area
MIF	35652-37651	1024	Operator area

TABLE 16 : Common Areas

The operator library always occupied the area 33452_8 - 37651_8.

D.3.2 Relocatable Areas

There were several relocatable locations and working areas which were labelled for external referencing by DAP 16 subroutines. This was done in preference to other methods for the following reasons.

a) To simplify the structure of the common areas.

b) To relieve the information load carried by subroutine arguments.

c) Debugging purposes.

d) To utilize the full power of the Honeywell loading routines.

NAME	LENGTH (Decimal)	FUNCTION
BPAT	4565	Scanning Area
GRM	1	No. Improvements Parameter
IGEN	1	Total No. Changes Parameter
WLIP	64	Working Operator Area
HIST	36	Working Response Histogram Area
NOGI	1	Max.No.bad modifications (M_o)
FBFM	480	Modified Operator pointers

TABLE 17 : Relocatable Areas

D.4 Storage of Operators

Operators were stored in the common data area MIF (35652_8-37651_8). Up to 64 operators could be stored in this area each operator occupying 16 words.

Each element of the operator was coded into one 16 bit word as follows:

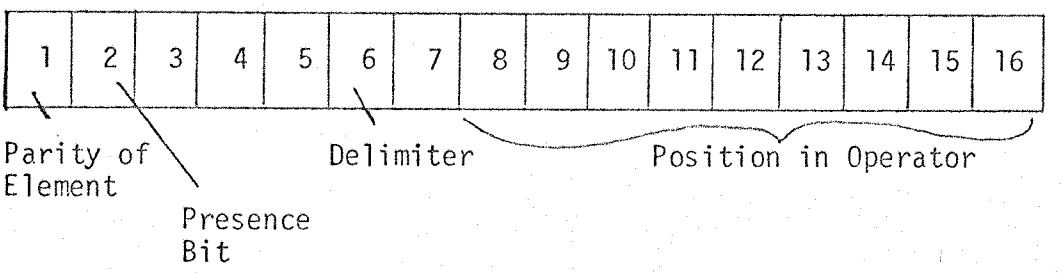


FIGURE 23

Bit 1 is 1 if the element was black seeking.

Bit 1 is 0 if the element was white seeking.

Bit 2 is 1 if an element was represented.

Bits 8-16 represented a number (0-479) indicating the position of the element in the 30 x 16 operator. The element word value 2000_8 delimited the list of operator elements.

64 operator responses, corresponding to the operators stored in MIF, were stored in MIFH. Each response histogram consisted of 36 numbers which were packed into 18 words in MIFH. Each pair of histogram slot values was packed with the first value occupying bits 1-8 and the second occupying the remainder.

D.5 Scanning

The speeds of both the evolutionary programme and the classifier programme were linearly dependent upon the speed of the scanning routine. Some effort was therefore given towards the design of a fast character scanning routine. This was achieved by utilizing the parallel nature of the Honeywell DDP 516 modifier register.

The scanning structure consisted of three data areas detailed in Table 18.

FUNCTION	LENGTH	NAMES
Working operator area	64	WLIP, WLIS
Modified operator pointers	480	FBFM
Binary character scan area	4565	BPAT

TABLE 18

The scanned area itself was sufficiently large to allow a blank margin to surround the character and to ensure that the operator was scanned over the character in all possible relative positions.

The working operator was represented by two lists WLIS and WLIP. WLIS was an ordered list of operator elements (0 or 1) which was delimited by -1. WLIP was an list of indirect pointers to the 30 x 16 array FBFM and thereby indicated the position of the corresponding elements in WLIS within the 30 x 16 operator.

FBFM was an array of 30 x 16 = 480 modified addresses which pointed to the extreme bottom left of the character scan area with zero in the modifier (Figure 24). This address list was arranged so that the relative positions of operator elements could be carried through to the character scan areas. By altering the value of the modifier, the addresses within FBFM were all effectively altered. This in turn caused the operator to move over the character.

