Artificial Intelligence: The Five-Lettered-Word Game

by

Man-Wah M. Leung

Computing Science Department

Submitted in Partial Fulfillment of the Requirements of the

University Undergraduate Fellows Program

1977 - 1978

Approved by

Dr. Udo W. Pooch

April 1978

ABSTRACT

Most of the games studied by Artificial Intelligence researchers are games of perfect information. In this research, however, the author studies a game of imperfect information. This is because often in real life, decisions have to be made prior to the knowledge of the outcomes of events.

Numerous tree generating and searching techniques and learning have been developed and implemented in computer programs that play games; but the problem of searching a game tree is greatly complicated by the introduction of uncertainty. Theoretically, the best strategy can be found by using both the Simplex method for solving Matrix Games and the techniques for searching game trees. But this combination of the two methods make the process so tedious that implementation of the combined method is practically infeasible. Turning away from these conventional methods, the author has developed a heuristic which observes the opponent's strategy and adapts to it by modifying its own strategy.

ACKNOWLEDGEMENTS

This research was carried out under the University Undergraduate Fellows Program of Texas A & M University. The computer programs written for the research were run on the Amdahl 470/V6 computer at the Texas A & M campus, with research funds supported by the Computing Science Department. The paper is edited on WYLBUR, an online editing and remote job entry system supported on the Amdahl.

The author owes his thanks to Dr. Thomas E. Wehrly of the Institute of Statistics and Dr. Anthony P. Lucido of the Computing Science Department for their suggestions and help; to his roommates, Group II of the University Undergraduate Fellows and their advisors for their donation of a list of secret words used by the programs; to his roommate, Doran Brown, for carefully proof-reading the paper; and especially to his advisor, Dr. Udo W. Pooch of the Computing Science Department, for his guidance throughout the course of the research.

TABLE OF CONTENTS

ABSTRACT i

ACKNOWLEDGEMENTS ii

1. INTRODUCTION 1

2. REVIEW 3

3. DESCRIPTION OF THE GAME 8

4. PRELIMINARY RESEARCH 9

5. DESIGN PHILOSOPHY 10
   5.1 Language Consideration 10
   5.2 Perspectives of the Five-Lettered-Word Game 11
   5.3 Objectives 13

6. THE ADAPTABLE HEURISTIC 14

7. IMPLEMENTATION OF THE HEURISTIC 19
   7.1 The First Program 19
   7.2 The Second Program 22
   7.3 The Third Program 23

8. EXPERIMENTS AND RESULTS 26
   8.1 The First Round 26
   8.2 The Second Round 26
   8.3 The Third Round 28
   8.4 The Final Round 29
      8.4.1 Preparation 29
      8.4.2 Prediction 30
      8.4.3 Results 31

9. CONCLUSIONS 36

10. AFTERTHOUGHT 37

BIBLIOGRAPHY 39

APPENDIX A 40

APPENDIX B 45

1. INTRODUCTION[1]

Since the beginning of the computer age in the late nineteen forties, technology has advanced at an almost unbelievable rate. Being in their fourth generation, the fastest computer nowadays can execute an instruction in a matter of nanoseconds. The memory of the computer has evolved from the bulky, expensive and slow vacuum tubes to the compact, relatively cheap and fast integrated circuits and bubble memories (Time, p.59). So much of data processing is handled by the computer that it is hard to imagine what life would be like without the aid of this machine. Yet, the computer has a potential far beyond the performance of mechanical calculations. It can also be programmed to play games, make decisions, solve problems, understand natural languages and perform many other intelligent tasks. These latter aspects of computer programming are largely studied in a particular field of Computer Science - Artificial Intelligence (AI).[2]

Perhaps Game Playing is one of the most impractical areas of study in AI. Yet Game Playing is studied by many AI researchers and the author for several reasons. First, Game Playing is just as challenging as any other area in AI research. Second, the

---

[1] The format and style of this paper mainly follow the guidelines of Kate L. Turabian's A Manual for Writers of Term Papers, Theses, and Dissertations.

[2] In view of the frequent use of some long terms in the paper, the author finds it appropriate to use abbreviations instead of the full spelling of the words so as to make the sentences clearer and less clumsy to read. Abbreviations will be introduced in parentheses in conjunction with the full spellings upon the first occurences of the terms.

rules and goals of games make games well defined problems suitable for computers to solve. Third, the study of games teaches us how to solve complex problems by computers (Hart, p.14). Lastly, the study of Game Playing enables us to understand more about human intelligence.

Chess and Checkers are two of the most popular games being studied in Game Playing. In this research, the author focuses his attention on a less conventional one - the Five-Lettered-Word Game (5-L-W-G). Primarily, this is because he has not learned any list-processing language such as LISP, which is almost exclusively the only type of programming language to be used for writing efficient game playing programs. Besides being a two-person zero-sum game (Jackson, pp.118-9), the 5-L-W-G has the unconventional property of imperfect information (Jackson, p.125). The author also feels more comfortable in choosing this game to study because of his familiarity with it.

2. REVIEW

Since most of the terminologies used in the study of Game Playing are quite self-explanatory, they will not be explained here. The reader is encouraged to refer to the section on Strategy in chapter 4 of Philip C. Jackson's Introduction to Artificial Intelligence in case he feels uncomfortable with the terms used in this paper.

In reviewing the literature on Game Playing, it is found that most of the methods used in Game Playing programs are related to the searching of game trees. Let us consider the example in Figure 1. The initial state of this game is the root node of the tree, that is, node 'a'. The leaf nodes represent the final states of the game, and the numbers below them represent the "payoffs" (Johnson, pp.134-8) for player A, or in other words, the benifit that A will receive from B at the end of the game. Suppose the game has proceeded to node 'd', then A should make the move to node 'i' since its payoff, 9, is greater than that of node 'h'. Similarly, if the game is at nodes 'e' or 'f', then A should make his move to nodes 'l' or 'm' respectively. Now, since B is trying to minimize A's payoff, if the game is at node 'b' or 'c', he should make his move to 'd' or 'g' respectively. Applying the same strategy for A as before, he should make his first move to node 'b'. Therefore, if both players are being rational, the path of the moves would be (a, b, d, i), which results in a payoff of 9. The methods of alternately maximizing the minimum payoff by A and minimizing the

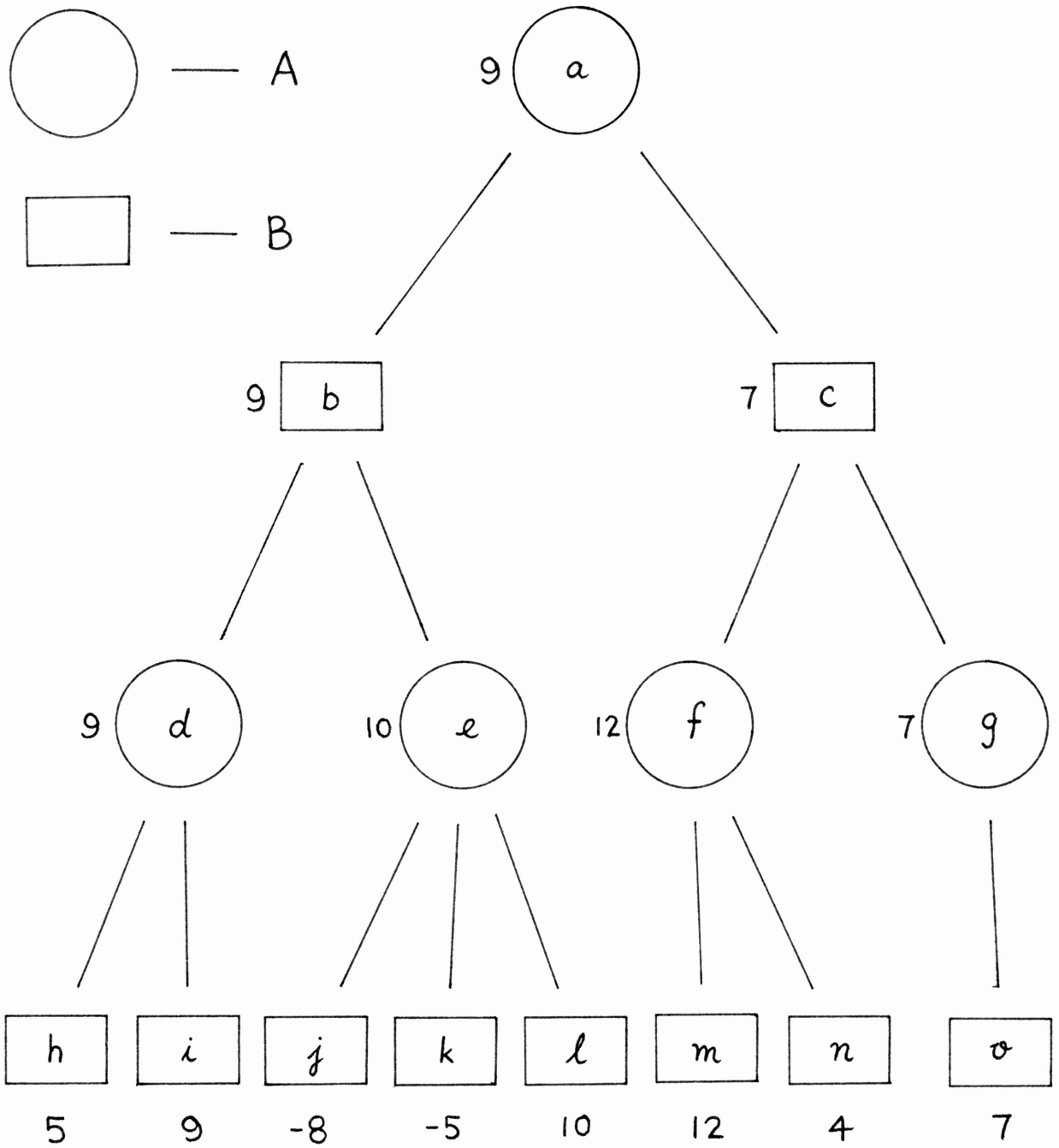

Figure 1: Example of a game tree.

maximum payoff by B are called maximin and minimax respectively.

In the above example we were able to trace from the leaf nodes back to the root node, since the game tree was given explicitly. In a game of moderate complexity, however, the game tree is much larger. For instance, it is estimated that the game tree of Checkers has an average depth of 100 and an average branching factor of 6 (Jackson, p.125). The number of possible plays is essentially infinite even for today's largest computers. Since it is not practical, and in many cases impossible, to search the entire game tree, techniques have been developed for generating (Jackson, p.134) and searching game trees to reduce the searching to a minimum number of steps. An example of a searching technique is the "Alpha-Beta Technique" (Slagle, pp.15-20). However, the searching of the entire game tree is not manageable even with these techniques. Consequently, other techniques have been developed to search only a part of the game tree. These include the "Depth-first Search", "Breadth-first Search", "Hill-Climbing", "Best-First Search" and "Branch-and-Bound" techniques (Winston, pp.89-102).

In some games such as Poker or Bridge, a player does not know every state of the game, and cannot predict the results of his moves. In their book Games and Programs, Singleton and Tyndall give a good example of games of imperfect information:

> "Example 2 (Tricks) Player A and Player B have three cards each, numbered 1,2,3. Simultaneously each selects one card and shows it. The higher card wins the trick and captures the lower card, or, in case of a tie, both cards go into the next trick. The players repeat for a second trick with their two remaining cards apiece. Finally, each player shows his single remaining card, but now if the cards tie they go to the winner (if any) of the preceding trick (as in Casino). The player who

takes the most cards wins the game. If no winner is determined in a play, it is a draw." (Singleton and Tyndall, p.29)

Let us consider the representation of a move in a game in which the two players make their moves at the same time without the knowledge of each other's moves. Figure 2 represents a Matrix Game for players A and B. If A chooses strategy A-I, then the minimum payoff he receives is 2, whereas if he chooses A-II, the minimum payoff he receives is 4. Therefore in order to maximize his minimum payoff, he should choose strategy A-II regardless of what strategy B uses. Similarly, in order to minimize his maximum payoff to A, B should choose strategy B-I regardless of what A's strategy is. The use of "pure strategies" (Luce and Raiffa, p.51) A-II and B-I by A and B respectively results in a "saddle point" (Singleton and Tyndall, p.47) which corresponds to the element (2,1) of the matrix. A deviation of either player's pure strategy will result in a suffering of the less payoff he receives (in the case of A) or the more payoff he gives (in the case of B).

