A computer program for generating diagnostic keys

R. J. Pankhurst

University Engineering Laboratory, Trumpington Street, Cambridge

Diagnostic keys are widely used in biology for the identification of specimens, but the technique is applicable to recognition processes in general. A key is constructed from a matrix which describes the properties of a set of objects. It is usually possible to contrive a large number of different keys for one set of objects, although the keys will not be equally useful. The construction and editing of keys by hand requires much labour, and a FORTRAN program is described here to do this automatically. The algorithm explores all the possible keys in a tree-searching process, and selects an optimal key heuristically. A special purpose data structure is used to represent the key in computer memory.

(Received July 1969)

Diagnostic keys are used for identifying an individual object out of a set of objects which resemble one another. They are principally used by biologists, although the problems involved are of a very general nature which form part of the study of classification or taxonomy. Two complementary problems occur in taxonomy. These are:

(*a*) Given a set of objects, examine their characteristics in order to find a classification, i.e. group the objects into subsets, and assign names to the subsets.
(*b*) Given a classification and an object, identify that object, i.e. given a list of the characteristics of named subsets which are known to exist and an additional object, decide which subset the object belongs to (recognise it, or find its name).

Keys are used for the second type of problem, once a case of the first type has been solved. In other words, assuming a classification, a key is used to do an identification. Keys could be used in cases such as medical diagnosis, analysis of personality by psychologists, or fault tracing in motor car engines. Keys have the greatest usefulness when the objects to be distinguished resemble one another closely, and when the user is unfamiliar with the objects. Keys have been used historically since at least the eighteenth century (Voss, 1952).

An example of a key is given in **Fig. 1.** The objects (birds) concerned were chosen for their familiarity rather than as an illustration of where a key might be useful. Suppose one has seen a bird which is presumed to be one of these five. Start at label 1 of the key. If the bird could fly, go to label 2, otherwise to label 4. At label 2, if it was a water bird, we know that it was a duck, otherwise go to label 3 to distinguish between a sparrow and an eagle. At this point, the identification should be checked by careful comparison with a detailed description, or with a photograph, or if necessary, by asking an expert. There is still the possibility that the object examined was not included in the key anyway. This shows up as an unlikely result or as poor agreement with the description, in which case a different key might give the right answer.

1	Bird flies	2
	Bird cannot fly	4
2	Lives by water	*Duck*
	Lives elsewhere	3
3	Less than 30 cm long	*Sparrow*
	More than 30 cm long	*Eagle*
4	Lives by water	*Penguin*
	Lives elsewhere	*Chicken*

Fig. 1. Key for birds

The information from which the key of Fig. 1 was derived is shown in **Table 1.** Each object (species, taxon or concept) is described by a set of pairs of characters (attributes, features or properties) with values (or states). For example, the object 'duck' is represented by (flies, yes), (lives by water, yes), (length, about 30 cm), (bill, flat and spoon-shaped), (talons, none). For example, the character 'flies' has the value 'yes'. Characters may be binary- or multi-valued. Binary valued characters have only two values, such as (0, 1) or (yes, no) or (present, absent). Multi-valued characters take on more than two values, but can always be reduced to a set of binary valued characters if desired. For example, the values under the character 'bill' in Table 1 can be split into (curved, not curved), (flat, not flat), etc. The values of a character to be used in a key have to be mutually exclusive, a point which is sometimes overlooked.

The values of characters as considered here are, in general, alphanumeric strings of arbitrary length. A special case is when the value is unknown or missing. Values can be missing because they are simply inappropriate, e.g. in botany, no value can be given for

'shape of teeth on leaf' for a plant which has no teeth on its leaves. A value can be missing because that character on a given object is too variable to be useful, or because the author of the original classification did not make a complete record. The convention in biology is to call a missing value 'not coded', abbreviated as NC (Sokal and Sneath, 1963).

