Difficult Data Placement Problems

David A. Bell
School of Computer Science, Ulster Polytechnic, UK

The problems of data placement are investigated abstractly, and we use the theory of NP-completeness to prove some results which indicate the level of difficulty of some increasingly realistic data placement problems. As a consequence of these theorems it is established that the likelihood of finding a deterministic algorithm to solve any of these data placement problems with acceptable performance is very small, and other, possibly heuristic, methods are suggested. A particularly interesting data placement problem considered in this study is formulated as the layered data placement problem, which is shown to be NP-complete.

INTRODUCTION

It is a basic fact of physical database (DB) design that those potential storage schemes (defined below) for the organization of data in a DB system, DBS, which have small associated access times for queries based on multiple attributes of multiple relations, often have high associated redundancy. A question which naturally arises is:

Problem 1

Do there exist some query collections, and record sets used in their evaluation, for which it is possible to have storage schemes in which minimum access time is combined with zero or minimum redundancy or wasted space?[2]

The answering of this question will provide building blocks suitable for construction of complex structures to satisfy a variety of queries in particular applications.

Definition 1

A *storage scheme* is a combination of a *filing scheme* and a *placement scheme*. A *filing scheme* is a method for storing a set of records which all describe a single entity-type and have the same simple format—i.e. a file—in the store of a DBS. Placement schemes are explained below.

Such storage methods include a wide range of 'hashing algorithms' which have traditionally been applied to accessing files in primary storage, and also a collection of auxiliary data accessing structures among which tree-based techniques are most familiar. The latter also originated in primary storage applications but have more recently been given considerable attention as solutions to secondary (external) storage accessing problems, and this has led to the advent of a proliferation of 'external filing schemes', some of which combine hashing and tree techniques.[1]

In the present context, that of modelling the process of storing data on rotating storage devices, RSDs, such as disks or drums, we are not concerned with decisions relating to specific filing schemes. We are more concerned with the decision posed as Problem 2 below:

Problem 2

How can data subcollections, files, records and attribute-values associated with a query collection, Q, be juxtaposed (placed) on secondary storage devices of a DBS so that responses to the most important queries are quickest and the use of storage capacity is as favourable as possible?

Solutions to this problem are *placement schemes*. It is our intention to derive versions of Problem 2 which model the problem of placement of data in a DBS, so that insights into solution methodologies may be permitted. As various formulations of the problem are presented, their degrees of difficulty are studied and approaches to their solution are suggested which are the most likely to be rewarding.

IDEALIZED PLACEMENT PROBLEMS

In order to analyse general placement problems and develop a theory of placement it is useful to start from first principles in an idealized universe in which the storage of the DBS considered is composed of *linear* or *one-dimensional* stores.

In such stores the physical records are scanned and searched in a linear pattern. The next record to be accessed from a given initial position is located by traversing the intervening records sequentially one-by-one in a single linear direction. Thus we are considering very much simplified RSDs in which all the data are on a single track, organized in much the same manner as for a tape medium—the traditional one-dimensional store. Clearly detailed insights gained from such a study are of limited value, but this is a worthwhile analysis as an appreciation of the difficulty of placement of data in a DBS is imparted by it, and basic theoretical foundations are established.

We begin with a definition of a very desirable property for a given collection of queries, Q, with respect to a given collection of records, possibly representing different entities.

Definition 2

In a given DB, the *ordered consecutive retrieval property*, *OCRP*, is held by a collection of queries Q, with respect

CCC-0010-4620/84/0027-0315$03.00

to a collection of tuples, T, stored on a linear store, if for any $q \in Q$, all the tuples in T which are accessed during the evaluation of q can be stored without redundancy in consecutive storage cells of the store of the DBS, in accordance with the order in which they are referenced during the evaluation of q.

A weaker version of this property is the consecutive retrieval property proposed and analysed by Ghosh. [2, 3]

Definition 3

The *consecutive retrieval property*, *CRP*, is held by Q with respect to the tuple collection T if for any $q \in Q$ all tuples in T pertinent to the evaluation of q can be stored without redundancy in consecutive cells of the store of a DBS.

