Data Subset Selection by Boolean Calculation

R. C. BRILL
The University of Michigan Computing Center, 1075 Beal Avenue, Ann Arbor, Michigan 48109

AND

G. F. ESTABROOK
The University Herbarium and Botany Department, University of Michigan, Ann Arbor, Michigan 48109

Received 25 March 1984

ABSTRACT

In this journal in 1969 the authors described an unconventional method, based on concepts from systematic biology, for storing and retrieving data, whose salient features are: (1) data are stored in such a manner that the space required for their storage is close to the information-theoretic minimum; and (2) the selection of data subsets for retrieval is effected by performing Boolean arithmetic directly on the stored data. Here we describe new algorithms to increase the power of data subset selection in the context of this method and describe their potential for parallel processing.

1. INTRODUCTION

The data structure described in this paper, as well as Algorithm A and an earlier version of Algorithm B, embodied in an information-management system called TAXIR, was first described by us in this journal in 1969 [8]. TAXIR was originally designed with concepts from the methods of biological classification, and was first used for the curation of biological collections and as a monographic tool. Today, TAXIR serves approximately a dozen user communities in North America, South America, and Europe, and is still undergoing development [6, 10]. We designed and programmed the first version of TAXIR to test the method and were encouraged by its performance to develop it further. First at the University of Colorado [4] and later at the University of Michigan [5] the system became more oriented toward user power and convenience. Gradually it came to include a high-level user language for data entry, correction and querying, as well as report-generation capabilities that permit ordered, labeled, formatted output, including options

0025-5564/84/$03.00

for totals, subtotals, and N-way tabulations. Several variants of the system have been described [3, 7, 15, 17], and a number of interesting applications have been reported [1, 2, 9, 11–13, 16, 18–21, 23].

The motivation for the present paper came with the development of an improved version of Algorithm B, and two new algorithms, C and D. This gave us the opportunity to rephrase our description of the method in the context of today's TAXIR, which has evolved considerably over the years. We are further motivated by the current interest in database machines and parallel processing, inasmuch as we believe this method to be well suited to a parallel-processing environment.

2. BACKGROUND INFORMATION

This paper is concerned only with the subset-selection capability of the TAXIR system. This section presents the minimal user-level background information needed to follow the argument. For a complete user-level description of TAXIR see Brill [5].

A *data bank* is a set of *items*, each item corresponding to one entity in some collection of interest, such as the specimens of a museum, the patients in a hospital, the responses to a medical treatment, specimens in a genus under study, etc. The pieces of information that characterize an item belong to information categories called *descriptors*. A descriptor partitions the set of items into mutually exclusive subsets called *descriptor states* or just *states*. For example, if the items represent plant collections, the descriptor **MONTH OF COLLECTION** will have the states **JAN**, **FEB**, etc., and will partition the data bank in the obvious way; e.g., the state **JAN** will have as members all the plants collected in January.

A user creates a data bank by defining the relevant descriptors. Then data may be entered at will. An item is entered by listing all the states to which the item has been assigned, one state for each descriptor defined. A data bank can be thought of as a two-dimensional matrix of items versus descriptors, with a descriptor state at each coordinate. Thus, TAXIR is a "flat file" database system: no hierarchical structures, no multiple occurrences. See [6] for a discussion of these limitations. For the convenience of the user and for ease of internal processing, TAXIR assigns each descriptor a unique code number. The code numbers are the consecutive integers starting with 1.

TAXIR supports three types of descriptors.

FROM-TO: This type is for numeric data. At data bank creation time, the user defines a range and an increment (e.g., **FROM -100 TO 100 BY .01**).

ORDER: This type is for alphanumeric data where the complete set of states is known in advance. At data bank creation time, the user supplies a fixed ordered list of states (e.g., **JAN, FEB, MAR, APR, MAY, JUN, JUL, AUG, SEP, OCT, NOV, DEC**).

NAME: This type is for alphanumeric data where the set of states is not known in advance. Any value supplied in the data becomes a valid state on entry, and the list of states may grow or diminish throughout the life of a data bank.