Table 1

Classification of birds

	FLIES	LIVES BY WATER	LENGTH	BILL	TALONS
duck	yes	yes	30 cm	flat and spoon-shaped	no
penguin	no	yes	30 cm	triangular	no
eagle	yes	no	100 cm	curved	yes
chicken	no	no	30 cm	straight, narrow	no
sparrow	yes	no	10 cm	triangular	no

Some characters have proved more useful for the identification of objects than others. As an example, with a plant it is easier to find the colour of a flower than to examine leaf cells. This is a subjective distinction; whether or no such characters differ in their objective importance for classifying objects (species) has been a controversial matter in biology for several centuries. A more useful character may be said to have a lower 'cost'. The cost is usually not measurable in economic terms, except for chemical tests used in medicine or microbiology, so it is usually expressed as a scheme of character weights. The weights do not have measurable numerical values in biology, but are just indicated by printing the more useful characters in italics. When characters are variable, the different values can be assigned different probabilities of occurrence. Again, in biological cases, these probabilities are not often measured, since they are not constant. For this reason, constant characters are strongly preferred.

The probability with which an object of each type may occur can also be considered. The correct identification of a rarity is not a common event. With biological species, numerical measures of the relative abundance of species are not usually available. For medical diagnosis, however, records of the incidence rates of different diseases have been kept.

A diagnostic key may also be thought of as a decision table or tree. The key of Fig. 1 is represented as a tree in **Fig. 2**. This example key is dichotomous, i.e. every node has at most two branches. Each node corresponds to a labelled question, and the twigs all end with the name of an object. Keys with more than two branches at a node (polychotomous keys) can exist. Keys are often referred to as artificial keys, with the implication that a natural key might exist. A natural key, if it can be found, is presumed to reflect an inherent structure in the objects concerned, with the most important distinguishing characters appearing first. The keys generated by the program considered here are artificial, since they are produced by considerations of convenience rather than naturalness.

Different forms of artificial key are possible. The two common forms are known as the bracketed or parallel type and the indented type, although these terms are not very helpful. The first type of key is as in Fig. 1. The alternative questions that are possible at a node at a given level (as in Fig. 2) are written together. In the second type of key, as in **Fig. 3,** the questions at a given node are set apart while the dependent alternatives are explored in between. In terms of the tree of Fig. 2, the first type corresponds to tracing out the tree from left to right first and then from the top to the bottom, whereas the second type corresponds to tracing out the tree from top to bottom first and then from left to right. The indentation shown in Fig. 3 can be used in either type of key and it reflects the level of a node within the tree. The lower the level of a node the further the question is displaced to the right as the key is written out.