The scan area BPAT was so arranged that sequential locations ran from top to bottom and left to right. This meant that as the modifier was incremented, the operator moved downwards and from left to right. Also it meant that the operator scanned the character as if the scan array were wrapped around a cylinder with a horizontal axis, thus removing the need for more than one stepping variable (i.e. the modifier). The margin above and below the character was also sufficiently wide to prevent the operator simultaneously overlapping the top and the bottom of the scanned character.

The search for a fit was speeded up yet again by checking the black seeking elements for a match before the white seeking elements. This had the effect of speeding the scan up over the large blank spaces outside the character area.

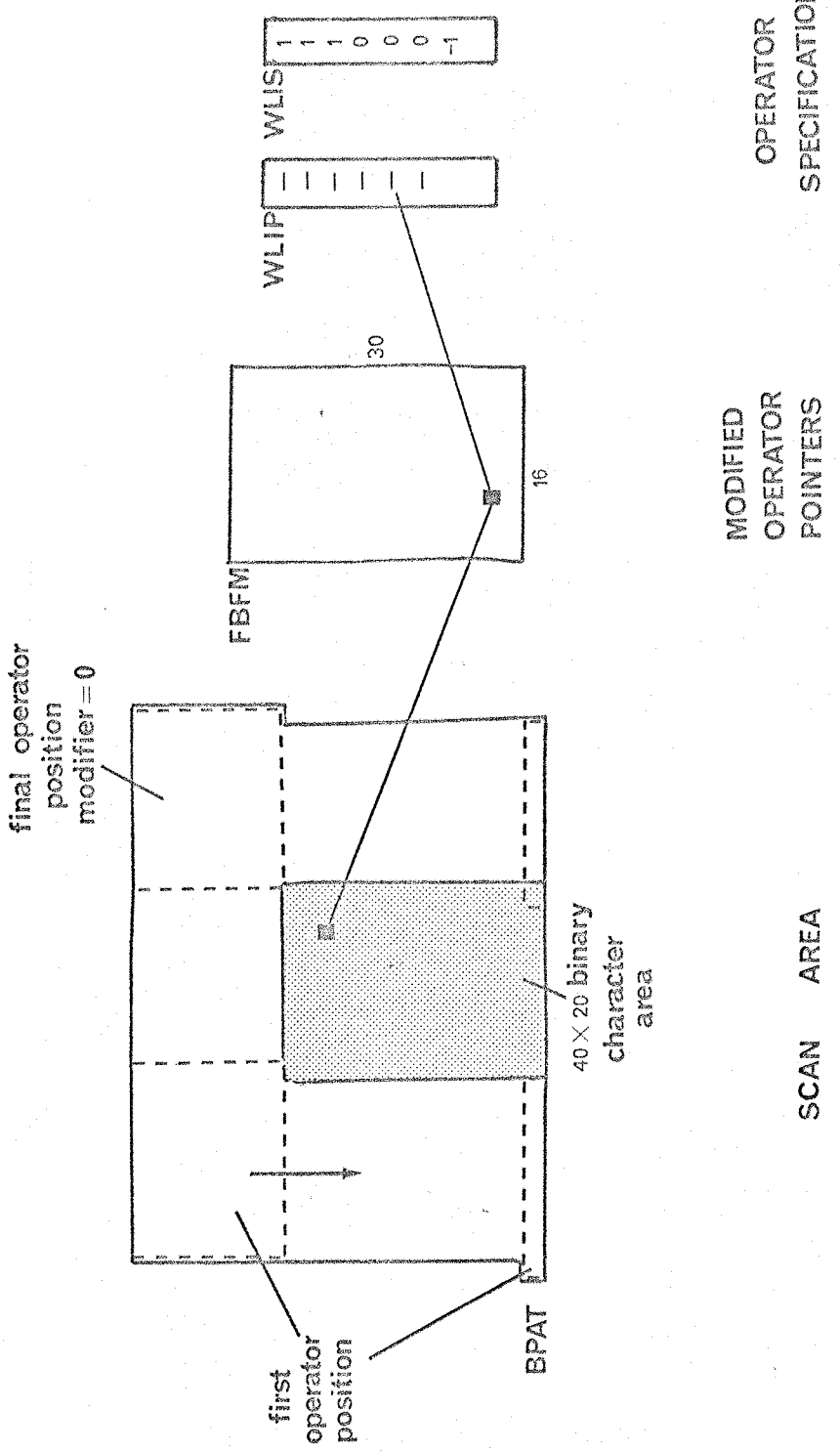


Figure 24. Scanning configuration.

Using these ideas the inner loop of the scanning routine (i.e. the match/no match check) was reduced to the following DAP 16 machine code instructions:

```
NST   LDA*  STAT
      SPL                  test if finished.
      JMP   OUT   EXIT     operator matches.
      SNZ                  test points of element.
      JMP   ZERO           white seeking.
      LDA*  ADDR           black seeking.
      SNZ                  test for black.
      JMP   NEXT  EXIT     no match.
      IRS   STAT           } increment
      IRS   ADDR           } pointers.
      JMP   NST            continue.
ZERO  LDA*  ADDR
      SZE                  test for white.
      JMP   NEXT  EXIT     no match.
      IRS   STAT           } increment
      IRS   ADDR           } pointers.
      JMP   NST            continue.
```

Each element was checked for a match in 8 or 9 instructions or 15-16 cycles (μs). The loop in which these instructions were nested, incremented the modifier ready for the next operator position.

APPENDIX E ORTHOGONAL OPERATOR SET

In this appendix the 64 OCR operators and corresponding response histograms are listed. Firstly the 64 histograms are listed in 64 rows each column corresponding to one of the 36 classes indicated at the top. Secondly each of the 64 operator configurations is listed as a row of 3-tuples. Each 3-tuple corresponds to an operator element, the first two numbers indicating the row and column co-ordinates, and the third the parity of the element. A set of zero co-ordinates does not correspond to any element.

Operator Histograms

```
   0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ
 1 222101001012212030200000203322000013
 2 003333313133022333003133032331113212
 3 333133333033330303002103033111313
 4 300133103112333020003301301020330000
 5 003333313313213321103011132331000313
 6 303303323333030132032021333330333322
 7 301123303333333333033033332331132320
 8 303220133333230210031000322200333233
 9 102303303003333031203020000030000303
10 300232313313130033030133333310333001
11 333333333012322313132113332331013 13
12 102130230210101100003000000001031333
13 203213002002333030300200303020100003
14 313333133332332120332113313121333313
15 303233333333333333023021333302123303
16 300303001001312030301100301130100300
17 302012203313333332003000333302000111
18 310312303133333131313311323231312311
19 301000010101332220003000232202010120
20 023333313233003223013032020230012313
21 301131001201332030103300303120301302
22 333333333132311302031103333331113 23
23 303011102101132123013033332200323020
24 300011302222120032001033313100323100
25 300001203202131023011033333320303020
26 103313333302022220020121332300013 03
27 313303323303333131321121313131101303
28 131331303133011132003033022221003301
29 111303333313023331110011132331101201
30 311333323333331030103313303231333331
```

```
31 333233333303333030302100303233000203
32 323333333323333232313322323332302313
33 303321231021331220230000332112010003
34 301003003300310030000000003330000200
35 101133313333013330002023032310001201
36 110313202302232010100210100031110023
37 130101001002032212332222120212201100
38 303210123313332233033021333330333323
39 200001203202311122000010332230000000
40 303221230310131111032010313100333113
41 200001000301311030000000302020011000
42 300003302201021030000002201030330000
43 303233023233030023003003033300003002
44 300032303233321032001102303210200300
45 302303303000000000000000000000000000
46 000000003000000000000000300000000000
47 000020000001313000002300300200103300
48 302020300233233021103303001330022102
49 000000001000000000000000000010000300
50 200000101300010000000001303000333000
51 300000103303320020000000003010000000
52 322333313213230033033031333330332221
53 200000101301010010000021303000303000
54 100102003022211033003133303320313100
55 313333321320130011003023323213113322
56 000102100000010000000001000000001300
57 213323232201102020200000201021000103
58 301101313312322212002021033211031211
59 203333233300330030003100213230100303
60 301021323312030032000023333111333031
61 202000002001331130002001222102000001
62 100000003101010020000001303020100301
63 300013200302332031001301302130321000
64 003010223012112113013033032310013303
```