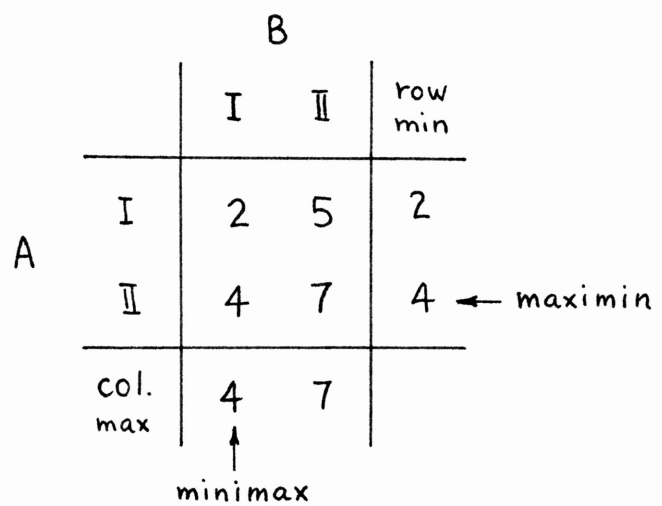

Figure 2: Example of a Matrix Game.

Not all Matrix Games have saddle points. The optimal strategies for the players in such a game can be found by using linear programming techniques. These optimal strategies are called "mixed strategies" (Singleton and Tyndall, p.54) since they are probabilistic combinations of pure strategies.

It always seems to be incomplete to discuss game playing programs without mentioning Samuel's Checker Program (Samuel, pp.71-105), the one so successfully written that it can beat human checker players who are at the masters level. Like most other programs, the Checker program generates and searches a partial game tree at each move. In any move, since most root nodes of the partial game tree do not represent the end of the game, the payoffs of these root nodes are unknown. In order to apply the minimax procedure, a polynomial, whose parameters represent the intermediate goals (e.g. piece advantage) of the game, is used as a static evaluation function (Samuel, p.75) for the board position.

The most important features of Samuel's Checker Program that account for its success are "rote learning" and "learning by generalizations" (Samuel, p.82). The learning techniques involve: the dynamic selection of the parameters from a set of possible terms and the determination of the coefficients by the program itself. Since these techniques are rather complicated and long, they will not be further explained. The interested readers are advised to read the original paper written by Samuel.

3. DESCRIPTION OF THE GAME

The Five-Lettered-Word Game (5-L-W-G), also manufactured as a game kit under the name JOTTO, is a word game similar to Master Mind and many other word games. Since this game has several variations, its rules will be stated below.

The 5-L-W-G is played by two players. At the start of the game, each player thinks of a non-capitalized, five-lettered English word as his secret word. Then they alternately take turns to guess the opponents' secret word by using at each turn any five-lettered word that satisfies the requirements of a secret word. After a player has announced his guess word and before the opponent takes his turn, the opponent has to tell the player the number of alphabets in the current guess word that match his (opponent's) secret word, regardless of the positions of the matched alphabets. The rules for computing the Number of Matched Alphabets (NMA) is mathematically defined as follows: Denote an alphabet by 'k'. Let G(k) and S(k) be the number of 'k's in the guess word and the secret word respectively. Then

$$NMA = \sum_{k=a}^{z} \text{FLOOR}\,(G(k), S(k))$$

where Floor(p,q) represents the minimum of p and q.

An example of NMA's of different guess words for the secret word TUTOR is given in Table 1.

guess word	APPLY	ACUTE	WORTH	TOOTH	OTTER	TUTOR	TROUT
NMA	0	2	3	3	4	5	5

Table 1: Examples of NMA's for the secret word TUTOR

4. PRELIMINARY RESEARCH

Prior to this research, the author has written a paper on the same topic for his course Engr. 385: Problems for Co-op. Students. In the previous research, a FORTRAN program was written, which attempted to mimic the reasoning in the author's strategy of playing the 5-L-W-G. The algorithm of the program was not even close to the flexible strategy; as a result, the program took, on the average, of almost twice as many guesses as that needed by a good player to finish the game. His dissatisfaction with the results has motivated the author to pursue the research further.

5. DESIGN PHILOSOPHY

The failure of the previous research has led the author to consider the solution approach of the problem seriously.

5.1 Language Consideration

FORTRAN, being a scientific oriented language, is very poor in the manipulation of character strings. Therefore, it was decided that a better language (in terms of character processing) should be used. Had the author learned any symbol manipulation language prior to the beginning of the research, that language would have been chosen. The only alternatives left were PL/I and assembly language. Although PL/I is a fairly powerful language, it was discarded because of its expensiveness. The System/370 assembly language was finally chosen because of the above reasons and its efficiency, and with the theory that since assembly language is in the lowest level except for machine language, it can accomplish anything which is possible in any higher level language.

Owing to the high costs of interactive online programming facilities available, the author was forced to use batch processing. Therefore even though the 5-L-W-G is a two-person game, the programs written will achieve only the goal of minimizing the number of moves it takes to match all the five alphabets in the opponent's secret word (fed into the program as input data), but not the goal of choosing the most difficult secret words for the opponent to guess. Nevertheless, it is

believed that some kind of reversed strategies can be applied to achieve the latter goal since both goals are contradictory to each other.

Since we are dealing with only one aspect of the problem, and in order to reduce the ambiguity of the words "player" and "opponent", we shall borrow two terms from Slagle: "Min" for the name of the program, who is trying to minimize the number of moves to finish the game; and "Max" for its opponent, namely the programmer, who is trying to maximize Min's moves.

5.2. Perspectives of the 5-L-W-G

As pointed out by Jackson, a game of strategy requires that the rules of the game be finitely describable and that the number of alternatives for each move be finite. Although the set of five-lettered English words is finite, to enumerate all its elements is very cumbersome. In order to make it practical, the author has copied a list of 2139 five-lettered words from a junior dictionary.[3] The list of words is given in Appendix A. As a modification of the rules, the secret words and guess words must be chosen from the list. Although the list is only a small subset of all the five-lettered words, it nevertheless represents words in the whole spectrum; therefore the AI techniques developed hereafter should also be applicable to the superset.

---

[3] Due to the long process of preparation of the list, the words have not been double checked for correct spelling. It has also been discovered that the word SPEND is duplicated in the list. All in all, these errors should not affect the validity of the AI techniques involved.

One very important property in the definition of NMA has been discovered. This symmetric property is quite hidden; yet when pointed out, it becomes obvious. By definition, if word 'A' matches word 'B' by m alphabets, the converse also holds.

Since the 5-L-W-G is a two-person, zero-sum game of imperfect imformation, the Matrix Game theory may be considered. For the list of approximately two thousand words, ~~we can~~ a table of the NMA's for each pair of words can be constructed. This is shown in Table 2. The matrix in this table is denoted by T, and only Min's first move is considered. The payoff function for Min can be defined as:

$$P(i,j) = \begin{cases} 0 \text{ if } T(i,j)=5, \text{ otherwise} \\ \\ \text{number of elements in row } i \text{ of } T \\ \text{which have the value } T(i,j). \end{cases}$$

where the guess word and secret word are the i'th and j'th words in the list.

	ABACK	ABAFT	ABASE	...	ZEBRA
ABACK	5	3	3	...	2
ABAFT		5	3	...	2
ABASE			5	...	3
.					.
.					.
.					.
ZEBRA					5

Table 2: Table of NMA's

After Min has announced his guess word and Max has told him the NMA for that word, Min can eliminate all the words in the list that does not match his guess word. Thus he can reduce the set of words to a smaller one. The number of elements in this reduced set corresponds to the element T(i,j) of the table T.

Using the Simplex method, Min's mixed strategy can then be calculated. However, since about 2,000,000 (=2,000x2,000/2) entries of the table have to be calculated, and since the matrix to be solved by the Simplex method is large, the solution is quite tedious to obtain. It becomes even more complicated when we consider the whole game instead of only the first move, because the application of the strategies which are optimal to each single guess may not be optimal for the whole game. This, and the fact that the 5-L-W-G is a game of imperfect information, makes the AI techniques for generating and searching games trees inapplicable. In view of this, the author his resorted to finding an approach different from searching game trees.

5.3 Objectives

The Method of Elimination (i.e., compressing the list of words after each guess by eliminating all the words that do not qualify to be the secret word) is a fairly effective algorithm for playing the 5-L-W-G, especially if the list of words has a high entropy (Watanabe) of alphabets, that is, if the alphabets are quite randomly distributed in the list. This brute force approach, however, is not by any means an AI technique.

Consequently, the objective of the research is to include some kind of learning in the computer program.

6. THE ADAPTABLE HEURISTIC

From the author's experience with different people's strategies of playing the 5-L-W-G, he has noticed that in general, players do not choose their secret words casually, but with some criteria in their minds. Usually, a player feels that certain kinds of secret words are more difficult to guess than other kinds. Examples include words that contain duplicate letters and those that contain a rarely occuring letter such as 'z'. This observation led the author to make the following hypothesis:

If Min finds out that Max predominantly chooses his secret words with certain characteristics, then Min should be able to improve his performance (finish the game with fewer guess words used) by choosing guess words that have the same characteristics.

To begin with, the vague description of the characteristics of words is first changed into precise definitions which make the characteristics representable in the computer.

The words are Classified into five Classes of sixteen Types as shown in Table 3. Except for the first Class, all the Types in any one Class are mutually exclusive. For example, if a word is of Type 11 (1 vowel), then it cannot be of Type 12 (2 vowels). These sixteen Types cam be represented by a vector of binary digits, and each word is associated with a bit vector, whose bits

Class		Type	e. g.
I rare occuring alphabets	0	contains 'J'	JUROR
	1	contains 'Q'	QUEUE
	2	contains 'V'	KNAVE
	3	contains 'X'	SIXTY
	4	contains 'Z'	OZONE
	5	contains none of the above	OFTEN
II multiple letters	6	double letters	FLOOD
	7	double-double letters	QUEUE
	8	triple letters	DADDY
	9	otherwise	SLING
III number of vowels	10	no vowels	LYMPH
	11	1 vowel	LIGHT
	12	2 vowels	WHEEL
	13	more than 2 vowels	ADIOS
IV starting alphabet	14	begins with a vowel	EAGLE
	15	otherwise	CHASE

Table 3: Classification of words into classes

are called the Property Bits of the word since they represent the properties of the word. For example, the bit vector of the word ACUTE is '0000010001000110'.

The adaptable heuristic introduced below consists of two functions whose parameters are modified at the end of each game so as to adapt themselves to the strategy of Max and hence predict the properties of the next secret word.

The property bits of the last secret word is broken into four vectors corresponding to the four Classes. The Class Vector of Types for the i'th Class of the last secret word is defined as:

$$C(i) = (t(i,1), t(i,2), \ldots, t(i,n))$$

where n = number of Types in Class i, and

t(i,k) = bit value of the k'th Type in Class i of the last secret word;

t(i,j) = 1 for some $1 \leq j \leq n$,

t(i,k) = 0 for all $k \neq j$.

The Estimate Probability Vector for the i'th Class is defined as:

$$E(i) = (e(i,1), e(i,2), e(i,3), \ldots, e(i,n))$$

where n = number of Types in Class i.

e(i,k) = Estimated Probability of t(i,k) for the next secret word being 1; and such that

$0 \leq e(i,k) \leq 1$,

and $\sum_{k=1}^{n} e(i,k) = 1$

Let the adjustment function for the vector E be a linear function such that

$e(i,j) = 1$ and $t(i,j) = 1 \Rightarrow e'(i,j) = 1$,

$e(i,j) = 0$ and $t(i,j) = 1 \Rightarrow e'(i,j) = \alpha$,

where α is a constant such that $0 \leq \alpha \leq 1$.

This implies that the greater α is, the more adjustment an Estimated Probability receives.

The Types corresponding to the properties of the last secret word are rewarded, that is,

for t(i,j) = 1,

$e'(i,j) = e(i,j) + \alpha (1 - e(i,j))$

All the other Types are punished, that is,

for all k such that t(i,k) = 0,

$$e'(i,k) = e(i,k) \times (1 - \alpha)$$

where e'(i,j) and e'(i,k) are the new Estimated Probabilities.

Now, the Distributional Probability Vector of Class i is defined as:

$$D(i) = (d(i,1), d(i,2), d(i,3), \ldots, d(i,n))$$

where

$$d(i,j) = \frac{\text{number of words having the Property Type}}{\text{total number of words in the word bank}}$$

The Best Normalized Estimate e" can now be defined as:

$$e'' = \max_{i} \left(\frac{e(i,j)}{d(i,j)} \times t(i,j) \right)$$

where d(i,j) is the Distributional Probability;

and the Best Class, C", is defined as the Class containing the Best Normalized Estimate.

The second adjustment function is very similar to the first one. Prior to each game, each Class C(i) is assigned a Weight w(i) such that

$$\text{for all } i, \quad 0 \leq w(i) \leq 1$$

$$\sum_{i=1}^{4} w(i) = 1$$

The Weight Vector is a vector composed of the Weights of the four Classes.

$$W = (w(1), w(2), w(3), w(4))$$

The Weight of a Class represents the estimation of the probability of Max's secret word being in that Class.