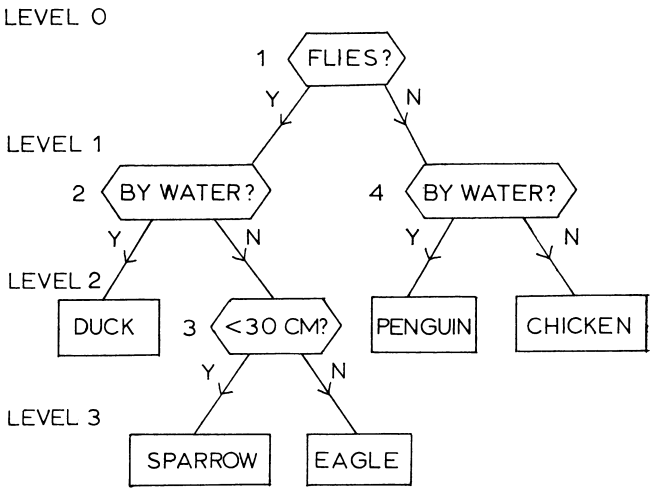

Fig. 2. Tree version of simple key

Dichotomous keys are often preferred to polychotomous keys because it is easier to choose between two alternatives than between many. Combinations of characters may be used in the questions posed, e.g. the questions labelled 3 in Fig. 3, but since correlations between characters may not always occur, single characters appear most frequently. It requires less mental effort to answer a question of the form 'Is A true?' than the more complicated form 'Is A true and B true and . . . ?' The shortest possible key gives the fastest identification, provided that the character values are equally easy to recognise, and it has been shown that the shortest key is obtained when the tree is dichotomous and the objects are divided into equal sized groups at each node (Osborne, 1963). Hence a key offers the advantage of being the fastest method of identification. It has certain disadvantages: (i) some of the characters used in the key may not be available on the object; (ii) the values of the characters used may not be easy to observe for subjective reasons. One author (Leenhouts, 1966) has proposed that one should not use a key at all, but work from a synopsis, i.e. a summary of the classification as in Table 1. The user is then free to use what characters he pleases, but he has to make considerably more effort since no help is given in the process of comparison. No allowance has been made in the above discussion for the fact that the values of characters might be variable. Evidently a set of objects must have at least some constant properties for the original classi-

```
1     STEM 0-10 CM.                                                    2
 2     STERILE ROSETTES ABSENT, CAPITULA MORE THAN 3 CM.              17.J.FONTQUERI
 2     STERILE ROSETTES PRESENT, CAPITULA UP TO 3 CM.                   3
  3     CAPITULA OBCONICAL, INVOLUCRAL BRACTS LAX,PATENT OR RECURVED.  15.J.HUMILIS
  3     CAPITULA SUBGLOBOSE, INVOLUCRAL BRACTS APPRESSED.              16.J.TAYGETEA
1     STEM MORE THAN 10 CM.                                            4
 4     PAPPUS SHORTER THAN ACHENE.                                    11.J.POLYCLONOS
 4     PAPPUS LONGER THAN ACHENE.                                       5
  5     INVOLUCRAL BRACTS LAX,PATENT OR RECURVED.                        6
   6     CAPITULA MORE THAN 3 CM.                                         7
    7     STEM LEAFY THROUGHOUT.                                 10B.J.MOLLIS.SSP.MOSCHATA
    7     STEM LEAFY AT BASE.                                              8
     8     INVOLUCRAL BRACTS LANCEOLATE.                                10.J.MOLLIS
     8     INVOLUCRAL BRACTS LINEAR.                                  14.J.GLYCACANTHA
   6     CAPITULA UP TO 3 CM.                                             9
    9     STEM WOODY AT BASE.                                          6.J.ALBICAULIS
    9     STEM HERBACEOUS.                                                10
    10     BASAL LEAVES SUBGLABROUS ABOVE, TOMENTOSE BENEATH.           9.J.EVERSMANII
    10     BASAL LEAVES PUBERULENT ABOVE, TOMENTOSE BENEATH.           12.J.LEDEBOURI
  5     INVOLUCRAL BRACTS APPRESSED.                                     11
  11     BASAL LEAVES SUBGLABROUS ABOVE, TOMENTOSE BENEATH.               12
   12     DISTAL CROWN OF ACHENE INCONSPICUOUS.                            13
    13     CAPITULA SUBGLOBOSE.                                         8.J.CYANOIDES
    13     CAPITULA HEMISPHERICAL.                                    13,J.CONSANGUINEA
   12     DISTAL CROWN OF ACHENE CONSPICUOUS                               14
    14     RHIZOME ABSENT.                                            2.J.STOECHADIFOLIA
    14     RHIZOME PRESENT.                                           3.J.TZAR-FERDINANDI
  11     BASAL LEAVES ARACHNOID TOMENTOSE.                                15
   15     STERILE ROSETTES PRESENT.                                        16
    16     BASAL LEAVES PINNATIFID, CAPITULA OBCONICAL.                 4.J.PINNATA
    16     BASAL LEAVES ENTIRE, CAPITULA HEMISPHERICAL.                 7.J.KIRGHISORUM
   15     STERILE ROSETTES ABSENT.                                         17
    17     STEM WOODY AT BASE, BASAL LEAVES ENTIRE.                     1.J.LINEARIFOLIA
    17     STEM HERBACEOUS, BASAL LEAVES PINNATIFID.                    5.J.TANAITICA
```

Fig. 3.

A computer generated key

fication to be possible at all. However, when variation is present an identification process should lead to the name of the object with the highest probability of being correct.

Keys and computing

A better method of using a computer for identification of biological specimens is to do this on-line. Goodall (1968) has described a system for on-line recognition of botanical specimens. This is based upon a question and answer procedure. The user first gives the name of the group of objects (genus) to which he thinks his object (plant) belongs. The system gives out a list of the characters which are available, and the user chooses one and selects a value for it. The system then proposes a new choice, and this is repeated until the object is identified. The procedure is like that involved in using a key, except that the user is completely free to choose which of the available questions he will answer at each stage. This is one special case of the many computer systems used for recognition problems. It has the advantages of giving the most freedom for subjective preference and being very easy to use. Unfortunately, it is the more expensive method, and cannot be used in the field or if there is no means of access to the computer. For these reasons, printed keys will very likely continue to be useful for some time.