In addition to the direct access filing schemes outlined above, linking mechanisms are required to navigate between records in different files (or even within a single file). The different types of accessing mechanisms combine to make available access paths for queries on the DB. In terms of access path structure, Definition 2 means that at least one access path for each query $q \in Q$ must be composed of records which are grouped contiguously to form files as required. The files so defined all describe a single entity-type by definition, but may be interspersed by records containing attributes describing some other entity, in the order in which they are needed to evaluate the query. Definition 3 on the other hand merely means that if # instances of records are needed in the access path for any query $q \in Q$ then they are stored in # consecutive record-sized storage cells.

Example:

Consider a single relation EMP

EMP (Name, No, Dept, Manager, Salary)

composed of six tuples as in Table 1.

Table 1

Tuple-id	Name	No	Dept	Manager	Salary
t_1	Abbot	16	A	Abbot	20 000
t_2	Carey	25	A	Abbot	15 000
t_3	Jones	94	B	Smith	22 000
t_4	Pepper	106	A	Abbot	22 000
t_5	Smith	112	B	Smith	28 000
t_6	Thompson	131	A	Abbot	15 000

There are 3 queries against this relation with the following qualification predicates:

q_1: (Salary = 22 000)
q_2: (Salary = 22 000 $\vee$ Manager = Name)
q_3: (Dept = A)

whose targets are met from (some of) the attribute-values from tuples

t_3, t_4
t_1, t_3, t_4, t_5
t_1, t_2, t_4, t_6,

respectively. A possible permutation of the tuples to meet the requirements of both OCRP and CRP—assuming that the order of tuple presentation is not important for these queries—for q_1, q_2, q_3 against EMP is:

$$t_6\text{–}t_2\text{–}t_1\text{–}t_4\text{–}t_3\text{–}t_5 \qquad (1)$$

Here only hashing or indexing access mechanisms (on, for example, Salary, Dept and Manager: Name) are needed and are assumed to exist as required.

Consider now an additional query:

$$q_4 = \text{(Salary} > \text{Manager's Salary)}$$

(note: this query is abbreviated for clarity) which has target tuple t_4.

There are two possibilities here:

(a) q_4 may be a member of the 'original query set' used to determine initial access mechanism needs and therefore have tailored access paths so that only one record (t_4) need be accessed; so CRP and OCRP are both demonstrated for all four queries.

(b) q_4 may be an *ad hoc* query without tailored hash and link element support, and may therefore have an access path involving access to:

$$t_1, t_2\text{–}t_1, t_3\text{–}t_5, t_4\text{–}t_1, t_5, t_6\text{–}t_1 \qquad (2)$$

possibly using 'foreign keys' to link an employee with his manager.

Clearly the set of four queries has the CRP with respect to the six tuples with arrangement (1), but can it be said to have the OCRP if it is an *ad hoc* query?

To have the OCRP t_1 must simultaneously be adjacent to t_2, t_4, t_6, which is clearly impossible, and so (q_1, q_2, q_3, q_4) does not possess the OCRP with respect to the tuples ($t_1, t_2, t_3, t_4, t_5, t_6$) if q_4 is an *ad hoc* query. If some mechanism were provided to allow managers' salaries to be 'remembered' when they are hit, then the arrangement $t_5\text{–}t_3\text{–}t_4\text{–}t_1\text{–}t_2\text{–}t_6$ would meet the requirements of OCRP for all 4 queries.

THE INTRINSIC DIFFICULTY OF FUNDAMENTAL PLACEMENT PROBLEMS

The theory of NP-completeness[16] supplies techniques for proving that a given problem is 'just as hard as' a large family of other problems, which have become widely recognized as being exceptionally hard through their persistent resistance to efforts of experts at solving them for many years. The foundations of this theory were laid by Cook[4] when he drew attention to a class NP of decision problems—problems with 'yes' or 'no' answers—which can be solved in polynomial time by a non-deterministic Turing machine. Most apparently *intractable* decision problems—in the sense that no polynomial time algorithm can possibly solve them—which have been identified in practice can be classified as NP problems. There exists a set of NP problems—the 'hardest' members of the NP class—to which every other NP problem can be reduced polynomially, and the classical members of this subset were defined by Karp.[5] These are collectively referred to as the class of *NP-complete decision problems*.