After the game, the Best Class C" is rewarded by

$$w'(i) = w'(i) + \beta (1 - w(i))$$

where $0 \leq \beta \leq 1$, and C(i) corresponds to C".

All the other Classes are punished similarly to the punishment of Types.

$$w'(j) = w(j)\ (1 - \beta\ ), \quad \text{for all Non-Best Class } C(j)$$

Note that the degree of adaptation can be tuned by changing the values of α and β.

$\alpha = 0$ => no adjustment of Estimates made;

$\alpha = 1$ => Estimates depend completely on the last secret word (because e'(i,j) = t'(i,j));

$\beta = 0$ => no reword or punishment of Classes;

$\beta = 1$ => Classes will be fully rewarded or punished.

Lastly, the Fitness of Type j in Class i is defined to be the joint probability (Wonnacott and Wonnacott, p.88) of the corresponding Estimate and Weight.

$$f(i,j) = e(i,j) \times w(i)$$

and the Fitness Vector is defined as:

$$F = (f(1,1),\ f(1,2),\ \ldots\ ,\ f(1,6),$$
$$f(2,1),\ f(2,2),\ \ldots\ ,\ f(2,4),$$
$$f(3,1),\ f(3,2),\ \ldots\ ,\ f(3,4),$$
$$f(4,1),\ f(4,2))$$

Note that

$$\sum_{i=1}^{4} \sum_{j=1}^{n(i)} f(i,j) = 1$$

Hence the Fitness vector forms the probability distribution for the events of a sample space.

7. IMPLEMENTATION OF THE HEURISTIC

All together, three programs were written for testing and comparison. The Method of Elimination was used in all the three programs.

7.1 The First Program

The first program, PGM0, was written to establish the data structure and subroutines used by the later programs. More important was that it served as a control experiment with which the later programs could be compared.

Since words are frequently eliminated from the list, a linked list is used for higher efficiency. The structure of the list is shown in Figure 3. Each node in the list consists of three fields. The first field (bytes 0-3) contains the link to the next cell. The second field (bytes 4-8) contains the five-lettered word, and the third field (bytes 9-10) contains the word's Property Bits. The list is circularly linked because it is not necessary to have a head node.

Figure 4 shows the Process of Elimination. Starting from a node, its word is compared with the last guess word. If it does not match the guess word with the same NMA as that matched by the secret word, then the current node can be eliminated. This is done by setting the link of the last cell equal to the link of the current cell. Otherwise, no changes will be made so that the node still remains in the linked list. The next node is then

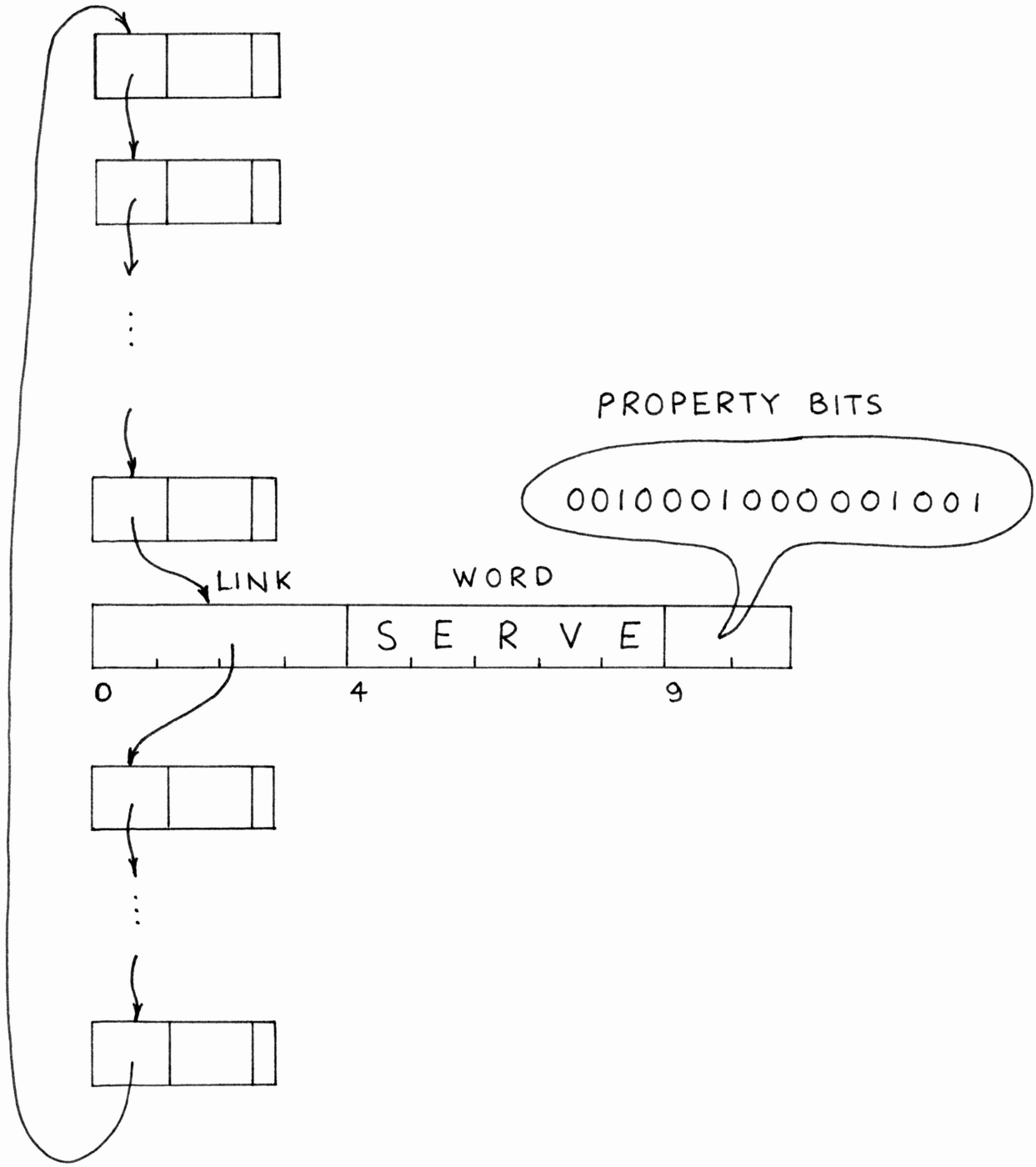

Figure 3: Circularly linked list of the five-lettered words.

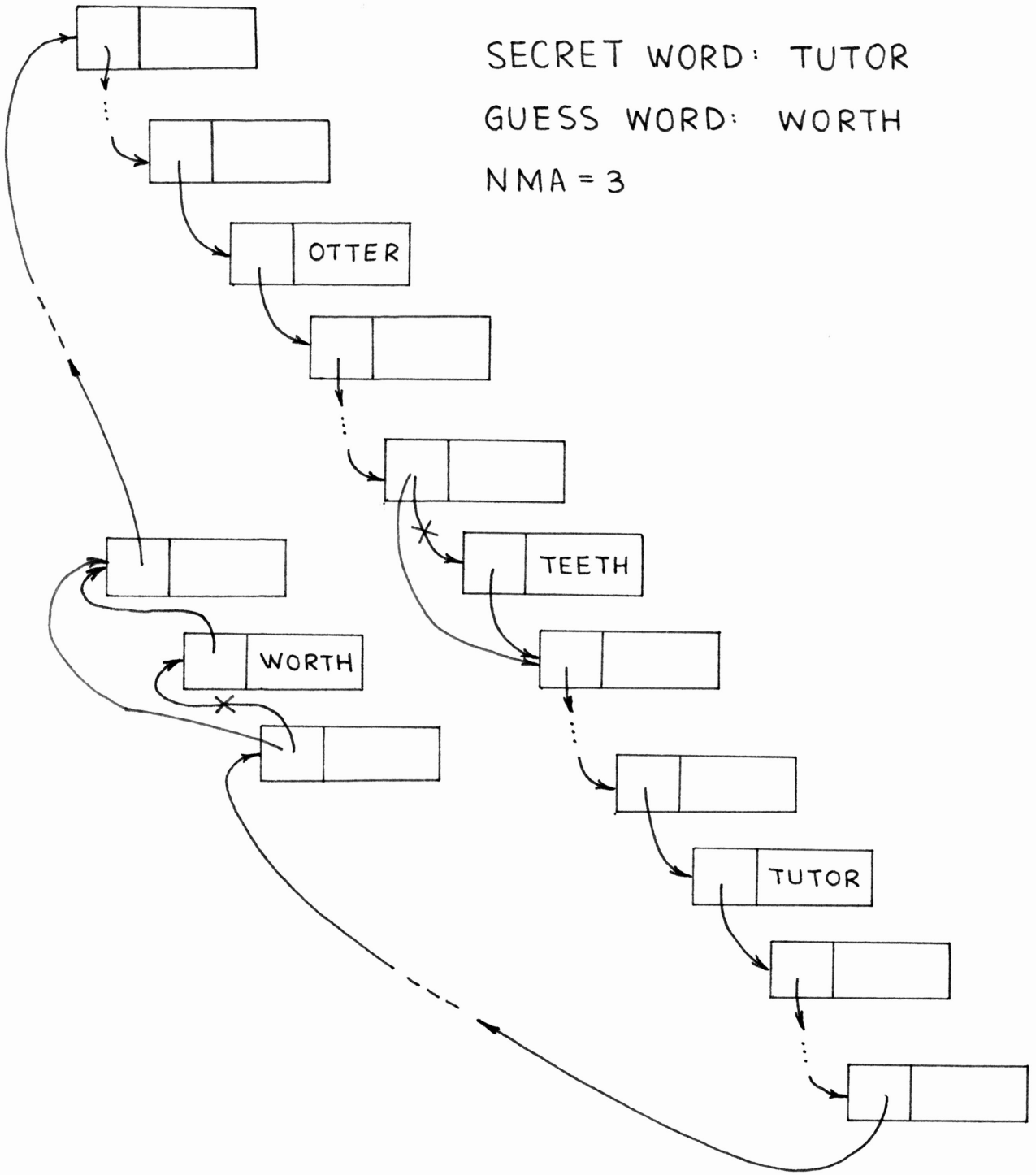

Figure 4: Process of Elimination.

examined, and the process continues until the examination has completed the whole cycle of nodes. The new linked list then contains the reduced set of the words.

The algorithm for PGM0 is given as follows:

1. Randomly choose a guess word from the current list.
2. Announce the guess word and compute the NMA for this guess word.
3. If NMA = 5, then EXIT (game is finished).
4. Perform the Process of Elimination.
5. Go to Step 1.

One point needs to be noted. When a word is deleted from the list, it will never be chosen as a guess word for the subsequent guesses in the same game. This also holds true for the second and the third program.

7.2 The Second Program

The second program, PGM1, incorporates the adaptable heuristic. Arbitrary values could have been assigned to the Weight Vector of the Classes and the Estimate Probability Vector, since the program was supposed to adjust these values by playing the game. However, for comparison purposes, the initial conditions for all the three programs should be the same. Hence to be fair, all the Weights were set to 0.25 and the Estimate Probability Vector was set equal to the Distributional Probability Vector. The same was also done for the third program.

The algorithm for PGM1 is given as follows:

1. Calculate the Fitness for each Type.
2. Randomly choose a starting node in the linked list.
3. Choose a Type by a random function weighted by the Fitness of each Type. Call this Type 't'.
4. Start from the node chosen in step 2.
5. Examine the Property Bit of the Type chosen in Step 3 for this node.
6. If the Property Bit is "on", or if the examination is done for a complete cycle of the list, then choose the current word as the guess word and go to Step 9.
7. Proceed to the next node.
8. Go to Step 5.
9. Announce the guess word and compute NMA for the guess word.
10. If NMA = 5, then go to Step 13.
11. Perform the Process of Elimation.
12. Go to Step 2.
13. Reward and punish the Classes and Types, and EXIT.

7.3 The Third Program

The Reduction Factor is defined as the ratio of the number of words in the list just before the Process of Elimination.

$$RF = \frac{N'}{N}$$

Since the elimination of disqualified words results in the increasing orderedness of the remaining set of words, the entropy

decreases during the first few guesses. Hence the Reduction Factor generally decreases as the number of guesses made.

To compensate for this fact, the author modified PGM1 to become PGM2, the third program. When the List has shrunk down to sixty words or less, the program changes from the adaptable heuristic to a minimax procedure, which is essentially a one-level Breadth-First Search. Since the minimax procedure is quite exhaustive, if PGM2 encounters a word that has a reduction factor of at least 3, then it will quit the procedure and use the current examined word as the next guess word.