The traditional key used by biologists is often only available for more or less familiar objects (species), and a worker who wants to specialise may often have to manage without. The effort required in inventing a key from scratch by hand is considerable, which is why keys are often not available. Further, practical experience with a hand-produced key will often show up the need for revisions, so that the labour has to be repeated. Lastly, yet more effort is needed in order to produce a best key (in some sense), out of the many keys that are usually possible, so keys are not usually optimised at all. These are the reasons for wishing to generate keys by computer. Morse, Beaman and Shetler (1968) state plainly the need for the development of key generation programs as a part of the Flora North America project.

Some previous work has been done in this direction by biologists. Moller (1962) gives a statistical theory for allowing for the observed statistical variation in some of the characters of objects, in order to choose a key which gives the maximum probability of correct identification. Binary state characters only are used, of which no characters may have missing values. Computing is used only for the statistical calculations, and not for producing the printed key.

Morse and his co-workers describe two similar computer programs for editing existing keys (Morse, Beaman and Shetler, 1968) and (Morse, 1968). The principle used is that the tree structure of the key and the questions or statements that appear in the printed form can be described separately. The user specifies a different tree and the system edits the statements and prints them in a different order to get a new key. Either of the two types of key can be selected.

Niemela, Hopkins and Quadling (1968) describe a system for choosing a key (for microbes). The matrix of species versus character values has to be complete, i.e. no missing values are allowed, and the characters must be binary valued (± 1). The final key is produced only in numerical form, so that the work of writing out the key remains, and only dichotomous keys can be produced, since characters are considered only one at a time at each node. Characters can be selected or ignored, but selective weighting of characters is not included. Tests are made to detect redundant characters, i.e. characters must have more than one value amongst the species considered. Two means of selecting a character for making a further division at a node of the key were tested. (i) Summing each set of character values for all species. Since ± 1 are used as values, the character which divides the set of species most nearly into two equal groups will give the summed value nearest zero. (ii) Minimising the theoretical number of different keys which could result from further subdivisions from a given node. This amounts to minimising the logarithm of the number of combinations, which is a function of factorials.

Morse (1969) has developed his earlier work into a key constructing program which also prints the key. Character value pairs are treated as inseparable couplets, which is an unusual approach. The couplets can only represent two different values for a character, and only one couplet can be considered for use in each question. Missing values are allowed for. Couplets are selected according to whichever divides the objects into two groups of most nearly the same size.

Artificial intelligence work has also been concerned with the construction of decision trees, equivalent to keys, although the application of this to biology appears to have been overlooked. In the following discussion, each of the projects mentioned has been concerned with only dichotomous trees using only binary valued characters. The use of more than one character per test (question or lead) has not been considered, and the effect of missing values has been neglected. In each case, the techniques for storage and manipulation of the decision tree in the computer are similar to those used by the key program.

Hunt, Marin and Stone (1966) describe experiments in both computing and psychology concerned with what they call concept learning. Data concerning objects is available, and is arranged in the familiar character-value matrix. One of the character values of the objects shows

whether or no the object is a case of a known phenomenon or concept. For example, hospital patients either suffered infection in surgical wounds or did not. The concept is 'risk of infection' which needs to be identified. The objects are the patients, and the characters and values are a set of clinical tests and their results. Each object is known to coincide with the concept (a positive instance) or not (a negative instance). The decision tree gives only two possible results, true or false, corresponding to risk of infection or otherwise in the example. The decision tree is therefore concerned to detect positive instances as soon as possible, whereas a biological key tries to distinguish any one object from many others with equal efficiency. Further, there is nothing corresponding to the concept to be recognised in the key problem.

Slagle (1964) assumes that a decision tree has been given, together with the cost and probability of successful outcome of each test, and shows how to rearrange the tree to minimise the average cost. The same set of tests is retained in each case. The tests are re-arranged in the order of ascending ratio of the cost to the probability. This is an interesting extension to the key problem, but unfortunately the costs and probabilities are not in general available in the applications for which the key program was needed.

Winston (1969) also typifies objects with a character-value matrix, and he takes the cost of a test into account with the probability of the object occurring, but not the probability of the testing. Possible ways of branching the tree are compared by minimising a function of the cost and probability, so as to choose the cheapest test and at the same time divide the group of objects into two groups each with nearly equal probability of occurrence. Methods of rearranging the tree by considering simultaneously more than one consecutive branch are discussed and are shown to improve the tree. It is suggested that more sophisticated rearrangement algorithms could be prohibitively expensive in computer time and give only slightly better trees.