In addition to the states defined by the user, TAXIR recognizes in each descriptor a state called UNKNOWN, as a catchall for missing information. TAXIR assigns a unique code number to each state of a descriptor. The code number 0 is assigned to the UNKNOWN state, and consecutive integers are assigned to the rest of the states.

When retrieving data, a TAXIR user employs a Boolean expression to specify the subset of items desired. The simplest operand in such a Boolean expression takes the form

desc relop ds

where desc is a descriptor name or code number;

where relop is one of the relational operators $=$, $\neq$, $>$, $\geqslant$, $<$, $\leqslant$ ($=$ and $\neq$ may be applied to any descriptor; the other operators may be applied only to FROM-TO or ORDER descriptors); and

where ds is a descriptor-state name or code number, defined for descriptor desc.

The Boolean operators NOT, AND, and OR may be used, with that order of precedence. Also parentheses, nested to any depth, may be used to override the order of precedence.

Example 1. Some simple Boolean expressions in TAXIR:

```
GENUS = ROSA
LEAF LENGTH > 40
YEAR = 1980 AND MONTH > JUN AND NOT (MONTH = JUL AND
  DAY = 4)
```

3. INTERNAL STORAGE OF DATA

The data entered into TAXIR are stored in *datasets*, one for each descriptor. A dataset contains for each item in the data bank the code number of the state to which the item has been assigned. For descriptors of type ORDER or NAME, a dictionary of state names and code numbers is maintained. For FROM-TO descriptors, no dictionary is needed, as the correspondence between state names (i.e., numeric values) and code numbers is established by a simple arithmetic transformation. The descriptor-state code numbers are stored in an unconventional manner.

Let S be a data bank.

Let D_d be the dataset for the dth descriptor of S.

Let Z be the number of items in S.

Let N_d be the number of bits in the binary representation of the largest code number for descriptor d.

Then the size of D_d is Z by N_d bits, that is, Z columns by N_d rows. The code number for the zth item is stored in the zth column. Note that the bits of any one code number are not contiguous in storage, but are the corresponding bits of separate computer words. A dataset can be seen as a set of code numbers hanging by their low-order bits like bats hanging by their feet in a cave.

Example 2. **MONTH** is descriptor 16 in some hypothetical data bank S. The 8 items of S are assigned to the following states of **MONTH**: **JAN**, **FEB**, **MAY**, **UNKNOWN**, **DEC**, **JUL**, **MAY**, **OCT**. The code numbers for those states are 1, 2, 5, 0, 12, 7, 5, 10.

D_{16}, of size $Z = 8$ by $N_{16} = 4$, will look like this:

10100110
01000101
00101110
00001001

This data structure makes possible the retrieval algorithms described in the next sections. It also achieves a compactness approaching the information-theoretic minimum. With no prior knowledge of the frequency distribution of the items through the states, the amount of information needed to designate to which one of m descriptor states an item belongs is known to be $\log_2 m$ bits. If we wish to realize the advantages of storing information in whole bits, then the minimum number required will be the least integer $\geqslant \log_2 m$, which is N_d, the number of bits used per item by this data structure. For a more detailed discussion of storage efficiency, see [8].

4. SUBSET SELECTION

TAXIR scans a user's Boolean expression from left to right and converts it to an internal Boolean expression, whose operands are dataset rows. This internal expression is then translated to Polish notation, eliminating parentheses and reducing the size of certain expressions. In this form it is used to drive the query execution processor, which simply performs the Boolean calculation requested by the expression on the appropriate dataset rows.

The result of this calculation is a bit string of length Z, called a *result string*, which describes the membership of the subset requested by the user. Summing the ones in the result string yields the number of items in the result. Since the zth dataset column contains the code number of the state to which

item z has been assigned, then a one in position z of the result string indicates that item z belongs to the result and the value z is the address to all the information stored about item z. Savage [22], in discussing Boolean expressions at an abstract level, has provided a sound mathematical base for comparing integer values one bit at a time. In this spirit we construct four algorithms, whose practical application arises in the special context of the TAXIR data structure described above.

For clarity, the algorithms are written in a simple, hypothetical programming language, which should be readily understandable to readers.