A Listing of PGM2 is given in Appendix B. Because the DC instructions of the linked list is too long, that part of the program is not shown. An example of a game played by this program is shown in Figure 5.

```
THE SECRET WORD IS:  TOWER
GUESS-WORD:  DEVIL
NMA = 1
RANGE = 00904
GUESS-WORD:  BAIRN
NMA = 1
RANGE = 00390
GUESS-WORD:  SHAWL
NMA = 1
RANGE = 00136
GUESS-WORD:  FLUNG
NMA = 0
RANGE = 00065
GUESS-WORD:  HITCH
NMA = 1
RANGE = 00024
ENTER SUBROUTINE SIXTY
NEXT GUESS-WORD HAS REDUCTION FACTOR OF AT LEAST 3
GUESS-WORD:  WROTE
YOU GOT IT!
THE SECRET WORD IS:  QUEUE
GUESS-WORD:  CHIVE
NMA = 1
RANGE = 00963
GUESS-WORD:  CURRY
NMA = 1
RANGE = 00459
GUESS-WORD:  ANGER
NMA = 1
RANGE = 00128
GUESS-WORD:  SEPOY
NMA = 1
RANGE = 00064
GUESS-WORD:  FIRST
NMA = 0
RANGE = 00018
ENTER SUBROUTINE SIXTY
NEXT GUESS-WORD HAS REDUCTION FACTOR OF AT LEAST 3
GUESS-WORD:  CACAO
NMA = 0
RANGE = 00005
ENTER SUBROUTINE SIXTY
GUESS-WORD:  EXUDE
NMA = 3
RANGE = 00001
GUESS-WORD:  QUEUE
YOU GOT IT!
```

Figure 5: *Sample of 2 games played by PGM2.*

8. EXPERIMENTS AND RESULTS

The three programs were written and tested with five sets of data (secret words). Both PGM1 and PGM2 used 0.25 for the value of both α and β for all the runs.

8.1 The First Round

The first set of data, which contained twenty words, was generated randomly by PGM0. A list of the words is shown in Figure 6. The purposes of this set of data were:

1. To fnd out the cost of running the programs so that the author could make an estimation of the total cost and hence make use of the budget wisely.
2. to find out generally the number of moves PGM0 needs to make to solve the problems.
3. To compare the performance of pgm0 and PGM1.

This set of data was intended to be only a preview of the following experiments. Therefore PGM2 was not run on this set.

It took PGM0 6.7 moves to finish the game. PGM1's performance was quite close. This was anticipated because PGM1 should not be able to deduce any significant strategy of Max from the set of randomly generated secret words.

8.2 The Second Round

The second set of secret words were obtained from Group II

DETER
PHOTO
WROTH
FIBRE
JIMMY
VISOR
SUGAR
* PITCH
HORSE
EATEN
* SALVE
TWANG
* STAGE
* CASTE
GRAIN
ADOPT
PROOF
DECOY
SNIFF
KNEEL

Figure 6: Randomly generated secret words.

ABOUT
* OFTEN
CAMEO
* STALL
* SPARE
ACORN
FJORD
* KIOSK
DEIGN
* ANKLE
THIGH
ANKLE
SHRUG
* CRIME
STUDY
YOUNG
MOOSE
PIANO
TUTOR
* SWALE
* WORST
* TUTOR
OZONE
* SLING
* RUSTY
* TERSE
WEDGE
IMAGE
* FIERY
BATHE

Figure 7: Secret words chosen by the Fellows.

of the University Undergraduate Fellows and their advisors because a good program should not be restricted to playing with only one player; instead it should be able to adjust itself to changing strategies, which might very well be Max's strategy anyway. This set of data was intended to be a cruel test for PGM1 and PGM2.

A total of fourteen people contributed their secret words. Those words that were not in the word bank were discarded, which left thirty words in the set. A list of these words is shown in Figure 7.

The mean of the number of moves for PGM0, PGM1, and PGM2 were found to be 6.73, 6.4, and 6.6 respectively. The results were a little surprising because PGM1 and PGM2 were not expected to do so much better than PGM0.

8.3 The Third Round

To confirm the above results, it was decided that the sample taken was too small to draw firm conclusions. Since the author had also collected a list of secret words when he played the game with his three roommates, he decided to use their words for the next round of tests. Similar to the Fellows' data, the roommates' data were also screened by the list of words in the word bank. Twenty-three words remained to be tested. A list of these words is shown in Figure 8.

This time the mean number of moves were 6.65, 7.09 and 6.96, much to the surprise and disappointment of the author. But upon

	TOWER
	QUEUE
	CHARM
	KNOCK
	QUIRK
	BENCH
	WHICH
*	EAGLE
	PEACE
	CARAT
	XYLEM
	JUDGE
*	ENJOY
	DADDY
*	TREAD
	SMEAR
*	DOWEL
	TOXIN
	SIXTY
	CLOSE
*	EXCEL
*	KNIFE
	RHYME

Figure 8: Secret words chosen by the roommates.

detailed investigation of the games played, the author thought that too much luck was involved. Therefore he decided to take more tests.

8.4 The Final Round

8.4.1 Preparation

At this point several things were done. First, the fourth set of data was created by selectively choosing from the first three sets of data those words that were guessed by at least one of the programs with either less than five or more than nine guesses. This action seemed to favour reducing the element Luck,

for all three programs had to prove whether they were really so good or so bad at guessing these words. This new set of words also contained twenty-three words, which are marked by *'s in Figures 6 to 8.

Second, the fifth set of data was created to give PGM1 and PGM2 a chance to prove their adaptation. To eliminate discrimination, the words THIGH and RHYME were chosen to be the secret words because all the three programs had taken six guesses to guess THIGH and seven guesses to guess RHYME in the earlier rounds of tests. Both words were used twelve times as secret words to make the total number of secret words for all four sets (not including the first set) a nice round number = 100.

Lastly the seat for the random number generator was changed from 8193 to 847981089 so that the second and third set of data could be retested.

8.4.2 Prediction

Before the final production runs were made, the results of the experiments were predicted. The data were designed to be as fair to all the three programs as possible: The Fellows' data favoured PGM0 because it consisted of a very well mixed strategy, which in effect, was close to random selection of secret words; the roommates' data represented a normal mixed strategy which had some consistent criteria in choosing secret words; the selected set of repeating secret words favoured PGM1 and PGM2.

The average performance of PGM0 was expected to be constant

over all the four sets of data, whereas PGM1 and PGM2 were expected to improve their performance from the Fellows' data to the roommates' data to the repeated data (but not the selected data). Since PGM2 used the minimax procedure for part of the game, it was expected to be slightly better than PGM1.

The author was also aware that chance would play an important role in the game, therefore unexpected results might be interpreted as victims of Luck. A more important point was that the adaptable heuristic was not directly related to the method of playing the game. For instance, if PGM1 found out that Max was using a lot of double lettered words and the secret word of this game was DITTO, he could only benefit from his knowledge by using a guess word that contained two T's, but if he used other double lettered words (e.g., CLOCK) as his guess word, then the result might be worse than choosng randomly. The knowledge gained by PGM1 would actually come into use when the list of words had been narrowed down to a small number of words that had different properties. Thus, it was hoped that overall, PGM1 and PGM2 would improve their performance despite the fact that the method of playing the game and the knowledge acquired from playing previous games were incompatible.

8.4.3 Results

The last four sets of data were bound into a single batch and run with the three programs. The results are shown in Figure 9. As in the earlier runs, PGM1 and PGM2 "mischievously"

SECRET-WORD	PGM0		PGM1		PGM2	
ABOUT		8	*	5	*	5
OFTEN		8		7	*	6
CAMEO	*	6		8		7
STALL		9		9	*	7
SPARE	*	5		7		7
ACORN		9		5	*	4
FJORD	*	7	*	7	*	7
KIOSK	*	6		8	*	6
DEIGN	*	5		8		8
ANKLE		6	*	5		9
THIGH		6	*	5	*	5
ANKLE		10		7	*	4
SHRUG	*	6		7	*	6
CRIME		7	*	5		7
STUDY		8		6	*	4
YOUNG	*	5		7	*	5
MOOSE		7	*	6		7
PIANO	*	6		10		8
TUTOR		7		7	*	6
SWALE		10	*	4		13
WORST		8	*	7		10
TUTOR	*	7	*	7	*	7
OZONE	*	8		9	*	8
SLING	*	6	*	6		7
RUSTY	*	7	*	7		8
TERSE		7		8	*	6
WEDGE	*	6	*	6	*	6
IMAGE		7		8	*	4
FIERY		6	*	5		7
BATHE		8		7	*	6
TOWER	*	5		9	*	5
QUEUE	*	7	*	7	*	7
CHARM	*	5		6		6
KNOCK	*	3		7		6
QUIRK		7		6	*	3
BENCH		6	*	4		6
WHICH		6		5	*	4
EAGLE	*	8		9		9
PEACE	*	6		7		7
CARAT	*	6		7		10
XYLEM		8	*	7		8
JUDGE		10		9	*	6
ENJOY	*	6		7		8
DADDY	*	5		6		7
TREAD		7	*	6	*	6
SMEAR		11		9	*	7
DOWEL	*	6	*	6		8
SIXTY		10		9	*	8

Figure 9: Results of all the data run with the second seat of ramdom numbers.

TOXIN	*	7		9			8
CLOSE		7	*	6	*		6
EXCEL		8		6	*		5
KNIFE	*	6		7	*		6
RHYME	*	7		8	*		7
PITCH	*	5		7			8
SALVE		10	*	8	*		8
STAGE		7	*	6			14
CASTE	*	5		11			6
OFTEN		8		8	*		6
STALL		7		7	*		5
SPARE	*	3		6			5
KIOSK		7	*	6			8
ANKLE	*	8		9			9
CRIME	*	7		9			10
SWALE		11	*	7	*		7
WORST	*	4		8			9
TUTOR	*	5		8			7
SLING	*	4		7			8
RUSTY	*	8	*	8	*		8
TERSE		6		7	*		5
FIERY	*	6		9			8
EAGLE		9	*	4			9
ENJOY		9		10	*		7
TREAD		7		7	*		6
DOWEL		9		9	*		6
EXCEL		8		10	*		7
KNIFE		7		9	*		6
THIGH	*	5		6			8
THIGH	*	4		9			7
THIGH		9	*	4			9
THIGH	*	6		8	*		6
THIGH		8	*	6			9
THIGH		8	*	5			8
THIGH		8	*	5			6
THIGH		7		6	*		5
THIGH		7	*	5			6
THIGH		6	*	5			8
THIGH		9	*	6			9
THIGH		8	*	6	*		6
RHYME	*	6		8			7
RHYME		10	*	5			7
RHYME	*	7		8	*		7
RHYME		9	*	6			8
RHYME		7	*	4			6
RHYME		9	*	3			8
RHYME		7		6	*		5
RHYME		9	*	4			5
RHYME		7	*	3			4
RHYME		6	*	4	*		4

Figure 9 (continued)

RHYME	8	6	* 5
RHYME	7	8	* 2
TOTAL	705	681	680
MEAN	7.050	6.810	6.800
SIGMA	1.635	1.668	1.842
# OF "*"	41	40	49

NUMBER OF GUESSES TO FINISH A GAME

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
PGM0	0	0	2	3	10	22	27	18	10	6	2	0	0	0	0
PGM1	0	0	2	7	12	22	24	15	14	3	1	0	0	0	0
PGM2	0	1	1	7	12	24	22	20	8	3	0	0	1	1	0

Figure 9 (continued)

performed poorer on the roommates' data than on the Fellows' data. They also showed a poor performance on the selected data. However, both of them showed significant improvement on the set repeated secret words, while PGM0 remained the same. This is clearly shown in Figure 10. For each game, an '*' was marked for the program that finished in the least number of guesses. The total number of *'s for each program were counted as the number of games won. PGM0 and PGM1 came very close, but PGM2 was far ahead of the others. The average number of guesses taken by each was also computed, and both PGM1 and PGM2 were found to be better than PGM0 by 0.24 and 0.25 words respectively.