Desirable features of a key

This is a subjective matter. Metcalf (1954) gives an account of his preferences with a very amusing example key for an imaginary genus Paradoxus. The following criteria are suggested:

(i) Common objects should be reached in the key by the shortest path. Also highly distinctive objects should key out quickly.

(ii) Dichotomous keys are preferred to polychotomous keys since it is easier to choose between two rather than many alternatives. Similarly each question in the key should not involve too many characters, since this makes the question harder to answer. However, in cases where an object can only be distinguished by having a certain combination of character values the use of many characters at once may be essential.

(iii) Care is needed in the use of characters which are known to be variable, because they cause uncertainty in identification. One way to avoid the problem is to add extra objects (with perhaps the same name) with different character values to cover the range of variation. If sufficient constant characters are available, the variable characters can be ignored.

(iv) The key should be ordered according to subjective considerations of convenience, and not according to the classification which lies behind it, since this ordering may be different. For example, plants are classified by their flower structure, whereas keys for identifying them can take advantage of simple obvious properties like flower colour. If the availability of the characters of the object varies, as for instance biological species present different features at different seasons, then a variety of differently ordered keys should be available.

(v) Where the use of the second type of key is concerned, it is best to put the shortest sub-tree from a node first, so that it is easier to cast forward for the alternative questions at this level.

(vi) It is a help to be able to trace the key backwards if an error is suspected. This can be done easily with the second type of key owing to the structure; with the first type of key backward pointers (question numbers) would have to be added.

(vii) A key may not need to use all the characters of an object to give an answer. When the object keys out, it is useful to have all the remaining distinctive characters printed at that key branch, since there is always a finite possibility that the specimen does not fit the key at all, and this acts as a check. A distinctive character in this context is any character whose value for the identified object is different from the values of the same character for all the other objects belonging to other branches from the same node of the key.

Features of the key program

The program produces keys which cover most of the possible requirements listed above. Its facilities are:

(i) Program accepts any of the possible strings of characters from the FORTRAN set as an object, character or value.

(ii) The characters can be binary or multi-state as required. Correspondingly, dichotomous or polychotomous keys are likely to be produced.

(iii) Characters can be given numerical weights to reflect the users's preferences.

(iv) Either type of key can be produced, both with or without indentation.

(v) Unusual species are keyed out first wherever possible.

(vi) The maximum number of characters to be used in questions may be specified.

(vii) Unknown or missing character values are allowed for throughout.

(viii) Detects redundant characters and gives error messages if no key can be determined.

(ix) Program will print out the object-character value matrix (synopsis) in a readable form.

(x) When an object keys out, all the remaining distinctive characters for that object are included.

(xi) The program prints the final key in a form which is ready for use, except that a little editing might be needed to improve the literary style.

Data preparation

There is good reason to try and use an object-character matrix which is as complete as possible. A matrix with too few characters, even if complete, may be insufficient to produce a key. A matrix with an adequate number of characters may still not be sufficient for a key if too many of the values are missing. The initial requirement is that at least one character should have a complete set of values.

The user assigns a positive numerical index to the string which represents each character, and similarly for the values. A zero index is used for missing values. The program internally assigns an index to each object. The character weights are positive integers. The largest weight is given the most importance. Some attention has to be paid to the question of style when using characters and values. Suppose character 23 represents flower colour, when a key for flowering plants is being produced. Then if FLOWER COLOUR is the string used for character 23 and in one case it has value 15, which is RED, then the phrase FLOWER COLOUR RED might appear in the key. In such a case it is better to write the string for character 23 as FLOWERS so that FLOWERS RED results.

The user should beware of using 'or' in the values of characters. It is possible to say of a number of species of animal for example, 'fur brown or grey'. This is true of a group of different objects but is not necessarily true of the individuals within it. 'Fur brown or grey' could appear in the key where a particular animal keys out when the animal only ever has brown fur. Thus, although the key works correctly, it could give a false impression of the characters of an object.

Experience with preparing data from descriptions of biological species shows that when the original data specifying the classification is not presented entirely systematically then much effort has to be spent on trying to complete the matrix. This situation arises because the traditional classification methods of biologists are partly subjective, so that the manner in which descriptions of species have been set out has not been specially rigorous. The characters to be used by the key program have to be selected from the written descriptions of the species, and this may have to be an iterative process. The best procedure seems to be to make up a chart of the species and their characters and to assign indices to the characters and values last of all.