These considerations have provided a theoretical basis

for reducing a large collection of individually complex problems to a single question: 'Are NP-complete problems intractable?' This is now one of the leading open questions in contemporary mathematics and informatics. Many investigators are willing to conjecture that NP-complete problems are intractable—but no rigorous theoretical basis has yet been established for this assumption. In this paper the technique of establishing that certain problems are NP-complete is used, with the intention of showing, not that they are intractable, but that they possess a well-defined level of complexity, and, at least, that dramatic progress will be required to produce an algorithm to solve them in polynomial time. Hence other methods of solution should probably be sought.

We now establish that some basic data placement problems are NP-complete.

A fundamental decision problem confronting the physical database designer is that of determining whether or not a given collection of queries has the CRP with respect to a given set of records. Ghosh[2,3] has established, using a set-theoretic approach, several basic theorems which provide assistance in answering this query if, for example, it is added to a set of queries possessing the CRP under certain conditions, or if it belongs to a set of queries whose pertinent record sets are nested. Many of Ghosh's results have been confirmed and more powerful results obtained by Eswaran[5] by adopting a graph-theoretic approach to analysing the CRP. Necessary conditions for the possession of the property were established using concepts from graph theory such as Hamiltonian paths, linear graphs and intersection graphs. A third approach by Nanako[7] was to use set-closure properties to establish necessary and sufficient conditions for CRP.

We now examine the CRP and similar properties at a level of abstraction slightly above this by looking at the intrinsic difficulty associated with determining if a set of queries has the property.

Problem 3

The *CRP(Q,T)-problem* is the name given to the problem of determining whether a given set of queries, Q, has the CRP with respect to a given set of tuples, T.

Definition 4

A binary matrix (i.e. one with only 0, 1 as entries) has the *consecutive ones property* if all its columns can be permuted so that each row has all its ones occurring consecutively.

Theorem 1

The CRP(Q,T)-problem is solvable in polynomial time.

Proof. Let $|T|$ be the cardinality of T. The CRP(Q,T)-problem is clearly equivalent to determining if a matrix has the consecutive ones property. Fulkerson and Gross[8] have shown that this problem is solvable in polynomial time. □

Note: the problem of checking the OCRP for a set of queries and a set of tuples can be similarly mapped onto a slightly more elaborate check of the consecutive ones property, and hence is not NP-complete either.

Definition 5

The *consecutive retrieval property with bounded range*, *CRBR(Q,T,K)*, is held by query collection Q with respect to tuple collection T and given $K \in Z^+$ if there exists $m \leq K$ consecutive record-sized storage cells in the store of the DBS, containing for any $q \in Q$, all tuples of T pertinent to the evaluation of q without redundancy.

This property may be restated as follows: given a finite set of tuples T, and a collection $\{T_i\}, 1 \leq i \leq n$, of subsets of T, containing tuples pertinent to queries $\{Q_i\}, 1 \leq i \leq n$, respectively, and $K \in Z^+$, then $\{Q_i\}$ has the CRBR (Q,T,K) w.r.t. $T \Leftrightarrow$ a sequence of tuple occurrences $s \in T^*$ with $|s| < K$ can be selected, such that $\forall\, i$ the elements of T_i occur consecutively (without redundancy) in s. Here T^* denotes the set of all finite subsets of T (with repetition allowed).

Problem 4

The *CRBR(Q,T,K)-problem* is the name given to the problem of determining whether a given set of queries Q has the CRBR(Q,T,K) with respect to a given set of tuples T and a given constant, $K \in Z^+$.

We now establish the, initially perhaps counter-intuitive, result that despite the fact that the CRP(Q,T)-problem is mathematically tractable, by Theorem 1, the CRBR(Q,T,K)-problem is intrinsically difficult.