A. ALGORITHM A

The simplest Boolean expression in TAXIR contains no Boolean operators and consists merely of an operand of the form "desc = ds", as in the expression **MONTH = MAY**. The following, as well as all previous, definitions apply:

Let C_i denote the ith row of D_d.

Let DSC be an integer variable used to hold descriptor-state code numbers.

Let DSC_i be the ith bit of the code number in DSC.

Let T be a character-string variable.

Let || be the character-string concatenation operator.

Let a character string be delimited by single quotes (e.g., 'NOT').

When TAXIR recognizes an expression of the form "desc = ds", d is set to the code number for desc, DSC is set to the code number for ds, and then Algorithm A, which builds an internal Boolean expression in the variable T, is executed.

```
/* Algorithm A generates a Boolean expression to handle the case of
   "desc = ds". */
T := 'D_d ('
do i := 0 to N_d - 1
   if DSC_i = 0 then
      T := T || 'NOT'
   end if
   T := T || 'C_i'
   if i ≠ N_d - 1 then
      T := T || 'AND'
   end if
end do
T := T || ')'
```

Example 3. Building a Boolean expression of the form "desc = ds". Given:

the data bank of Example 2,
user's Boolean expression: **MONTH = MAY**,
code number for **MAY**: $5 = 0101_2$.

Executing Algorithm A on the expression **MONTH = MAY** yields the internal Boolean expression

$$D_{16}\ (C_0 \text{ AND NOT } C_1 \text{ AND } C_2 \text{ AND NOT } C_3)$$

The appearance of the term D_{16} serves as a signal to the query execution processor to address the correct dataset. Executing this expression on dataset D_{16}, illustrated in Example 2, yields the result string 00100010, which tells us that items 3 and 7 belong to the subset requested.

B. *ALGORITHM B*

A second algorithm is used for Boolean expressions of the type "desc ⩾ ds", as in **MONTH ⩾ OCT**. The same definitions apply as before.

```
/* Algorithm B generates a Boolean expression to handle the case of
   "desc ⩾ ds". */
i := 0
until DSC_i = 1
   i := i + 1
end until
T := '(C_i'
i := i + 1
until i = N_d
   if DSC_i = 1 then
      T := '(' || T || ') AND C_i'
   else
      T := T || 'OR C_i'
   end if
   i := i + 1
end until
T := D_d || T || ')'
```

Example 4. Building a Boolean expression of the form "desc ⩾ ds". Given:

the data bank of Example 2,
user's Boolean expression: **MONTH ⩾ OCT**,
code number for **OCT**: $10 = 1010_2$.

We get the internal Boolean expression

$$D_{16}\ ((C_1 \text{ OR } C_2) \text{ AND } C_3)$$

We get the result string 00001001 (i.e., items 5 and 8).

The reader is left to work out how the other relational operators ($\neq$, $>$, $<$, $\leqslant$) may be handled using these two algorithms as a basis, mentioning only that for $<$ and $\leqslant$ some provision must be made to exclude from the result items assigned to the UNKNOWN state (code = 0), as this state is not considered to be part of any range.

C. *ALGORITHM C*

Recently the authors discovered that a form of Boolean operand, hitherto unknown in TAXIR, can be usefully incorporated into the system:

desc1 relop desc2

where desc1 and desc2 are two descriptors from the same data bank whose states are defined to be identical and

where relop is defined as before, to be the relational operators =, ≠, >, ⩾, <, ⩽.

Here are some additional definitions for use in the algorithms below.

Let Relop, x, and y be the names of integer variables.

Let Equal, Not_Equal, Greater, Greater_or_Equal, Less, and Less_or_Equal be the names of integer constants whose values are irrelevant as long as they are distinct from each other.