Operator Configurations

1	3	5	1	3	6	1	7	6	0	16	16	0	24	11	1	26	8	0	26	9	0
2	2	1	0	2	10	1	3	16	0	8	15	1	14	10	1	19	11	0	0	0	0
3	2	12	1	17	2	0	18	12	0	23	13	1	0	0	0	0	0	0	0	0	0
4	1	9	1	1	12	0	3	9	1	11	15	0	13	14	0	18	15	1	0	0	0
5	6	3	1	12	3	0	13	4	0	16	2	0	18	1	1	18	2	1	24	10	0
6	7	15	1	10	7	0	11	14	1	15	7	1	26	14	0	29	10	0	0	0	0
7	7	1	0	20	11	1	21	10	0	22	7	0	23	5	1	0	0	0	0	0	0
8	3	4	0	4	8	1	5	2	0	5	8	1	21	2	1	22	7	0	26	8	0
9	4	7	1	7	14	0	9	13	0	12	2	1	19	6	0	25	7	1	28	8	0
10	2	10	0	6	15	1	11	5	1	12	12	1	15	8	0	24	12	0	0	0	0
11	7	4	1	9	4	0	14	4	0	20	5	1	0	0	0	0	0	0	0	0	0
12	2	6	0	7	14	1	9	16	0	12	12	1	14	13	0	21	7	1	0	0	0
13	1	12	1	8	10	0	11	3	0	22	11	1	22	16	1	0	0	0	0	0	0
14	6	16	1	8	11	0	13	12	0	23	12	1	0	0	0	0	0	0	0	0	0