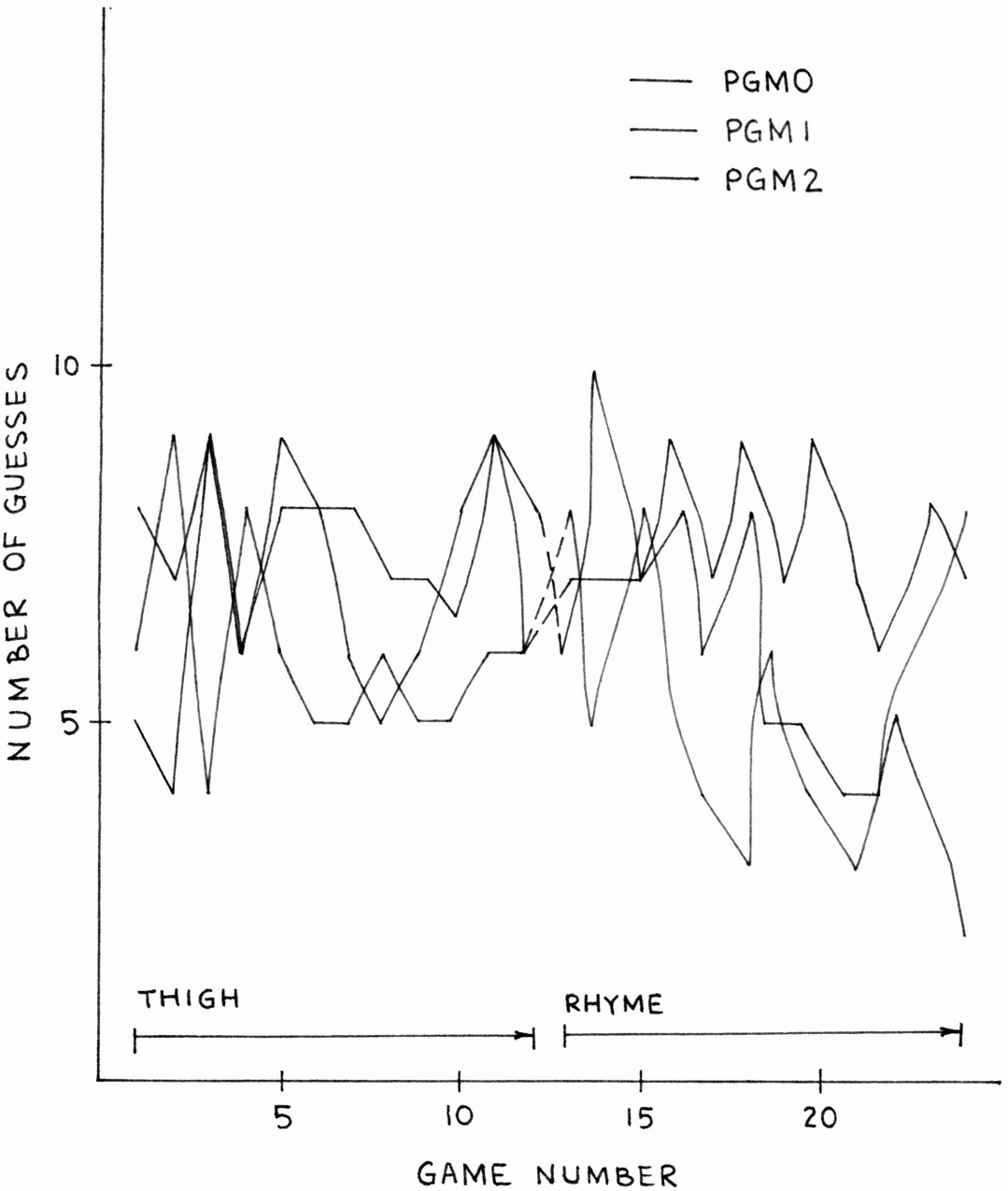

Figure 10: Performance of the three programs on the set of repeated secret words.

9. CONCLUSIONS

The adaptable heuristic implemented was actually very primitive and over-simplified. The philosophy behind this was that efficiency was considered more important than effectiveness. Thus, considering the large number of samples taken and the incompatibility between learning and the method of playing the game, the average performance of PGM1 and PGM2 are interpreted as significantly better than that of PGM0. In spite of the plurality of games won by PGM2 over PGM1, the means show that there is actually no difference between the two programs. This further proves that in the final stages of a game, the "intuitive" approach is no worse than the minimax method. Therefore this heuristic is very adaptable to processes that change slowly with time.

10. AFTERTHOUGHT

When the author started writing this paper, a mistake was discovered in the programs. It may be remembered that the first Class of Types, the Class of Rare Alphabets, contains events that are not mutually exclusive. However, this is overlooked in the program logic. The rewarding scheme assumes that at most one Type of Rare Alphabets is rewarded. This is violated by words (such as VIXEN) that contain more than one distinct Rare Alphabet. Fortunately, none of the words in the data is of this kind. Thus the results of the experiments still hold good. The above situation can be handled by modifying the rewarding scheme to handle the reward of more than one Type in Class 1, and adjusting the punishing scheme accordngly.

Although the author intended this research to serve more as an academic exercise than anything, he has been searching for a justification of an application of the adaptable heuristic. Until recently, a strong analogy between the 5-L-W-G and the job scheduler of an Operating System was realized. Typically, a job scheduler has to take into consideration the priority of jobs, the estimated execution times, the estimated lines to be printed, the number of I/O devices required, the types of jobs (whether I/O bound or CPU bound) and sometimes the deadlines of the jobs to be run.

Owing to the complicated requirements of the jobs, all job schedulers that exist today have to assume a particular job mix in order to optimize the average turnaround time, system

throughput or resource utilization. When the job mix of an installation changes, system programmers often have to modify the parameters of the job scheduler so as to obtain better performance. Conceptually, the author sees an analogy between the characteristics of jobs and those of five-lettered words. It is therefore suggested that a technique similar to the adaptable heuristic might be developed and used by the job scheduler to make it self-adjustable.

BIBLIOGRAPHY

1. Hunt, Earl B. Artificial Intelligence. Academic Press, 1975.

2. Jackson, Philip C. Introduction to Artificial Intelligence. Mason & Lipscomb, 1974.

3. Luce, R. Duncan and Raiffa, Howard. Games and decisions. New York: John Wiley & Sons, 1957.

4. Samuel, Arthur L. "Some Studies in Machine Learning Using the Game of Checkers." In Computers and Thought, pp. 71-105. Edited by Edward A. Feigenbaum and Julian Feldman. McGraw-Hill, 1963.

5. Singleton, Robert R. and Tyndall, William F. Games and Programs. San Francisco: W. H. Freeman and Company, 1974.

6. Slagle, James R. Artificial Intelligence: The Heuristic Programming Approach. McGraw-Hill, 1971.

7. Struble, George W. Assembler Language Programming: The IBM System/360 and 370. 2nd ed. Addison-Wesley, 1975.

8. "Toward an Intelligence Beyond Man's." Time 111(February 20, 1978), 59.

9. Watanabe, Satoshi. Knowing and Guessing: A Quantitative Study of Inference and Information. New York: Wiley & Sons, 1969.

10. Winston, Patrick H. Artificial Intelligence. Addison-Wesley, 1977.

11. Wonnacott, Thomas H. and Wonnacott, Ronald J. Introductory Statistics. 2nd ed. John Wiley & Sons, 1972.

APPENDIX A

List of all five-lettered words used in the programs.

ABACK	ABAFT	ABASE	ABATE	ABBEY	ABBOT	ABEAM	ABHOR
ABIDE	ABODE	ABOUT	ABOVE	ABUSE	ABYSS	ACORN	ACRID
ACTOR	ACUTE	ADAPT	ADDER	ADDLE	ADEPT	ADIEU	ADIOS
ADMIT	ADOBE	ADOPT	ADORE	ADORN	ADOWN	ADULT	AFFIX
AFIRE	AFOOT	AFORE	AFOUL	AFTER	AGAIN	AGAPE	AGATE
AGAVE	AGENT	AGILE	AGLOW	AGONY	AGREE	AHEAD	AISLE
ALARM	ALBUM	ALDER	ALERT	ALGAE	ALIAS	ALIBI	ALIEN
ALIGN	ALIKE	ALIVE	ALLAY	ALLEY	ALLOT	ALLOW	ALLOY
ALOES	ALOFT	ALONE	ALONG	ALOOF	ALOUD	ALPHA	ALTAR
ALTER	AMASS	AMAZE	AMBER	AMBLE	AMEND	AMISS	AMITY
AMONG	AMOUR	AMPLE	AMPLY	AMUCK	AMUSE	ANGEL	ANGER
ANGLE	ANGRY	ANISE	ANKLE	ANNEX	ANNOY	ANNUL	ANTIC
ANVIL	AORTA	APACE	APART	APHID	APHIS	APPLE	APPLY
APRON	ARDOR	ARENA	ARGON	ARGUE	ARISE	ARMOR	AROMA
ARRAS	ARRAY	ARROW	ARSON	ASHEN	ASHES	ASIDE	ASKEW
ASPEN	ASPIC	ASSAY	ASSET	ASTIR	ATLAS	ATOLL	ATOMY
ATONE	ATTIC	AUDIT	AUGER	AUGHT	AUGUR	AVAIL	AVAST
AVERT	AVOID	AWAIT	AWAKE	AWARD	AWARE	AWFUL	AWOKE
AXIAL	AXIOM	AZURE	BACON	BADGE	BADLY	BAGAY	BAIRN
BAIZE	BAKER	BALKY	BANAL	BANDY	BANJO	BARGE	BARON
BASAL	BASES	BASIC	BASIN	BASTE	BATCH	BATHE	BATON
BAYOU	BEACH	BEARD	BEAST	BEECH	BEEFY	BEFIT	BEFOG
BEGAN	BEGET	BEGIN	BEGOT	BEGUN	BEIGE	BELAY	BELCH
BELIE	BELLE	BELLY	BELOW	BENCH	BERRY	BERTH	BERYL
BESET	BESOM	BEVEL	BILLY	BIPED	BIRCH	BIRTH	BISON
BITCH	BLADE	BLAME	BLAND	BLANK	BLARE	BLAST	BLAZE
BLEAK	BLEAR	BLEAT	BLEED	BLEND	BLENT	BLESS	BLEST
BLIND	BLINK	BLISS	BLITZ	BLOAT	BLOCK	BLOOD	BLOOM
BLOWN	BLUES	BLUET	BLUFF	BLUNT	BLURT	BLUSH	BOARD
BOAST	BOBBY	BOGEY	BOGGY	BOGUS	BONNY	BONUS	BOOST
BOOTH	BOOTY	BORAX	BORIC	BORNE	BORON	BOSOM	BOSSY
BOTCH	BOUGH	BOUND	BOWER	BOXER	BRACE	BRACT	BRAID
BRAIN	BRAKE	BRAND	BRANT	BRASS	BRAVE	BRAVO	BRAWL
BRAWN	BREAD	BREAK	BREAM	BREED	BRIAR	BRIBE	BRICK
BRIDE	BRIEF	BRIER	BRINE	BRING	BRINK	BRINY	BRISK
BROAD	BROIL	BROKE	BROOD	BROOK	BROOM	BROTH	BROWN
BRUIN	BRUIT	BRUNT	BRUSH	BRUTE	BUDGE	BUGGY	BUGLE
BUILD	BUILT	BULGE	BULKY	BULLY	BUNCH	BUNNY	BURGH
BURLY	BURNT	BURST	BUSHY	BUTTE	BUXOM	BUYER	BYLAW
CABAL	CABIN	CABLE	CACAO	CACHE	CADET	CAIRN	CALYX
CAMEL	CAMEO	CANAL	CANDY	CANNA	CANNY	CANOE	CANON
CANST	CANTO	CAPER	CAPON	CARAT	CARDS	CARET	CARGO
CAROL	CARRY	CARVE	CASTE	CATCH	CATER	CAULK	CAUSE
CAVIL	CEASE	CEDAR	CHAFE	CHAIN	CHAIR	CHALK	CHAMP
CHANT	CHAPS	CHARM	CHART	CHARY	CHASE	CHASM	CHEAP
CHEAT	CHECK	CHEEK	CHEEP	CHEER	CHESS	CHEST	CHICK
CHIDE	CHIEF	CHILD	CHILL	CHIME	CHINA	CHINK	CHIRP
CHIVE	CHOIR	CHOKE	CHORD	CHORE	CHOSE	CHUCK	CHUNK
CHURL	CHURN	CHUTE	CIDER	CIGAR	CILIA	CINCH	CIVET
CIVIC	CIVIL	CLAIM	CLAMP	CLANK	CLASH	CLASP	CLASS