Key generating algorithm

A simplified flow diagram of the algorithm is given in **Fig. 4.** The process as a whole may be described as one algorithm which generates another. The algorithm generated is the key itself, and this is a recognition procedure which is optimal in some sense and therefore efficient. The key generating algorithm chooses heuristically between possible ways of branching a key by using a comparison function.

Initially, all the objects and characters can be considered. The objects can be divided into groups by considering the values of the characters, supposing first of all that the characters are all equally weighted. Only those characters which have no missing values within the current set of objects can be used. Suppose one character is selected and it has two different values. This will then divide the objects into two groups. Two or more characters could be selected and these with their values can be used to group the objects, usually giving a higher number of groups. The maximum number of characters that can be considered is always known. The selection of subsets is achieved by an algorithm that generates all the possibilities in turn. For example, out of a maximum of six characters one might want all subsets of size three. There are then ${}^6C_3 = 20$ different sets, such as 123, 124, 125, 126, 134, 135, 136, etc. Hence, all the possible trees can be enumerated.

Subsequently, at later nodes of the key, some of the characters will have already been used and also some of the objects may have been eliminated. Hence, only a sub-matrix of objects and characters has to be considered. If the characters are weighted differently, then the highest weighting among the available characters is found, and only the characters with this weighting are

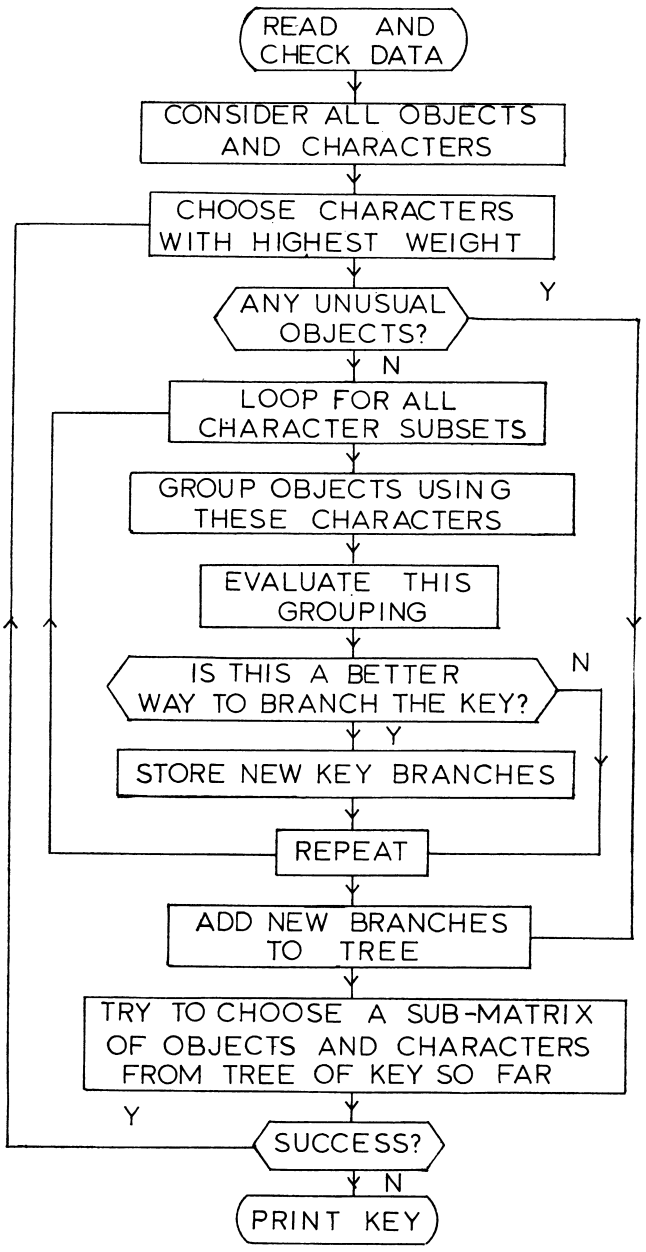

Fig. 4. Flow diagram of key generating program

used to generate new key branches. If these characters turn out to be redundant, in that their values are all identical or missing amongst the relevant subset of objects, then the characters with the next highest weighting are considered instead. This is repeated until some useful character is found. When characters are considered in combinations, only sets of characters with equal weighting are chosen.