Theorem 2

The CRBR(Q,T,K)-problem is NP-complete.

Proof. A non-deterministic Turing machine can guess the arrangement of the tuples. Let $|T|$ be the cardinality of T. Let Σ be an alphabet of $|T|$ symbols. There is a 1–1 correspondence $f: T \rightarrow \Sigma$ and, *a fortiori*, $f: T_i \rightarrow \Sigma_i$, for each subset T_i of T and some subset Σ_i of Σ, due to the fact that both, by definition, have a 1–1 correspondence with members of Z^+.

Similarly there is a 1–1 correspondence $g: \Sigma^* \rightarrow T^*$, where Σ^* is the set of strings w of Σ and T^* is the set of sequences of occurrences s of tuples in T. Kou[9] has shown, using a polynomial time transformation from the classic Hamiltonian path problem, which Karp[5] had previously shown to be NP-complete, that the problem of determining if there is a string w in Σ^* with $|w| < K$ such that $\forall i$ the elements of Σ_i can be stored consecutively without redundancy within w is NP-complete.

Hence the CRBR(Q,T,K)-problem is also NP-complete or else an NP-complete problem can be transformed into one solvable in polynomial time by a 1–1 mapping. □

MORE REALISTIC PLACEMENT MODELS AND PROBLEMS

Much of the work done on consecutive placement of records was, by implication at least, concerned with

single-relation queries. Just how applicable these concepts are to realistic database design problems was not considered beyond some queries on single-relation 'databases'.

Explicitly stated for multi-relational queries, the CRP(Q,T)-problem becomes the following:

Given a set, Q, of queries on a collection of relations $\{R_i\}$, $1 \leq i \leq n$, does there exist an organization of tuples in $\{R_i\}$, without duplication, such that $\forall$ query in Q all pertinent tuples can be stored in consecutive storage cells in the store of a DBS?

Clearly if $\bigcup_i(R_i)$ is treated as one large file consisting of multiple record-types (corresponding to different entities) and the different entity-types can be juxtaposed in any way desired, then the results of both Ghosh and Eswaran are directly applicable to a multi-relation database.

However, the restriction to linear storage devices is extremely untypical of RSDs, and the condition of consecutiveness is extremely restrictive and unlikely to be met in a real system design situation.

Hence the more tolerant CRBR(Q,T,K) property is more realistic, but we have an immediate possibility of confusion in trying to introduce a measure of distance. In Definition 5, K is a number of records, so the type (size) of records involved has a large impact upon the level of redundancy in terms of storage capacity—and ultimately upon retrieval performance.

For RSDs we assume a model of storage where each track of each cylinder is divided up into a fixed number of pages. We now use a well-known definition of distance between two tuples t_i and t_j, denoted $d(t_i, t_j)$.

Assume that there are x pages to the track and y cylinders on our RSDs, and define:

$$\text{latency cost} = rd_{ij}$$

where r is the time taken to rotate over 1 page, and d_{ij} is the number of pages in the direction of rotation separating the pages holding tuples i and j.

$$\text{seek cost} = sl_{pq}$$

where s is the time taken to move the R/W-head system of the RSD over 1 cylinder, and l_{pq} is the number of cylinders separating the cylinders p, q holding tuples i and j, respectively.

The *distance function* $d(t_i, t_j)$ is given by:

$$rd_{ij} + sl_{pq} = \begin{cases} r(j-i-1) + s(p-q) & \text{if } 1 \leq i < j \leq x \\ r(j-i-1+x) + s(p-q) & \text{if } 1 \leq j < i \leq x \\ r(x-1) + s(p-q) & \text{if } i = j \end{cases}$$

Note: this distance function ignores the contribution of transfer time to costs.

This concept of distance can be applied to CRBR(Q,T,K) in an obvious way in order to help in the generalization of the placement problem.

