searchable, more precision would be possible and this mode of searching would be more valuable. The complete U.S. classification number is given for U.S. patents. However, since U.S. patents issue more slowly than their foreign equivalents, relatively few U.S. patents are abstracted in *Chemical Abstracts*. Therefore, a search of U.S. class numbers normally results in the retrieval of only a small fraction of the pertinent patents covered in *Chemical Abstracts*. It should be mentioned that the Condensates record also includes the patent classification used on Japanese Kokai.

The availability of the *Chemical Abstracts Patent Concordance* in an on-line system would provide quicker and better access to equivalent patents. Currently a manual search must be conducted through the volume and weekly patent concordances. Often an on-line search will not retrieve a previously known patent, and the only way to determine if an equivalent patent was indexed is by a manual search.

Another technique that I have found useful is to save a copy of the news and update information that is printed on-line. This allows me to update searches more easily and to answer questions about search coverage from requesters at a later date.

Often it is advantageous to search on the *Chemical Abstracts'* section number, as well as the appropriate index terms. I have found many useful references in this manner which were not subject-indexed to the desired concept. An example would be a search on Section 5 for articles on structure–activity studies for a question about pesticidal structure–activity relationships.

We are continuing to use DIALOG and ORBIT to search Condensates and the other available data bases. Our experience shows that both systems provide good service and are reliable. It is my opinion that ORBIT and DIALOG are valuable services and that all organizations providing computerized bibliographic retrieval service should utilize both.

Computerized Chemical Information Retrieval Techniques†

DANIEL U. WILDE

New England Research Application Center, University of Connecticut, Storrs, Connecticut

Received April 3, 1975

The introduction of powerful, low-cost computers now make it possible for most organizations to perform SDI and retrospective searching. The use of such a machine, located at the New England Research Application Center (NERAC) was reported earlier. New improvements to NERAC's SDI Chemical Search System are now discussed.

INTRODUCTION

For the past eight years, the New England Research Application Center (NERAC) has operated as a NASA Industrial Application Center at the University of Connecticut. The Center aids and promotes technology transfer by helping business and industry, colleges and universities, and state and local governments locate appropriate scientific and technical information. During this period, NERAC has performed some 8000 retrospective searches, while its data base has grown to over 2,250,000 documents. This data base now includes retrospective searching of nine files: National Aeronautics and Space Administration (NASA), Engineering Index (COMPENDEX), Department of Defense (DDC), Government Reports Abstracts (NTIS), Education Resources Information Center (ERIC), American Society for Metals (METADEX), Alloys Index (AI), World Aluminum Abstracts (WAA), and Abstracted Business Information (INFORM). Inspec and *CA Condensates* are also available for current awareness or Selective Dissemination of Information (SDI) bringing to eleven the total number of files available for SDI service.

When NERAC was established there were two choices for computational power. Time could be purchased on a large, extensive computer system, or a small machine could be rented for the exclusive use of the center. As was described earlier, NERAC decided to rent its own machine.[1,2] As a result, computer costs were fixed and increased computer usage decreased unit search costs. NERAC's present computer system consists of an IBM 370/115 with a 128K byte memory. Search questions are read via a 2501 card reader, and results are printed on a 5203 line printer. Data files are read from and intermediate results are saved on two 3410 tape drives with transmission rates of 80,000 characters per second at 1600 bits per inch. User programs and temporary data sets are saved on two 3340 disk drives with 140 million bytes of storage.

Random access searching of inverted files is not practical with such a limited machine. This is particularly true for large data bases, such as the NASA file with over 1,000,000 documents or Engineering Index with 400,000. Consequently, NERAC's data files are organized serially. This requires that all documents be examined individually in sequence during each search. Needless to say, search times can be expected to be especially long for character-by-character comparisons required for successful searching of *CA Condensates*.

Linear search methods are well known,[3] and the retrieval capabilities of NERAC's Chemical Search System have already been published.[4] This paper describes additional techniques for reducing linear search times and how they have been implemented on NERAC's small computer system without sacrificing retrieval capabilities.

† Presented before the Division of Chemical Literature in the Symposium on "User Reactions to CAS Data and Bibliographic Services," 169th National Meeting of the American Chemical Society, Philadelphia, Pa., April 7, 1975. This research was sponsored in part by the National Aeronautics and Space Administration, Contract NASW-2516.

REDUCING SEARCH TIMES

Equations that can be used to predict linear search times are shown in the Appendix. Analysis of these equations indicates that run time is a function of three sets of parameters. The *first* set is dictated by the data file and includes number of documents and average number of index terms per document. The *second* set is determined by the computer and includes average instruction execution time, start time of the input tape drive, and time to transmit one input character into memory. The *third* set is specified by the information center and includes average number of characters per input document and number of search terms per retrieval run. Let us now consider how these last two parameters can be varied by the information center so as to lessen search times.

The average number of input characters can be reduced by eliminating nonessential information from the data file. Document records are edited to conform to the needs and demands of center users. On a small machine with limited memory, reformating may require multiple passes. Here, the output tape is used for storage of intermediate results which become input to the next pass.