The comparison function F was chosen to have the following properties. The function has a minimum value of zero when (i) it divides a group of objects into two, and (ii) the sizes of the sub-groups of objects so divided are equal. Suppose N objects are divided into K sub-groups with n_i objects in each sub-group, where $i = 1, 2, \ldots, K$. Many suitable functions exist, and the following simple function is used and has, so far, been quite satisfactory:

$$F = F_1 + F_2$$

where

$$F_1 = (K - 2)^2,\ F_2 = \sum_{i=1,K} \left| 1 - \frac{n_i K}{N} \right|$$

The ideal case is clearly $K = 2$, $n_1 = n_2 = N/2$, then $F = 0$. F_1 increases rapidly for $K > 3$. F_2 also tends to increase for larger K, as well as being larger when the sub-groups are unequal. The function F has another useful property. Suppose a number of possible ways of grouping the objects with the characters have already been attempted, giving a minimum F value of F_{min}. K_{max} can be calculated from F_1 such that K_{max} is the smallest possible value of K which gives an $F > F_{min}$. In other words, for a given F_{min}, all ways of dividing the objects which give K_{max} or more sub-groups of objects can be rejected, since they cannot give a better value for F_{min}. The key program enumerates the sub-groups by considering first one character and the corresponding sub-groups, and then two characters, and so forth. There is then a high probability that the best arrangement will be found almost at once, and then the remaining possibilities can be quickly rejected. The key program does not attempt to optimise the key over more than one question or test at a time.

The data structure used to store the key as it is built is basically a simple list structure which corresponds to the tree of the key. Each node of the tree is represented by two blocks of storage of variable size linked together. At later stages of the key generating process extra pointers are added to the structure in order to represent the question labels before and after the question, and then the structure is no longer a simple list structure, since there may be more than one way of reaching a node from the root of the tree.

A pushdown stack is used in the process of selecting a new sub-matrix of objects and characters when new branches are added to a node. The index of the stack is the level of the node in the tree. The stack index is also used to indent the key when it is printed out by moving the questions to the right by the number of spaces given by the index. The sub-matrix is found by taking the table of objects which is stored in the data structure for the node, and by eliminating all characters which are used in the path between the root of the tree and the current node. The two types of key are obtained by a simple process of reordering the list structure. An exception to this occurs with keys of the second type, where it is desired to have the smallest group first. In this case the branches from a node are put in ascending order according to the number of objects in each group.

The detection of unusual objects is carried out by a simple and crude classification technique. Many more refined numerical techniques of taxonomy exist (Sokal and Sneath, 1963) but their use was not thought appropriate here. No attempt to detect an unusual object is made if any of the values of the available characters are missing, since the comparison technique requires an equal number of characters for each object. Each object is compared with every other object and a count is made of the number of characters which correspond. This count when averaged out gives an 'average likeness'. On the assumption (unjustified) that the 'likeness' values have a uniform distribution, a variance is calculated. Any object whose 'likeness' differs from the average by more than the heuristically derived standard deviation is said to be unusual. Unusual objects are then keyed out immediately without the usual tree searching process.

The whole algorithm has been written in ASA FORTRAN, for the sake of potential users outside the computer centre of origin. FORTRAN was unsuitable for writing the key generating algorithm in the following respects:

(i) the handling of alphanumeric input/output data is very clumsy;
(ii) a considerable amount of storage is wasted in implementing the dynamic storage allocation scheme for the list structure using FORTRAN arrays. This type of programming is usually carried out using absolute addresses as pointers, whereas within FORTRAN an array index has to be used instead.

The LEAP language and data structure (Feldman and Rovner, 1969) might have been more suitable for programming the key program, had it been available. LEAP is an ALGOL-type language with a data structure which stores relationships of the form 'Attribute of Object is Value', abbreviated as Attribute (Object) = Value. This is precisely the same form as the description of objects in terms of the values of the characters (attributes) as used by the key program. As in the earlier example, the ability of a duck to fly could be expressed as:

Ability to fly (Duck) = Yes.