CLEAN	CLEAR	CLEAT	CLEFT	CLERK	CLOAK	CLOCK	CLOSE
CLOTH	CLOUD	CLOUT	CLOVE	CLOWN	CLUCK	CLUMP	CLUNG
COACH	COAST	COBRA	COCKY	COCOA	COLIC	COLON	COLOR
COMBE	COMER	COMET	COMIC	COMMA	CONCH	CONEY	COPRA
COPSE	CORAL	CORPS	COUCH	COUGH	COULD	COUNT	COUPE
COURT	COVER	COVET	COZEN	CRACK	CRAFT	CRAMP	CRANE
CRANK	CRAPE	CRASH	CRASS	CRATE	CRAVE	CRAWL	CRAZE
CRAZY	CREAK	CREAM	CREED	CREEK	CREEL	CREEP	CREPT
CRESS	CREST	CRIED	CRIER	CRIES	CRIME	CRIMP	CRISP
CROAK	CROCK	CROFT	CRONE	CRONY	CROOK	CROON	CROSS
CROUP	CROWD	CROWN	CRUDE	CRUEL	CRUET	CRUMB	CRUSE
CRUSH	CRUST	CRYPT	CUBIC	CUBIT	CURIO	CURLY	CURRY
CURSE	CURVE	CYCLE	CYNIC	DADDY	DAILY	DAIRY	DAISY
DALLY	DANCE	DANDY	DATUM	DAUNT	DAVIT	DEALT	DEATH
DEBAR	DEBIT	DECAY	DECOY	DECRY	DEFER	DEIFY	DEIGN
DEIST	DEITY	DELAY	DELFT	DELTA	DELVE	DEMON	DEMUR
DENIM	DENSE	DEPOT	DEPTH	DERBY	DERMA	DETER	DEUCE
DEVIL	DIARY	DINGY	DIRGE	DIRTY	DITCH	DITTO	DITTY
DIVAN	DIVER	DIZZY	DODGE	DOGIE	DOILY	DOING	DOUBT
DOUGH	DOUSE	DOWDY	DOWEL	DOWER	DOWNY	DOWRY	DRAFT
DRAIN	DRAKE	DRAMA	DRANK	DRAPE	DRAVE	DRAWL	DRAWN
DREAD	DREAM	DREGS	DRESS	DREST	DRIER	DRIES	DRIFT
DRILL	DRILY	DRINK	DRIVE	DROLL	DRONE	DROOL	DROOP
DROSS	DROVE	DROWN	DRUNK	DRYER	DRYLY	DUCAL	DUCAT
DUCHY	DULLY	DUMMY	DUMPS	DUMPY	DUNCE	DURST	DUSKY
DUSTY	DWARF	DWELL	DWELT	DYING	EAGER	EAGLE	EARLY
EARTH	EASEL	EATEN	EATER	EAVES	ECLAT	EDUCE	EGRET
EIDER	EIGHT	EJECT	ELATE	ELBOW	ELDER	ELECT	ELEGY
ELFIN	ELITE	ELOPE	ELUDE	ELVES	EMBED	EMBER	EMEND
EMERY	EMPTY	ENACT	ENDOW	ENDUE	ENEMY	ENJOY	ENNUI
ENTER	ENTRY	ENVOY	EPHOD	EPOCH	EQUAL	EQUIP	ERASE
ERECT	ERODE	ERUPT	ETHER	ETUDE	EVADE	EVENT	EVERY
EVICT	EVOKE	EXACT	EXALT	EXCEL	EXILE	EXIST	EXPEL
EXTRA	EXUDE	EXULT	FABLE	FACET	FAINT	FAIRY	FAITH
FAKIR	FALSE	FAMED	FANCY	FARCE	FATAL	FATED	FAULT
FAUNA	FAVOR	FEAST	FEIGN	FEINT	FELLY	FELON	FEMUR
FENCE	FERRY	FETCH	FETID	FETUS	FEVER	FIBRE	FIELD
FIEND	FIERY	FIFTH	FIFTY	FIGHT	FILCH	FILET	FILLY
FILMY	FILTH	FINAL	FINCH	FINIS	FINNY	FIORD	FIRST
FIRTH	FISHY	FITCH	FITLY	FIXED	FJORD	FLAIL	FLAIR
FLAKE	FLAME	FLANK	FLARE	FLASH	FLASK	FLECK	FLEET
FLESH	FLICK	FLIES	FLINT	FLIRT	FLOAT	FLOCK	FLOOD
FLOOR	FLORA	FLOSS	FLOUR	FLOUT	FLOWN	FLUFF	FLUID
FLUKE	FLUME	FLUNG	FLUNK	FLUSH	FLUTE	FOAMY	FOCAL
FOCUS	FOGGY	FOIST	FOLIO	FOLLY	FORCE	FORGE	FORGO
FORTE	FORTH	FORTY	FORUM	FOUND	FOUNT	FOYER	FRAIL
FRAME	FRANC	FRANK	FRAUD	FREAK	FRESH	FRIED	FRILL
FRISK	FRITH	FROCK	FROND	FRONT	FROST	FROTH	FROWN
FROZE	FRUIT	FUDGE	FUGUE	FULLY	FUNGI	FUNNY	FURRY
FURZE	FUSSY	FUSTY	FUZZY	GABLE	GAILY	GAMUT	GAUNT
GAUZE	GAUZY	GAVEL	GAWKY	GAYLY	GAZER	GECKO	GEESE
GENIE	GENII	GENOA	GENUS	GHOST	GHOUL	GIANT	GIDDY
GIPSY	GIRTH	GIVEN	GIVER	GLADE	GLAND	GLARE	GLASS
GLAZE	GLEAM	GLEAN	GLIDE	GLINT	GLOAT	GLOBE	GLOOM

GLORY GLOSS GLOVE GNARL GNASH GNOME GODLY GOING
GOODY GOOSE GORGE GORSE GOUGE GOURD GOUTY GRACE
GRADE GRAFT GRAIL GRAIN GRAND GRANT GRAPE GRAPH
GRASP GRASS GRATE GRATE GRAVE GRAVY GRAZE GREAT
GREBE GREED GREEN GREET GRIEF GRILL GRIME GRIND
GRIPE GRIST GRITS GROAN GROAT GROIN GROOM GROPE
GROSS GROUP GROVE GROWL GROWN GRUEL GRUFF GRUNT
GUANO GUARD GUESS GUEST GUIDE GUILD GUILE GUILT
GUISE GULCH GULES GULLY GUMBO GUMMY GUNNY GUSTO
GYPSY HABIT HAIRY HALVE HAPLY HAPPY HARDY HAREM
HARPY HARRY HARSH HASTE HASTY HATCH HAUNT HAVEN
HAVOC HAZEL HEADY HEARD HEART HEATH HEAVE HEAVY
HELLO HELOT HELVE HENCE HENNA HERON HEWER HINGE
HITCH HIVES HOARD HOARY HOBBY HOIST HOLLY HONEY
HONOR HORDE HORNY HORSE HORSY HOTEL HOUND HOVEL
HOVER HUFFY HUMAN HUMID HUMOR HUMPH HUMUS HUNCH
HURRY HUSKY HUSSY HUTCH HUZZA HYENA HYING ICING
IDEAL IDIOM IDIOT IDLER IDYLL IGLOO IMAGE IMBED
IMBUE IMPEL IMPLY INANE INAPT INCUR INDEX INDUE
INEPT INERT INFER INGOT INLAY INLET INNER INSET
IRATE IRONY ISLET ISSUE IVIED IVORY JABOT JAPAN
JAUNT JELLY JENNY JETTY JEWEL JIFFY JIMMY JOINT
JOIST JOKER JOLLY JOUST JUDGE JUICE JUICY JUMBO
JUMPY JUNCO JUROR KAYAK KETCH KIOSK KITTY KNACK
KNAVE KNEAD KNEEL KNELT KNIFE KNOCK KNOLL KNOWN
LABEL LABOR LADEN LADLE LAIRD LAITY LANCE LANKY
LAPEL LAPSE LARCH LARGE LARVA LASSO LATCH LATHE
LAUGH LAYER LEACH LEAFY LEAKY LEANT LEAPT LEARN
LEASE LEAST LEAVE LEDGE LEECH LEMON LEMUR LEPER
LEVEE LEVEL LEVER LIBEL LIEGE LIGHT LIKEN LILAC
LIMBO LIMIT LINEN LINGO LINKS LISLE LITER LITHE
LIVEN LIVER LIVES LIVID LOATH LOCAL LODGE LOFTY
LOGIC LOOSE LORRY LOSER LOTUS LOUSE LOUSY LOVER
LOWLY LOYAL LUCID LUCKY LUCRE LUNAR LUNCH LUNGE
LURCH LURID LUSTY LYING LYMPH LYNCH LYRIC MACAW
MADAM MADLY MAGIC MAIZE MAJOR MAKER MAMMA MANGE
MANGO MANGY MANIA MANLY MANOR MAPLE MARCH MARRY
MARSH MASON MASSY MATCH MAUVE MAVIS MAXIM MAYBE
MAYOR MEALY MEATY MEDAL MEDIA MELON MERCY MERGE
MERIT MERRY MESSY METAL METER MIDST MIGHT MILCH
MILKY MINCE MINER MINOR MINUS MIRTH MISER MISTY
MITER MIXER MIXUP MODEL MOGUL MOLAR MONEY MOODY
MOOSE MORAL MORON MOSSY MOTEL MOTIF MOTOR MOTTO
MOULD MOULT MOUND MOUNT MOURN MOUSE MOUTH MOVER
MOVIE MOWER MUCUS MUDDY MUFTI MUGGY MULCH MULCT
MUMMY MUMPS MUNCH MURAL MURKY MUSIC MUSKY MUSTY
MYRRH NADIR NAIAD NAIVE NAKED NASAL NASTY NATAL
NAVAL NEEDS NEEDY NERVE NEVER NEWSY NICHE NIECE
NIGHT NINTH NITER NOBLE NOBLY NOISE NOISY NOMAD
NONCE NOOSE NORTH NOTCH NOTED NOVEL NUDGE NURSE
NUTTY NYMPH OAKEN OAKUM OATEN OCCUR OCEAN OCHER
ODDLY ODIUM OFFAL OFFER OFTEN OLDEN OLIVE OMEGA
ONION ONSET OPERA OPINE OPIUM OPTIC ORBIT ORDER
ORGAN ORIEL OSIER OTHER OTTER OUGHT OUNCE OUTGO

OVARY OVERT OVULE OWING OWLET OWNER OXBOW OXIDE
OZONE PADRE PAEAN PAGAN PAINT PALSY PANDA PANEL
PANIC PANSY PANTS PAPER PARCH PARRY PARSE PARTS
PARTY PASSE PASTE PASTY PATCH PATIO PATTY PAUSE
PEACE PEACH PEARL PEASE PECAN PEDAL PENAL PENCE
PENNY PEONY PERCH PERIL PERKY PETTY PHASE PHIAL
PHONE PHOTO PIANO PIECE PIETY PIGMY PILOT PINCH
PINTO PIOUS PIPER PIPIT PIQUE PITCH PITHY PIVOT
PLACE PLAID PLAIN PLAIT PLANE PLANK PLANT PLATE
PLAZA PLEAD PLEAT PLUCK PLUMB PLUME PLUMP PLUMY
PLUSH POACH POINT POISE POKER POLAR POLKA POLYP
POPPY PORCH PORGY POSSE POUCH POUND POWER PRANK
PRATE PRAWN PREEN PRESS PRICE PRICK PRIDE PRIME
PRINT PRIOR PRISM PRIZE PROBE PROEM PRONE PRONG
PROOF PROSY PROUD PROVE PROWL PROXY PRUDE PRUNE
PSALM PSHAW PUDGY PUFFY PULPY PULSE PUNCH PUPIL
PUPPY PURSE PUSSY PUTTY PYLON PYREX QUAKE QUALM
QUART QUASH QUASI QUEEN QUEER QUELL QUERY QUEST
QUEUE QUICK QUIET QUILL QUILT QUIRE QUIRK QUIRT
QUITE QUITS QUOIT QUOTA RABBI RABID RACER RADAR
RADII RADIO RAINY RAISE RALLY RANCH RANGE RANGY
RATHE RATIO RAVEL RAVEN RAYON RAZOR REACH REACT
READY REBEL REBUS REBUT RECUR REEDY REFER REFIT
REGAL REIGN RELAX RELAY RELIC REMIT RENEW REPAY
REPEL REPLY RETCH RHINO RHYME RIDER RIDGE RIFLE
RIGHT RIGID RIGOR RINSE RIPEN RISEN RIVET ROACH
ROAST ROBIN ROBOT ROCKY RODEO ROGUE ROOMY ROSIN
ROUGE ROUGH ROUND ROUTE ROVER ROWDY ROWEL ROWER
ROYAL RUBLE RUDDY RULER RUMOR RUPEE RURAL RUSTY
SABLE SABOT SABRA SADLY SAHIB SAINT SAITH SALAD
SALES SALLY SALON SALVE SALVO SANDY SAPPY SATYR
SAUCE SAUTE SAVER SAVOR SAVOY SAYST SCALE SCALP
SCALY SCAMP SCANT SCARE SCARF SCARY SCENE SCENT
SCHWA SCION SCOFF SCOLD SCONE SCOOP SCOOT SCOPE
SCORE SCORN SCOUR SCOUT SCOWL SCRAG SCRAP SCREW
SCRIM SCRIP SCRUB SCUFF SCURF SEAMY SEDAN SEDGE
SEEDY SEINE SEIZE SENNA SENSE SEPAL SEPIA SEPOY
SERGE SERUM SERVE SHADY SHAFT SHAKE SHALE SHALL
SHAME SHANK SHAPE SHARD SHARE SHARK SHARP SHAVE
SHAWL SHEAF SHEAR SHEEN SHEEP SHEER SHEET SHELF
SHELL SHIFT SHINE SHINY SHIRE SHIRK SHIRR SHIRT
SHOAL SHOCK SHONE SHOON SHOOT SHORE SHORN SHORT
SHOTE SHOUT SHOVE SHOWN SHOWY SHRED SHREW SHRUB
SHRUG SHUCK SHUNT SHYLY SIBYL SIDLE SIEGE SIEVE
SIGHT SILKY SILLY SINCE SINEW SINUS SIREN SIRUP
SISAL SISSY SIXTH SIXTY SKALD SKATE SKEIN SKIES
SKIFF SKILL SKIMP SKIRT SKULK SKULL SKUNK SLACK
SLAIN SLAKE SLANG SLANT SLASH SLATE SLAVE SLEEP
SLEET SLEPT SLICE SLICK SLIDE SLIME SLING SLINK
SLOOP SLOPE SLOSH SLOTH SLUMP SLUNG SLUNK SLUSH
SLYLY SMACK SMALL SMART SMASH SMEAR SMELL SMELT
SMILE SMIRK SMITE SMITH SMOCK SMOKE SMOKY SMOTE
SNACK SNAIL SNAKE SNAKY SNARE SNARL SNEAK SNEER
SNIFF SNIPE SNOOD SNOOP SNORE SNORT SNOUT SNOWY
SNUFF SOBER SOGGY SOLAR SOLID SOLVE SONNY SOOTH