The number of search terms per retrieval run can be reduced by examining the construction of search strategies. For simplicity, assume Boolean logic where terms may be connected in any Boolean combination using *or's* (+), *and's* (&), and *not's* (¬). From the equation

$$S = A + B\&(C + D) + E\&F$$

it can be seen that if S is to be true, then one of the following abbreviated equations, S_i, must also be true.

$$S_1 = A + B + E \qquad S_2 = A + B + F$$

$$S_3 = A + C + D + E \qquad S_4 = A + C + D + F$$

If for a given document an S_i is false, then S must also be false. Therefore, S need only be evaluated for those documents for which an S_i is true.

This approach leads to a two-pass retrieval procedure. In the *first* pass, the complete file is searched using an abbreviated strategy, S_i. Documents are read one at a time from the input file. When a document satisfies S_i, it is copied onto the output file. If a document does not satisfy S_i, it is skipped since it cannot possibly fulfill S. After the complete file has been scanned, the output file contains the subset of documents that satisfies S_i.

It is important to note that the value of an abbreviated strategy is not affected by term sequence. This means that terms can be arranged so as to further reduce search time. Consequently, both strategy and document terms can be sorted so that rapid collating–lookup techniques can be employed to determine whether strategy terms match document terms.

During the second pass, the subfile that satisfies S_i is searched using the complete strategy, S. When a document is found that satisfies S, it is copied onto the output file along with a tag that identifies its requestor. Since this pass uses the complete strategy, it is important to minimize computation.

Examination of the sample Boolean equation indicates that S is true if A is true. This means that when S_i is satisfied because A is true, then S is also true. Consequently, during the first pass when a document is written onto the output file because A is true, that document can be flagged signifying that it also satisfies S. During the second pass, such documents can be copied directly onto the output *without* any second pass term comparisons. Note that this is not the case for any of the other terms in the sample strategy; every other term is intersected with at least one other term; only A stands alone.

A & (B+C+D+E) & F & ¬G

A	PHOTO*
B	EMULSION*
C	SEC 074
D	*DYE*
E	DANIZ, T*
F	KODAK CO*
G	INFRARED

Figure 1. Example profile.

Card No.	True Entry	False Entry	Term	Field Code	First Pass Code
1	01	98	PHOTO*		
2	04	01	EMULSION*		
3	03	01	074	S	
4	02	01	*DYE*		
5	01	98	DANIZ, T*	A	
6	01	98	KODAK CO*	C	1
7	98	98	INFRARED		

Figure 2. Profile coding.

In general, first pass run time is minimized by selecting an abbreviated strategy which contains the least number of terms. For the S shown above, this implies S_1 or S_2. Second pass execution time is minimized by selecting the abbreviated strategy which produces the smallest output during the first pass. Here, to make a final choice, relative occurrences of terms must be known.

PROFILE EXAMPLE

A hypothetical user's question concerns photography. He would like information relating to uses of dyes and emulsions in photographic work at Eastman Kodak Co. References to works by T. Daniz are of particular interest, but allusions to infrared are undesirable. Terms used to express this search are given in Figure 1. In the Boolean logic expression, terms are represented by letters. In this case, A represents the character string, PHOTO*, where the asterisk indicates truncation. C specifies Section 74 of *Chemical Abstracts,* and E specifies the desired author. Here, right truncation is used to overcome spelling and punctuation inconsistencies. The same is true for F which represents the company name.

Examination of the Boolean strategy in Figure 1 indicates that there are three possible abbreviated strategies.

$$S_1 = A$$

$$S_2 = B + C + D + E$$

$$S_3 = F$$

S_2 can be eliminated from consideration because it contains four terms requiring four times the number of first-pass comparisons as the other two. Furthermore, D contains left truncation which requires a character-by-character search eliminating the possibility of rapid collating–lookup term comparisons.

Final choice of an abbreviated strategy for the first pass is between PHOTO* or KODAK CO*. Here, it should be noted that PHOTO* is a single character string, while KODAK CO* requires that the word KODAK be immediately followed by or adjacent to a word that begins with the character string CO. This means that either KODAK or CO* can be used. Consequently, the final choice depends upon the relative frequency of occurrence of the three strings: PHOTO*, KODAK, or CO*. The least frequent term will retrieve the fewest documents during the first pass and consequently minimizes the number of documents that must be searched against the complete strategy during the second pass.

The coding of this same example profile for single-pass computer processing was discussed in detail in the earlier paper.[4] Figure 2 shows how a first-pass code is added to those cards that contain terms in the abbreviated strategy. Here, the first-pass code of one on the card containing KODAK CO* indicates that the first word on that card is a term in the first-pass strategy. If CO* had been selected, a code of two would have been used to indicate the second word.