When TAXIR recognizes an expression of the form "desc1 relop desc2", Relop is set to the appropriate constant (i.e., Equal, Not_Equal, etc.); x is set to the descriptor code number of desc1; and y is set to the descriptor code number of desc2. If the operator is = or ≠, TAXIR then executes Algorithm C.

```
/* Algorithm C generates a Boolean expression to handle the cases of
   "desc1 = desc2" and "desc1 ≠ desc2". */
if Relop = Equal then
   T = 'NOT ('
else
   T = '('
end if
do i = 0 to N_x - 1
   T = T || '(D_xC_i XOR D_yC_i)'
   if i ≠ N_x - 1 then
      T = T || 'OR'
   end if
end do
T = T || ')'
```

Example 5. In a data bank of biological species types containing both collector and authority, we can ask which types were collected by one worker but named by another. This is the appropriate Boolean expression:

COLLECTOR ≠ AUTHORITY

Assuming that **COLLECTOR** has descriptor code 30 and **AUTHORITY** has descriptor code 40, and that the states of both descriptors are the 61 systematists who contributed collections or names to our type collection, then Algorithm C will generate the following internal Boolean expression:

$$\begin{aligned}&((D_{30}C_0 \text{ XOR } D_{40}C_0) \text{ OR } (D_{30}C_1 \text{ XOR } D_{40}C_1) \text{ OR}\\&(D_{30}C_2 \text{ XOR } D_{40}C_2) \text{ OR } (D_{30}C_3 \text{ XOR } D_{40}C_3) \text{ OR}\\&(D_{30}C_4 \text{ XOR } D_{40}C_4) \text{ OR } (D_{30}C_5 \text{ XOR } D_{40}C_5))\end{aligned}$$

D. ALGORITHM D

Algorithm D handles the case of "desc1 relop desc2" when relop is $>$, $\geqslant$, $<$, or $\leqslant$. This algorithm is derived from [22], which gives a simple Boolean function that determines when a two-bit binary integer, x, is greater than another such integer, y:

$$x_1 \bullet \bar{y}_1 + (x_1 \bullet y_1 + \bar{x}_1 \bullet \bar{y}_1) \bullet x_0 \bullet \bar{y}_0$$

where $\bar{}$ denotes NOT, $\bullet$ denotes AND, and $+$ denotes OR. This can be reduced to the more economical form:

$$x_1 \bullet \bar{y}_1 + (\overline{x_1 \oplus y_1}) \bullet x_0 \bullet \bar{y}_0$$

where $\oplus$ denotes "exclusive or". We have generalized this idea to the case of TAXIR, where we compare not one pair, but many pairs, of integers (i.e., descriptor-state codes) simultaneously, and where the integer lengths need not be limited to two bits. However, note that a one-bit code in TAXIR represents a descriptor with only two states, 0 (UNKNOWN) and 1 (some other state). Since UNKNOWN is not part of the range of state values, range comparisons ($>$, $\geqslant$, $<$, $\leqslant$) are not permitted for descriptors with only one known state.

```
/* Algorithm D generates a Boolean expression to handle the cases of
   "desc1 > desc2", "desc1 ≥ desc2", "desc1 < desc2", and "desc1 ≤ desc2".
   */
T := '('
if Relop = Greater or Relop = Greater_or_Equal then
   do i := N_x - 1 to 1
```

```
      T := T || 'D_xC_i AND NOT D_yC_i OR
          NOT (D_xC_i XOR D_yC_i) AND ('
    end do
    if Relop = Greater then
      T := T || 'D_xC_0 AND NOT D_yC_0'
    else
      T := T || 'D_xC_0 OR NOT D_yC_0'
    end if
  else
    do i := N_x - 1 to 1
      T := T || 'NOT D_xC_i AND D_yC_i OR
          NOT (D_xC_i XOR D_yC_i) AND ('
    end do
    if Relop = Less then
      T := T || 'NOT D_xC_o AND D_yC_0'
    else
      T := T || 'NOT D_xC_0 OR D_yC_0'
    end if
  end if
  do i := N_x - 1 to 1
    T := T || ')'
  end do
  /* Eliminate items in the UNKNOWN state. */
  T := T || 'AND ('
  if Relop = Greater or Relop = Greater_or_Equal then
    T := T || 'D_y'
  else
    T := T || 'D_x'
  end if
  do i := 0 to N_x - 1
    T := T || 'C_i'
    if i ≠ N_x - 1 then
      T := T || 'OR'
    end if
  end do
  T := T || '))'
```