15	3	13	0	9	15	0	13	15	1	19	6	0	25	7	1	0	0	0	0	0	0
16	8	13	1	9	16	0	12	13	0	17	11	1	20	16	0	27	13	1	30	15	0
17	9	8	0	12	10	0	18	7	1	18	9	1	18	10	1	20	9	0	29	3	1
18	16	5	1	27	6	1	28	8	0	30	8	1	0	0	0	0	0	0	0	0	0
19	13	12	1	19	12	0	21	4	1	25	4	1	26	3	0	29	4	1	0	0	0
20	12	1	0	13	9	0	17	9	1	25	4	1	27	8	1	30	6	0	0	0	0
21	1	6	1	5	10	0	12	7	0	15	6	1	20	9	1	24	13	0	0	0	0
22	7	5	1	7	6	1	10	6	0	21	8	1	0	0	0	0	0	0	0	0	0
23	2	9	0	4	9	0	6	9	0	6	10	1	15	1	1	26	7	0	0	0	0
24	2	4	1	9	14	1	10	13	0	11	14	1	0	0	0	0	0	0	0	0	0
25	1	11	0	9	12	0	9	14	1	12	4	1	16	9	1	24	11	0	0	0	0
26	3	15	0	5	2	1	14	1	0	15	8	1	21	2	1	0	0	0	0	0	0
27	9	15	1	28	13	0	30	13	1	0	0	0	0	0	0	0	0	0	0	0	0
28	5	4	1	10	1	1	15	4	0	15	7	1	16	7	1	28	2	0	0	0	0
29	6	5	1	6	8	1	8	5	0	17	5	1	22	2	0	25	15	0	28	1	0
30	11	4	1	13	8	0	28	4	0	28	8	1	0	0	0	0	0	0	0	0	0
31	6	6	1	11	6	0	14	16	0	27	8	1	0	0	0	0	0	0	0	0	0
32	1	1	1	13	2	0	14	2	1	26	10	0	0	0	0	0	0	0	0	0	0
33	9	15	1	12	7	0	13	4	0	19	6	0	19	14	0	28	8	1	0	0	0
34	12	6	1	16	3	0	16	12	0	21	6	0	25	5	1	27	15	0	29	6	1
35	6	12	0	10	16	0	12	12	1	23	15	1	26	7	1	29	5	0	0	0	0
36	5	12	0	7	13	1	14	13	1	21	14	0	26	14	0	28	16	1	0	0	0
37	4	2	0	6	5	1	14	4	0	15	5	1	21	3	0	25	5	1	30	5	0
38	3	9	0	3	12	1	19	4	1	24	15	0	0	0	0	0	0	0	0	0	0
39	7	9	0	17	7	1	19	10	1	20	10	0	20	11	1	22	2	1	0	0	0
40	1	3	0	3	7	0	3	11	1	4	11	1	7	2	0	16	3	1	23	15	0
41	8	1	0	8	11	1	12	1	1	13	9	0	28	7	1	0	0	0	0	0	0
42	8	13	0	9	6	1	18	15	1	20	12	0	27	11	1	29	9	0	0	0	0
43	22	6	1	23	13	1	26	4	0	26	13	0	27	15	1	0	0	0	0	0	0
44	5	5	1	6	2	0	15	2	1	21	13	1	0	0	0	0	0	0	0	0	0
45	5	9	1	27	9	0	28	11	1	29	7	1	0	0	0	0	0	0	0	0	0
46	5	11	1	6	1	1	6	12	0	18	12	1	20	2	0	23	4	1	0	0	0
47	12	6	1	14	6	1	17	9	0	25	15	0	30	7	1	30	14	1	0	0	0
48	5	1	1	6	6	0	8	12	0	16	1	1	18	7	1	20	13	0	25	8	0
49	9	2	1	9	15	0	12	2	0	13	3	1	14	10	1	16	7	1	26	12	1
50	12	14	1	13	9	0	14	4	0	15	3	1	23	11	1	0	0	0	0	0	0
51	6	11	1	8	1	0	8	2	1	17	10	0	21	11	1	25	8	1	0	0	0
52	12	10	0	14	1	0	14	14	1	15	8	0	19	6	1	19	11	1	30	7	0
53	5	5	1	8	6	0	11	5	1	15	16	1	0	0	0	0	0	0	0	0	0
54	2	5	1	17	13	0	17	15	1	25	7	0	0	0	0	0	0	0	0	0	0
55	5	14	0	9	8	1	15	14	1	18	11	0	19	15	1	21	7	0	0	0	0
56	1	9	1	3	16	0	10	7	0	15	8	0	17	16	1	20	15	0	22	9	1
57	5	5	1	7	5	0	18	2	0	27	7	1	0	0	0	0	0	0	0	0	0
58	18	12	1	21	3	1	24	2	0	30	4	1	0	0	0	0	0	0	0	0	0
59	9	4	0	11	4	1	20	4	0	22	5	0	29	4	1	0	0	0	0	0	0
60	1	3	1	1	4	0	3	13	1	7	11	1	16	10	0	22	12	0	0	0	0
61	3	8	1	10	3	1	10	4	0	17	5	0	18	3	1	0	0	0	0	0	0
62	3	4	1	9	4	0	15	3	1	19	13	1	26	8	0	0	0	0	0	0	0
63	8	9	1	12	13	0	12	16	0	14	2	0	22	14	0	25	16	1	0	0	0
64	1	10	0	2	3	0	2	16	1	11	11	1	16	6	1	0	0	0	0	0	0