SOOTY SOPPY SORRY SOUGH SOUND SOUSE SOUTH SPACE
SPADE SPAKE SPANK SPARE SPARK SPASM SPAWN SPEAK
SPEAR SPECK SPEED SPELL SPELT SPEND SPEND SPENT
SPERM SPICE SPICY SPIED SPIKE SPIKY SPILL SPILT
SPINE SPINY SPIRE SPIRT SPITE SPITZ SPLAY SPLIT
SPOIL SPOKE SPOOK SPOOL SPOON SPOOR SPORE SPORT
SPOUT SPRAT SPRAY SPREE SPRIG SPRIT SPURN SPURT
SQUAD SQUAT SQUAW SQUIB SQUID STACK STAFF STAGE
STAID STAIN STAIR STAKE STALE STALK STALL STAMP
STAND STANK STARE STARK START STATE STAVE STEAD
STEAK STEAL STEAM STEED STEEL STEEP STEER STERN
STICK STIFF STILE STILL STILT STING STINK STINT
STOCK STOLE STONE STONY STOOD STOOL STOOP STORE
STORK STORM STORY STOUP STOUT STOVE STRAP STRAW
STRAY STREW STRIP STROP STRUM STRUT STUDY STUFF
STUMP STUNG STUNK STUNT STYLE SUEDE SUGAR SUITE
SULKY SULLY SUNNY SURGE SURLY SWAIN SWALE SWAMP
SWARD SWARE SWARM SWATH SWEAR SWEAT SWEEP SWEET
SWELL SWIFT SWILL SWINE SWING SWIPE SWIRL SWISH
SWOON SWOOP SWORD SWORE SWORN SWUNG SYLPH SYRUP
TABBY TABLE TABOO TABOR TACIT TAINT TAKEN TALLY
TALON TAPIR TARRY TASTE TASTY TAUNT TAUPE TAWNY
TEACH TEASE TEENS TEETH TEMPO TEMPT TENON TENOR
TENSE TENTH TEPEE TEPID TERSE THANE THANK THEFT
THEIR THEME THERE THEWS THIEF THIGH THINE THING
THINK THIRD THONG THORN THOSE THREE THREW THROB
THROE THROW THRUM THUMB THUMP THYME TIDAL TIGER
TIGHT TILDE TILTH TIMID TINGE TINNY TIPSY TIRED
TITHE TITLE TOADY TOAST TODAY TODDY TOKEN TONGS
TONIC TOOTH TOPER TOPIC TOQUE TORCH TORSO TOTAL
TOTEM TOUCH TOUGH TOWEL TOWER TOXIC TOXIN TRACE
TRACK TRACT TRADE TRAIL TRAIN TRAIT TRAMP TRASH
TRAWL TREAD TREAT TREND TRESS TRIAD TRIAL TRIBE
TRICE TRICK TRIED TRILL TRIPE TRITE TROLL TROOP
TROPE TROTH TROUT TRUCE TRUCK TRULY TRUMP TRUNK
TRUSS TRUST TRUTH TRYST TUBER TULIP TUMOR TUNNY
TUTOR TWAIN TWANG TWEAK TWEED TWEET TWILL TWINE
TWIRL TWIST TYING ULCER ULTRA UMBEL UMIAK UNCLE
UNDER UNION UNMAN UNSAY UNTIE UNTIL UPPER UPSET
URINE USHER USUAL USURP UTTER VALET VALID VALUE
VALVE VAPID VAPOR VAULT VAUNT VENAL VERGE VERSE
VETCH VIDEO VIGIL VIGOR VILLA VIOLA VISIT VISOR
VISTA VITAL VIVID VIXEN VOCAL VODKA VOGUE VOICE
VOMIT VOTER VOUCH VOWEL VYING WAFER WAGER WAGON
WAIST WALTZ WASTE WATCH WATER WAVER WAXEN WEARY
WEDGE WEEDS WEEDY WEIGH WEIRD WENCH WHACK WHALE
WHARF WHEEL WHELK WHELM WHELP WHERE WHICH WHIFF
WHILE WHINE WHIRL WHISK WHIST WHITE WHOLE WHOOP
WHORE WHORL WHOSE WIDEN WIDOW WIDTH WIELD WINCE
WINCH WINDY WIPER WITCH WITHE WITTY WIVES WOFUL
WOMAN WOODY WORDY WORKS WORLD WORMS WORRY WORSE
WORST WORTH WOULD WOUND WOVEN WREAK WRECK WREST
WRING WRIST WRITE WRONG WROTE WROTH WRUNG XEBEC
XYLEM YACHT YEARN YEAST YIELD YODLE YOKEL YOUNG
YOURS YOUTH ZEBRA