Example 6. In a data bank of plant specimens, one descriptor, called **PETAL LENGTH** (code no. 21), has states UNKNOWN, 6 mm, 7 mm, 8 mm, 9 mm, and 10 mm. Another descriptor, called **STAMEN LENGTH** (code no. 22), has the same states. To find out which plants have longer stamens than

TABLE 1

Specimen name	Petal length (mm)	Stamen length (mm)	Petal color
S1	10	10	RED
S2	8	9	WHITE
S3	—	—	—
S4	8	—	BLUE
S5	7	6	BLUE
S6	8	7	WHITE
S7	—	8	—
S8	9	10	RED
S9	10	9	WHITE
S10	8	8	BLUE

petals, we can use the Boolean expression

STAMEN LENGTH > PETAL LENGTH

These two descriptors have the same states, defined in the order given. TAXIR will assign the state codes as follows: UNKNOWN = 0, 6 mm = 1, 7 mm = 2, 8 mm = 3, 9 mm = 4, 10 mm = 5. Let's suppose the data are as shown in Table 1, where a dash represents the UNKNOWN state (i.e., plant part not measured). Then the expression **STAMEN > PETAL LENGTH** will cause the system to generate the internal Boolean expression

$$(D_{22}C_2 \text{ AND NOT } D_{21}C_2 \text{ OR NOT } (D_{22}C_2 \text{ XOR } D_{21}C_2) \text{ AND}$$
$$(D_{22}C_1 \text{ AND NOT } D_{21}C_1 \text{ OR NOT } (D_{22}C_1 \text{ XOR } D_{21}C_1) \text{ AND}$$
$$(D_{22}C_0 \text{ AND NOT } D_{21}C_0)) \text{ AND } (D_{21}C_0 \text{ OR } C_1 \text{ OR } C_2))$$

when executed on the specimen data bank shown in Table 1, the result string will be 0100000100 (i.e., specimens S2 and S8).

E. COMBINING EXPRESSIONS

To show how more complex Boolean expressions may be built up, an additional rule is given: When TAXIR recognizes Boolean operators or parentheses in the user's Boolean expression, it appends them unchanged to the internal Boolean expression T.

Example 7. In the data bank of Example 6, another descriptor called **PETAL COLOR** (code no. 46) has states (codes) UNKNOWN (0), RED (1), WHITE (2), BLUE (3). The Boolean expression

STAMEN LENGTH > PETAL LENGTH OR PETAL COLOR = RED

causes TAXIR to generate the internal Boolean expression

$$(D_{22}C_2 \text{ AND NOT } D_{21}C_2 \text{ OR NOT } (D_{22}C_2 \text{ XOR } D_{21}C_2) \text{ AND}$$
$$(D_{22}C_1 \text{ AND NOT } D_{21}C_1 \text{ OR NOT } (D_{22}C_1 \text{ XOR } D_{21}C_1) \text{ AND}$$
$$(D_{22}C_0 \text{ AND NOT } D_{21}C_0)) \text{ AND } (D_{21}C_0 \text{ OR } C_1 \text{ OR } C_2)) \text{ OR}$$
$$D_{46}(C_0 \text{ AND NOT } C_1)$$

When executed on the specimen data bank of Example 6, the result string will be 1100000100 (i.e., specimens S1, S2, and S8).

5. DISCUSSION

Although TAXIR is very efficient at selecting data subsets [14], the new algorithms described here will contribute further to its speed in data subset selection. They will also add to the power of the TAXIR system to execute a wider variety of queries.

An additional value of these algorithms lies in their suitability for parallel processing. In a typical hardware environment, such as that provided by a 32-bit processor, the Boolean operands are 32 bits long, which means that the algorithms can calculate a result string for 32 items simultaneously. This power can be greatly extended through the use of special hardware capable of performing simple Boolean arithmetic in parallel on very long operands. We envision a TAXIR machine in which the storage device that holds the datasets is endowed with the ability to execute a small repertoire of instructions on the data stored there. Such a machine has not yet been designed or built, but in such an environment, the algorithms described above would provide extremely rapid retrieval of large datasets, using arbitrarily complex search criteria.