EXPERIMENTAL RESULTS

For the sample profile shown in Figure 2, the computer uses a single term abbreviated strategy during the first pass. Here, all documents are examined one at a time in sequence to see if they contain the word KODAK. This means that the complete input file is searched against only one term rather than seven as in a one-pass search system. In practice, this reduction averages four to one. Coupled with rapid collating term match techniques, the two-pass system now runs 500% faster than the earlier system.

With the reduction in the number of search terms, it is now possible to consider batching first-pass strategies. In the earlier one-pass system, there was not enough memory available for grouping searches when all terms had to be in memory at the same time. Batching is expected to reduce run times by another 200–300%.

ACKNOWLEDGMENT

I would like to thank Mr. Richard Smith for his aid in programming the NERAC search system.

APPENDIX—SEARCH TIME EQUATIONS

Total run time, T_{run}, for a linear search is a function of total central processor time, T_{cpu}, and total tape time, T_{tape}. If computer operations are overlapped, T_{run} equals the larger of T_{cpu} and T_{tape}. If they are not, T_{run} equals the sum of T_{cpu} and T_{tape}.

Total tape time, T_{tape}, is a function of total input time, T_{input}, and total output time, T_{output}. Generally, since the number of retrieved documents is much less than the number of data bank documents, T_{output} is very small relative to T_{input}. Therefore, for simplicity, T_{output} is ignored.

T_{input} is the sum of the time spent starting the input tape drive, T_{start}, the time transmitting information from the drive to memory, T_{trans}, and the time stopping the drive, T_{stop}. Even the most elementary tape drive sends an end-of-transmission signal at each inter-record gap, and, thus, there is no need to wait for the drive to stop. Therefore

$$T_{tape} = T_{start} + T_{trans} \tag{1}$$

The total time spent starting the input drive is the product of the time per start, t_{start}, and the number of starts during a search. If the file being searched contains N_{doc} and if the input tape is blocked at an average of n_{block} documents per block, then total start time is

$$T_{start} = (N_{doc}/n_{block})t_{start} \tag{2}$$

The total tape transmission time is the product of the number of documents, N_{doc}, the average number of characters transmitted per document, n_{char}, and the time to transmit each character, t_{char}. Therefore, total transmission time is

$$T_{trans} = N_{doc}n_{char}t_{char} \tag{3}$$

Total tape time is then found by substituting (2) and (3) into (1) producing

$$T_{tape} = N_{doc}\left\{\frac{t_{start}}{n_{block}} + n_{char}t_{char}\right\} \tag{4}$$

The total central processing time, T_{cpu}, can be expressed as the product of the number of documents, N_{doc}, and the average time to process each document, t_{doc}.

$$T_{cpu} = N_{doc}t_{doc} \tag{5}$$

Here, the average time to process each document is the product of the average number of instructions to process that document, n_{inst}, times the time to execute an average instruction, t_{inst}.

$$t_{doc} = n_{inst}t_{inst} \tag{6}$$

In processing a linear file, document terms must be compared against question terms. The algorithm chosen to perform these comparisons must be a function of the average number of index terms per document, i_{doc}, and the number of retrieval search terms, i_{search}. If each document term is compared against each search term, then the average number of instructions to process each document is

$$n_{inst} = i_{doc}i_{search}n_{comp} + n_{house} \tag{7}$$

where n_{comp} specifies the number of instructions to make a term comparison while n_{house} represents the instructions to perform housekeeping functions. If both search and document terms have been previously sorted, this expression can be improved to

$$n_{inst} = (i_{doc} + i_{search})n_{comp} + n_{house} \tag{8}$$

Substituting (6) and (8) into (5), the total CPU time becomes

$$T_{cpu} = N_{doc}\{(i_{doc} + i_{search})n_{comp} + n_{house}\}t_{inst} \tag{9}$$

Examination of eq 4 and 9 shows that run time is a function of three independent sets of constraints. The *first* set is dictated by the data file and includes the number of data bank documents and the average number of index terms per document. The *second* set is determined by the computer and includes average instruction execution time, start time of the input tape drive, and time to transmit one input character from the drive to memory. The *third* set is specified by the information center and includes the average number of documents per input block, the average number of characters per input document, and the number of search terms per retrieval run. Here, the number of documents per block should be adjusted so as to produce a balanced run where tape time and CPU time are nearly equal. Balancing is discussed in Gildersleeve.[5]

LITERATURE CITED

(1) Wilde, D. U., "Iterative Strategy Design," *Am. Doc.*, 90–91 (1969).
(2) Wilde, D. U., "Using a Small/Low Cost Computer in an Information Center," Proceedings of the ASIS Mid-Year Regional Conference, Dayton, Ohio, May 1972.
(3) Swid, R. E., "Linear vs. Inverted File Searching on Serial Access Machines," 26th Annual Meeting of the American Documentation Institute, Chicago, Ill., Oct. 1963.
(4) Wilde, D. U., and A. C. Starke, "A Chemical Search System for a Small Computer," *J. Chem. Doc.* **14**, 41 (1974).
(5) Gildersleeve, T. R., "Design of Sequential File Systems," Wiley-Interscience, New York, N.Y., 1971.