APPENDIX B

Source listing of PGM2

```
ADAPT     CSECT
          BALR  12,0
          USING *,12,11
          LA    11,*+4095
          LA    11,1(11)
          STM   14,12,12(13)
          ST    13,SAVEAREA+4
LASTCELL  EQU   9
THISCELL  EQU   10
          LA    13,SAVEAREA
          OPEN  (CARDIN,INPUT,PRINTOUT,OUTPUT)
*
AGAIN     MVC   RANGE(4),=F'2139'
          L     3,=F'2139'
          LR    5,3
          LA    2,LIST+12
          LA    4,LIST
LOOP11    ST    2,0(4)
          LA    2,12(2)
          LA    4,12(4)
          BCT   3,LOOP11
          S     4,=F'12'
          LA    2,LIST
          ST    2,0(4)
*
* READ IN THE SECRET WORD
          GET   CARDIN,INAREA
          MVC   SECWORD(5),INAREA
          PUT   PRINTOUT,PRINTSEC
*         SDUMP CLASSES,80
*
* COMPUTE FITNESS OF EACH TYPE
          LE    4,CLASSES
          LE    2,TYPES1
          MER   2,4
          STE   2,FITNESS1
          LE    2,TYPES1+4
          MER   2,4
          STE   2,FITNESS1+4
          LE    2,TYPES1+8
          MER   2,4
          STE   2,FITNESS1+8
          LE    2,TYPES1+12
          MER   2,4
          STE   2,FITNESS1+12
          LE    2,TYPES1+16
          MER   2,4
          STE   2,FITNESS1+16
          LE    2,TYPES1+20
          MER   2,4
```

```
          STE    2,FITNESS1+20
          LE     4,CLASSES+4
          LE     2,TYPES2
          MER    2,4
          STE    2,FITNESS2
          LE     2,TYPES2+4
          MER    2,4
          STE    2,FITNESS2+4
          LE     2,TYPES2+8
          MER    2,4
          STE    2,FITNESS2+8
          LE     2,TYPES2+12
          MER    2,4
          STE    2,FITNESS2+12
          LE     4,CLASSES+8
          LE     2,TYPES3
          MER    2,4
          STE    2,FITNESS3
          LE     2,TYPES3+4
          MER    2,4
          STE    2,FITNESS3+4
          LE     2,TYPES3+8
          MER    2,4
          STE    2,FITNESS3+8
          LE     2,TYPES3+12
          MER    2,4
          STE    2,FITNESS3+12
          LE     4,CLASSES+12
          LE     2,TYPES4
          MER    2,4
          STE    2,FITNESS4
          LE     2,TYPES4+4
          MER    2,4
          STE    2,FITNESS4+4
*         SDUMP CLASSES,80
*
* START OFF RANDOMLY FOR A GUESS WORD
*
* FOR THE FIRST GUESS:
          LA     1,PLIST1
          LA     15,RDIGIT
          BALR   14,15
          LR     4,0
          SR     2,2
          L      3,=F'12'
          MR     2,4
          LA     LASTCELL,LIST(3) ADDR. OF LAST CELL
          LA     THISCELL,12(LASTCELL)
*         RDUMP
          B      LABEL07
*
* FOR THE 2ND GUESS AND SO FORTH:
LABEL05   LA     15,RDIGIT
          LA     1,PLIST1
```

```
          BALR  14,15
          LR    3,0
          LTR   3,3
          BZ    LABEL07
LOOP14    LR    LASTCELL,THISCELL
          L     THISCELL,0(THISCELL)
          BCT   3,LOOP14
          B     LABEL07
*
* SEE IF RANGE IS NOT GREATER THAN 60
LABEL07   C     5,=F'60'
          BH    LABEL09
          C     5,=F'1'
          BE    PICKGUES
          LA    15,SIXTY
          BALR  14,15
          B     PICKGUES
*
* CHOOSE TYPE ACCORDING TO FITNESS PROBABAILTY DISTRIBUTION
LABEL09   LA    1,PLIST4
          LA    15,RDIGIT
          BALR  14,15
*         RDUMP
          ICM   0,8,=X'40'
          ST    0,FULLWORD
          LE    6,FULLWORD
          AE    6,=E'0.'
          SER   4,4
          LA    3,FITNESS1
          L     2,=F'16'
          L     4,TWOP15
LOOP12    AE    4,0(3)
          CER   4,6
          BH    LABEL1
          LA    3,4(3)
          SRL   4,1
          BCT   2,LOOP12
          PUT   PRINTOUT,ERRMSG1
          LM    14,12,12(13)
          BR    14
*
* CHOOSE GUESS WORD ACCORDING TO TYPE CHOSEN
LABEL1    C     4,=X'00000100'
          BNL   LABEL2
          MVC   TESTYPE+3(1),=X'0A'
          STCM  4,1,TESTYPE+1
          B     LABEL3
LABEL2    MVC   TESTYPE+3(1),=X'09'
          STCM  4,2,TESTYPE+1
LABEL3    L     2,RANGE
*         SDUMP TESTYPE,4
TESTYPE   TM    0(THISCELL),X'00'
          BO    PICKGUES       IF KEY MATCHES, CHOOSE WORD IN THISCELL
          LR    LASTCELL,THISCELL
```

```
         L      THISCELL,0(THISCELL)
         BCT    2,TESTYPE
*
PICKGUES MVC    GUESWORD(5),4(THISCELL)
         PUT    PRINTOUT,PRNTGUES
         LA     1,PLIST2        CONTAINS ADDRESS OF SECRET WORD AND
         LA     15,MATCHIT      ADDRESS OF GUESS WORD
         BALR   14,15           CALL MATCHIT
         C      0,=F'5'         DO ALL ALPHABETS MATCH?
         BE     BINGO           IF SO, CONGRATULATIONS
         ST     0,NMA           STORE THE NUMBER OF MATCHED ALPHABETS
         CVD    0,CVDNMA                  PRINT
         UNPK   UNPKNMA(1),CVDNMA(8) OUT
         OI     UNPKNMA,X'F0'    THE
         PUT    PRINTOUT,PRNTNMA  NMA
         L      7,RANGE         R7 IS FOR COUNTING NO.OF QUALIFIED WORDS
         L      5,RANGE         R5 IS FOR REDUCING THE NUMBER
         B      DELETE          1ST PASS THRU
LOOP2    LA     6,4(THISCELL) PREPARE
         ST     6,PLIST3+4      PARAMETER LIST
         LA     1,PLIST3
         LA     15,MATCHIT      CALL MATCHIT
         BALR   14,15
TESTED   C      0,NMA           ARE THE TWO NMA'S EQUAL?
         BE     KEEP            IF SO, STILL QUALIFIED
DELETE   MVC    0(4,LASTCELL),0(THISCELL) MOVE POINTER OF THIS CELL
         L      THISCELL,0(THISCELL)       TO THE LAST CELL
         S      5,=F'1'                    SUBTRACT 1 FROM RANGE
         B      CONTINUE
KEEP     LR     LASTCELL,THISCELL
         L      THISCELL,0(THISCELL)
CONTINUE BCT    7,LOOP2
         ST     5,RANGE         STORE NEW VALUE OF RANGE
         CVD    5,CVDRANGE                PRINT
         UNPK   UNPKRANG(5),CVDRANGE(8) OUT
         OI     UNPKRANG+4,X'F0'          THE
         PUT    PRINTOUT,PRNTRANG         RANGE
         LTR    5,5             ARE ALL WORDS DISQUALIFIED?
         BP     LABEL05         IF NOT,DO IT AGAIN
RUNOUT   DS     0H
         PUT    PRINTOUT,MESSAGE ELSE, SAY THE LIST IS EXHAUSTED
         B      AGAIN
BINGO    DS     0H
         PUT    PRINTOUT,YOUGOTIT
*        L      2,RN                    PRINT
*        CVD    2,PACKRN                OUT
*        UNPK   UNPKRN(10),PACKRN(8) THE RANDOM NUMBER
*        OI     UNPKRN+9,X'F0'          FOR
*        PUT    PRINTOUT,PRINTRN        THE NEXT RUN
*
* REWARD AND PUNISH TYPES AND CLASSES
*        PUT    PRINTOUT,0(THISCELL)
         SR     2,2
         SER    0,0
```

```
         ICM   2,12,9(THISCELL) LOAD THE PROP. BITS OF THE LAST SECWORD
*        RDUMP
         LA    3,TYPES1
         LE    2,ALPHA
         LE    4,=E'1.'
         SER   4,2
         LA    4,1
LOOP15   LTR   2,2
         BM    LABEL5
         LE    6,0(3)
         MER   6,4
         STE   6,0(3)
         B     LABEL6
LABEL5   LE    6,0(3)          BIT IS ON
         MER   6,4
         AER   6,2
         STE   6,0(3)
         DE    6,48(3)         NORMALIZE BY DIVIDING BY DIST. PROB.
         CER   0,6
         BNL   LABEL6
         LER   0,6             REPLACE WITH MAX.
         LR    5,4
LABEL6   LA    4,1(4)
         SLL   2,1
         LA    3,4(3)
         C     4,=F'16'
         BNH   LOOP15
         LE    2,BETA
         LE    4,=E'1.'
         SER   4,2
         LA    3,CLASSES
         LA    4,4
LOOP16   LE    6,0(3)
         MER   6,4
         STE   6,0(3)
         LA    3,4(3)
*        SDUMP CLASSES,80
         BCT   4,LOOP16
         C     5,=F'6'
         BH    LABEL7
         LA    5,1
         B     LABEL10
LABEL7   C     5,=F'10'
         BH    LABEL8
         LA    5,2
         B     LABEL10
LABEL8   C     5,=F'14'
         BH    LABEL9
         LA    5,3
         B     LABEL10
LABEL9   LA    5,4
LABEL10  S     5,=F'1'
         SLA   5,2
         AE    2,CLASSES(5)
```

```
          STE   2,CLASSES(5)
*         SDUMP CLASSES,80
          B     AGAIN
CLOSEFL   CLOSE (CARDIN,DISP)
          CLOSE (PRINTOUT,DISP)
          L     13,SAVEAREA+4
          STM   14,12,12(13)
          BR    14
          DS    0H
*****************************************************************************
RANDOM    STM   14,12,12(13)  THIS
          BALR  12,0          SUBROUTINE
          USING *,12,11       IS
          LA    11,*+4095
          LA    11,1(11)
          L     7,RN          ADOPTED FROM
          M     6,=F'65541'   THE
          ST    7,RN          TEXTBOOK
          LR    0,7           OF
          LM    1,12,24(13)   CS 204
          BR    14            WRITTEN
RDIGIT    STM   14,12,12(13)  BY
          BALR  12,0          STRUBLE.
          USING *,12,11       IT GENERATES A RANDOM NUMBER
          LA    11,*+4095
          LA    11,1(11)
          ICM   2,8,=B'00000110' DISABLE INTERRUPT
          SPM   2
          ST    13,SAV+4      BETWEEN 0 AND N-1
          LA    13,SAV        WHERE N IS THE PARAMETER PASSED INTO IT
          L     4,0(1)        ADDRESS OF PARAMETER
          L     5,0(4)        DIGIT LIMIT
          L     15,RANDAD
          BALR  14,15
          LPR   7,0           ABSOLUTE VALUE OF RN
          M     4,=F'2'
          MR    4,7
          LR    0,4
          ICM   2,8,=B'00001110' ENABLE INTERRUPT
          SPM   2
          L     13,SAV+4
          LM    14,15,12(13)
          LM    1,12,24(13)
          BR    14
          DS    0H
*****************************************************************************
MATCHIT   STM   14,12,12(13)  THIS
          BALR  12,0          SUBROUTINE
          USING *,12,11       FINDS
          LA    11,*+4095
          LA    11,1(11)
          SR    6,6           THE
          L     2,0(1)        NUMBER
          L     3,4(1)        OF
```

```
          MVC   SAVESEC(5),0(2) MATCHED
          L     4,=F'5'         ALPHABETS
NEXTONE   L     5,=F'5'         FOR
          LA    2,SAVESEC       A GUESS-WORD.
TRYAGAIN  CLC   0(1,2),0(3)     1ST
          BNE   NOTMATCH        PARAMETER
          MVI   0(2),C'0'       IS
          LA    6,1(6)          THE
          B     JUMP                 FIRST PASS-THRU'
NOTMATCH  LA    2,1(2)          ADDRESS
          BCT   5,TRYAGAIN      OF
JUMP      LA    3,1(3)          THE
          BCT   4,NEXTONE       SECRET
          LR    0,6             WORD.
          LM    14,15,12(13)    RESTORE
          LM    1,12,24(13)     REGISTERS
          BR    14              END OF MATCHIT
SAVESEC   DS    CL5             FOR MANIPULATING THE SECRET WORD
          DS    0H              FOR ALIGNMENT
*********************************************************************************
*
* THIS SUBROUTINE MAKES SURE THAT THE REDUCTION FACTOR FOR THE NEXT
* GUESS-WORD IS AT LEAST 3 OR THE MAXIMUM OF ALL QUALIFIED WORDS
SIXTY     STM   14,12,12(13)
          BALR  12,0
          USING *,12,11
          LA    11,**+4095
          LA    11,1(11)
          ST    5,MINOFMAX
          L     2,RANGE
          L     3,RANGE
          S     3,=F'1'
          ST    3,RANGEM1
          ST    13,SAVAREA1+4
          LA    13,SAVAREA1
          SR    4,4
          D     4,=F'3'
          ST    5,MAXNEXT
          PUT   PRINTOUT,=CL133' ENTER SUBROUTINE SIXTY'
LOOP31    SR    4,4
          SR    5,5
          SR    6,6
          SR    7,7
          SR    8,8
          LA    0,4(THISCELL)
          ST    0,PLIST5
          L     14,0(THISCELL)
          LA    14,4(14)
          ST    14,PLIST5+4
          L     3,RANGEM1
          L     15,AMATCHIT
LOOP30    LA    1,PLIST5
          BALR  14,15
          C     0,=F'5'
```

```
          BNL   JMP0
          A     0,=F'4'
          SLL   0,4
          STC   0,*+9
          STC   0,*+6
          LA    0,1(0,0)
JMP0      L     14,PLIST5+4
          S     14,=F'4'
          L     14,0(14)
          LA    14,4(14)
          ST    14,PLIST5+4
          BCT   3,LOOP30
          C     4,MAXNEXT
          BH    DISSAT
          C     5,MAXNEXT
          BH    DISSAT
          C     6,MAXNEXT
          BH    DISSAT
          C     7,MAXNEXT
          BH    DISSAT
          C     8,MAXNEXT
          BH    DISSAT
SAT       PUT   PRINTOUT,=CL133' NEXT GUESS-WORD HAS REDUCTION FACTOR OFX
                 AT LEAST 3'
          B     EXIT30
DISSAT    LR    0,4
          CR    0,5
          BNL   JMP1
          LR    0,5
JMP1      CR    0,6
          BNL   JMP2
          LR    0,6
JMP2      CR    0,7
          BNL   JMP3
          LR    0,7
JMP3      CR    0,8
          BNL   JMP4
          LR    0,8
JMP4      DS    0H
          C     0,MINOFMAX
          BNL   JMP5
          ST    0,MINOFMAX
          ST    THISCELL,SAVTHIS
          ST    LASTCELL,SAVLAST
JMP5      DS    0H
          LR    LASTCELL,THISCELL
          L     THISCELL,0(THISCELL)
          BCT   2,LOOP31
          L     THISCELL,SAVTHIS
          L     LASTCELL,SAVLAST
EXIT30    L     13,SAVAREA1+4
          LM    14,8,12(13)
          LM    11,12,64(13)
          BR    14
```

```
SAVAREA1 DS    18F
PLIST5   DS    2F
MAXNEXT  DS    1F
RANGEM1  DS    1F
MINOFMAX DS    1F
SAVTHIS  DS    1F
SAVLAST  DS    1F
AMATCHIT DC    A(MATCHIT)
         DS    0H
*********************************************************************
         LTORG
CARDIN   DCB   DSORG=PS,RECFM=FBA,MACRF=GM,BLKSIZE=800,LRECL=80,       X
               DDNAME=SYSIN,EODAD=CLOSEFL
PRINTOUT DCB   DSORG=PS,RECFM=FBA,MACRF=PM,BLKSIZE=1330,LRECL=133,     X
               DDNAME=SYSPRINT
CVDNMA   DS    1D
CVDRANGE DS    1D
PACKRN   DS    1D
RN       DC    F'847981089'
SAV      DS    18F
RANDAD   DC    A(RANDOM)
SAVEAREA DS    18F
NMA      DS    1F
         DC    CL8'CLASSES'
CLASSES  DC    4E'0.25'
TYPES1   DC    E'1.168224E-2,1.448598E-2,6.355140E-2'
         DC    E'1.635514E-2,1.308411E-2,0.8808411'
TYPES2   DC    E'0.2827103,1.168224E-2'
         DC    E'4.672897E-2,7.009346E-1'
TYPES3   DC    E'6.074766E-3,3.252336E-1'
         DC    E'5.682243E-1,1.004873E-1'
TYPES4   DC    E'1.387850E-1,8.612150E-1'
DISTRIB1 DC    E'1.168224E-2,1.448598E-2,6.355140E-2'
         DC    E'1.635514E-2,1.308411E-2,0.8808411'
DISTRIB2 DC    E'0.2827103,1.168224E-2'
         DC    E'4.672897E-2,7.009346E-1'
DISTRIB3 DC    E'6.074766E-3,3.252336E-1'
         DC    E'5.682243E-1,1.004873E-1'
DISTRIB4 DC    E'1.387850E-1,8.612150E-1'
FITNESS1 DS    6E
FITNESS2 DS    4E
FITNESS3 DS    4E
FITNESS4 DS    2E
PLIST4   DC    A(TWOP24)
TWOP24   DC    X'01000000'    MUST BE IN FULL WORD BOUNDARY
FULLWORD DS    1F
TWOP15   DC    X'00008000'
ALPHA    DC    E'0.25'
BETA     DC    E'0.25'
PLIST1   DC    A(RANGE)
PLIST2   DC    A(SECWORD)
PLIST3   DC    A(GUESWORD)
         DS    1F
RANGE    DC    F'2139'        NUMBER OF QUAL. WORDS IN EACH GUESS
```

```
INAREA    DS     CL80
PRINTSEC  DC     C'0THE SECRET WORD IS:   '
SECWORD   DC     CL111' '
PRNTGUES  DC     C' GUESS-WORD:   '
GUESWORD  DC     CL119' '
PRNTNMA   DC     C' NMA = '
UNPKNMA   DC     CL126' '
PRNTRANG  DC     C' RANGE = '
UNPKRANG  DC     CL124' '
PRINTRN   DC     C' NEW SEAT = '
UNPKRN    DC     CL121' '
MESSAGE   DC     CL133' ALL WORDS EXAMINED.'
YOUGOTIT  DC     CL133' YOU GOT IT!'
ERRMSG1   DC     CL133'0COMPUTATION ERROR WITH TYPE PROBABILITIES'
          DS     0F
LIST      DC     C'     ABACK'
          DC     B'0000011000001010'
          DC     C'      ABAFT'
          DC     B'0000011000001010'
          DC     C'      ABASE'
          DC     B'0000011000000110'
          DC     C'      ABATE'
          DC     B'0000011000000110'
          DC     C'      ABBEY'
          DC     B'0000011000001010'

                      .
                      .     <---------- this part omitted
                      .

          DC     C'      YOURS'
          DC     B'0000010001001001'
          DC     C'      YOUTH'
          DC     B'0000010001001001'
          DC     C'      ZEBRA'
          DC     B'0000100001001001'
          END    ADAPT
```