We would like to offer our thanks and appreciation to Dr. David J. Rogers, who gave TAXIR *its first home at the University of Colorado; to Dr. George W. Nace, who brought* TAXIR *to the University of Michigan; to Dr. Robert Bartels, who gave* TAXIR *the support of the University of Michigan Computing Center; and to Steven Tolkin, who offered valuable advice during the preparation of this paper.*

REFERENCES

1 C. Anzaldi and L. Mirri Passerini, *Un esperimento di strutturazione di dati floristici e vegetazionali*. Istituto per le Applicazioni del Calcolo, Pubblicazioni Serie III, N. 205, Roma, 1979.

2 F. A. Bisby, and M. T. Babac, A chemotaxonomic database, in *Databases in Systematics* (R. Allkin and F. A. Bisby, Eds.), Academic, 1984.

3 R. Borsuk, W. Dyer, and J. R. Hanley, P. Legendre, D. J. Rogers, B. A. Sim, L. Vincent, and D. Watt, *EXIR User's Manual*. Information Sciences/Genetic Resources Program, Univ. of Colorado, Boulder, 1976.

4 R. C. Brill, The *TAXIR Primer*, Occasional Paper No. 1, Inst. of Arctic and Alpine Research, Univ. of Colorado, Boulder, 1971.
5 R. C. Brill, The *TAXIR Primer*, MTS Version, 4th ed., Univ. of Michigan Computing Center, Ann Arbor, 1983.
6 R. C. Brill and G. F. Estabrook, Management of Almost Flat Files in Systematic Biology using TAXIR, *in Databases in Systematics* (R. Allkin and F. A. Bisby, Eds.), Academic, London, 1984.
7 R. G. Chenhall, *Museum Cataloging in the Computer Age*, American Association for State and Local History, Nashville, Tenn., 1975.
8 G. F. Estabrook and R. C. Brill, The theory of the TAXIR accessioner, *Math. Biosci.* 5:327–340 (1969).
9 G. F. Estabrook, A TAXIR data bank of flowering plant types at the University of Michigan Herbarium, *Taxon* 28:197–204 (1979).
10 G. F. Estabrook, Collaboration for the future development of TAXIR, *Territorio* 2.2:45–55 (1982).
11 J. V. Fayos and R. C. Brill, Computer retrieval and analysis of radiation therapy data, *Univ. of Michigan Medical Center J.* 39(4):159–165 (1973).
12 J. V. Fayos, Computerized radiation therapy data, *Cancer* 37(6):2736–2745 (1976).
13 L. W. Hudson, R. D. Dutton, M. M. Reynolds, and W. E. Walden, TAXIR—a biologically oriented information retrieval system as an aid to plant introduction, *Economic Botany* 25(4):401–406 (1971).
14 M. A. Kahn, *A Benchmark of Three MTS Database Management Systems—MICRO, SPIRES, TAXIR*, PRISM Associates, Ann Arbor, Mich., 1975.
15 E. W. Klaphake, *STIRS User's Manual*, Univ. of Colorado Computing Center, Boulder, 1973.
16 J. I. LaRue, Using a local computer for garden records management, *Amer. Assoc. Bot. Gard. Admin. Bull.* 13:16–18 (1979).
17 P. Legendre, *EXIR 3.0, Guide de L'Usager*, Univ. du Quebec, Montreal, 1978.
18 D. E. Moerman, *American Medical Ethnobotany, A Reference Dictionary*, Garland, 1977.
19 D. E. Moerman, Symbols and selectivity: A statistical analysis of native American medical ethnobotany, *J. Ethnopharmacology* 1:111–119 (1979).
20 G. W. Nace, C. M. Richards, and G. M. Hazen, Information control in the Amphibian Facility: the use of *R. pipiens* disruptive patterning for individual identification and genetic studies, *Amer. Zool.* 13:115–135 (1973).
21 S. Pignatti, *Checklist of the Flora of Italy with Codified Plant Names for Computer Use*, Consiglio Nazionale delle Ricerche, Roma, 1981.
22 J. E. Savage, *The Complexity of Computing*, Wiley, 1976.
23 A. R. Teixeira and C. P. Spiguel, Banco de dados do Programa Flora do CNPq, sobre plantas medicinais e farmacologia de produtos naturais, *Ciencia e Cultura* 32:48–58 (1980).