Performance

The computing time taken depends principally on the number of objects, the number of characters, and the maximum number of characters allowed per question. Both the following examples of timing are given for a maximum number of characters per question of two. The key reproduced in Fig. 3 took about two minutes on the Cambridge University TITAN computer (two microsecond cycle time) for 29 characters and 18 objects. A key for 32 characters and 140 objects took about seven minutes. The FORTRAN program, not including storage space, occupies 16K words. 3K words of storage were sufficient for working space and storage of the complete key for a matrix of about 30 by 30. The program runs noticeably faster with the use of unequal

character weighting, and when there are missing character values, since both these cases reduce the number of alternative keys to be considered.

At the present time, the program has been distributed to a number of users, of whom only a few have so far given their reaction. However, two notable successes can be claimed. In the first case, the author was challenged to reproduce by program a key in the Flora Europaea (Tutin, Heywood, Burges, Moore, Valentine, Walters and Webb, 1968). The key was for the European Genera of the Umbelliferae (carrot family) and involves 134 distinct objects. The resulting key had 20% fewer leads than the hand-made original, the leads were more compact and fewer characters were used.

A second successful application has been to produce a key for 86 cultivated varieties of potato with data provided by Mr. T. Webster of the National Institute for Agricultural Botany. No key had previously been available for the potatoes, since the task of constructing one by hand had proved daunting. The computer generated key provides the first systematic identification scheme to become available in this case.

Conclusions

The key generating program should prove very useful to biologists, geologists and others. The amount of effort required to set out the original data matrix remains, but once this is obtained, key revision is very easy when new objects or characters or a new weighting of the relative importance of characters is required. There is also reason to think that the key obtained is likely to be the best possible, rather than just any key. The user of this program should be able to obtain a greater variety and quality of keys than before. The principal effort that remains is in data preparation.

Acknowledgements

Thanks are due to Dr. S. M. Walters of the Herbarium of the Botany School at Cambridge for making practical suggestions on the design objectives of the program, to L. E. Morse for correspondence on his own work, and to the referee for bringing some of the artificial intelligence research to the author's notice.

References

Feldman, J. A., and Rovner, P. D. (1969). An ALGOL-based associative language, *CACM*, Vol. 12, No. 8, pp. 439–449.
Goodall, D. W. (1968). Identification by computer, *Bioscience*, Vol. 18, pp. 485–488.
Hunt, E. B., Marin, J., and Stone, P. J. (1966). *Experiments in induction*, Academic Press: New York and London.
Leenhouts, P. W. (1966). Keys in biology, Proceedings, Koninklijke (Nederlandsche) Akademie van Wetenschappen, Vol. 69C, pp. 571–596.
Metcalf, Z. P. (1954). The construction of keys, *Systematic Zoology*, Vol. 3, pp. 38–45.
Moller, F. (1962). Quantitative methods in the systematics of the Actinomycetales. IV. The theory and application of a probabilistic identification key, *Giornale Microbiologica*, Vol. 10, pp. 29–47.
Morse, L. E., Beaman, J. H., and Shetler, S. G. (1968). A computer system for editing diagnostic keys for Flora North America, *Taxon*, Vol. 17 (5), pp. 479–483.
Morse, L. E. (1968). Abstract, Construction of identification keys by computer, *American Journal of Botany*, Vol. 55, p. 737.
Morse, L. E. (1969). Private communication.
Niemela, S. I., Hopkins, J. W., and Quadling, C. (1968). Selecting an economical binary test battery for a set of microbial cultures, *Canadian Journal of Microbiology*, Vol. 14, pp. 271–279.
Osborne, D. V. (1963). Some aspects of the theory of dichotomous keys, *New Phytologist*, Vol. 62, pp. 144–160.
Slagle, J. R. (1964). An efficient algorithm for finding certain minimum-cost procedures for making binary decisions, *JACM*, Vol. 11, pp. 253–264.
Sokal, R. S., and Sneath, P. H. A. (1963). *Principles of numerical taxonomy*, W. H. Freeman & Co., San Francisco and London.
Tutin, T. G., Heywood, V. H., Burges, N. A., Moore, D. M., Valentine, D. H., Walters, S. M., and Webb, D. A. (Editors) (1968). *Flora Europaea*, Vol. 2, Cambridge University Press.
Voss, E. G. (1952). The history of keys and phylogenetic trees in systematic biology, *Journal of the Science Laboratory of Denison University*, Vol. 43, pp. 1–25.
Winston, P. (1969). A heuristic program that constructs decision trees, Artificial Intelligence Memorandum No. 173, Project MAC, M.I.T.