Problem 5

The *query set clustering problem*, *QC-problem* is stated as follows: Given a finite set, T, of tuples, and the distance function $d(t_1, t_2) \in Z^+$ as defined above, between every pair of tuples t_1 and t_2 in T, and positive integers K and $\{B_j\}$, $1 \leq j \leq K$. Can T be partitioned into K disjoint subsets $T_1, \ldots, T_n$ such that, $\forall i \leq K$ and $\forall t_1, t_2$ in T_i, $d(t_1, t_2) < B_i$?

Theorem 3

The QC-problem is NP-complete.

Proof. A result due to Brucker[10] implies that if all the B_i equal the same constant (B say) then the problem is NP-complete. In fact it can be shown even for $K = 3$ the problem remains NP-complete.

If the problem were not NP-complete for varying B_i in $\{B_j\}$, this would mean it was not NP-complete for $B = \max_j \{B_j\}$, which is a contradiction with Brucker's result above. □

Note. Variants of this problem with $\max\{d(t_1, t_2)\} < B_i$ or $\Sigma d(t_1, t_2) < B_i$ or ave $d(t_1, t_2) < B$ are also NP-complete. Also when the probabilities p_1 and p_2 of access to t_1 and t_2 respectively, are included, so that the condition becomes

$$p_1 p_2 d(t_1, t_2) < B_i$$

the problem remains NP-complete.

Another partitioning problem related to CRP is now given.

Problem 6

Given a finite set of tuples T, and a collection of subsets $\{T_i\}$, $1 \leq i \leq n$, of T corresponding to the sets of tuples pertinent to queries $\{Q_i\}$, $1 \leq i \leq n$, respectively. The *storage partitioning problem*, *SP-problem*, is stated as follows: Is there a set of disjoint partitions $P_1, \ldots, P_m$ of T such that at most one tuple corresponding to Q_i, $\forall i$, is in each P_j, and yet all tuples corresponding to Q_i (i.e. set T_i) are contained in $|T_i|$ consecutive partitions?

Theorem 4

The SP-problem is NP-complete.

The proof of this theorem follows as a special case of a general NP-complete partitioning problem analysed by Lipsky.[11]

It is clear from these results that any time spent trying to find an algorithm with polynomial execution time to solve these problems is very likely to be wasted. The corresponding real-world physical database design problems such as the problem of getting all tuples associated with a particular query (or application—see later) to be physically close to one another (e.g. on a single cylinder or page, where B_i is the (variable) cylinder or page size, respectively, in Problem 5, for instance), or to be placed on consecutive storage blocks (possibly the P_i of Problem 6 being simultaneously accessible storage units to allow parallelism) are therefore *proved to be extremely difficult problems*.

In more realistic models account must also be taken of the fact that queries are often logically grouped by the applications with which they are associated. These applications are likely to have (temporarily) fixed priorities or urgencies or probabilities, which imply a natural 'importance' ranking.

Definition 6

The set of applications $\{A_i\}$, $1 \leq i \leq n$, is said to possess the *application consecutive retrieval property*, *ACRP*, with respect to the tuples $\{T_l\}$, $1 \leq l \leq m$, if there exists an ordering of the $\{T_l\}$, without duplication, such that, for every A_j in $\{A_i\}$, all tuples pertinent to its collection of queries can be stored in consecutive storage locations.

A similar definition analogous to Definition 5 can also be given with 'queries' replaced by 'applications', and this can be extended to include the notion of distance as defined earlier in this section, rather than the notion of a bounded number of records (K) spanning the pertinent tuple set.

The major difference between these properties and the CRP of Ghosh *et al.*, is that in Definition 6 the individual queries are not constrained to have the CRP. The CRP only need hold for clusters of queries—those relating to the separate applications. All the theorems of Ghosh *et al.*, can be clearly applied within applications—although they must be extended to include more realistic treatment of DBS architectures. Ghosh has developed such a treatment in Ref. 12 and has thereby generalized his one-dimensional treatment considerably. However we believe that the concept of distance introduced here is more realistic.

Now we define an *ordering* on the set of applications $\{A_i\}$ as follows:

$$A_k \leq A_j \Leftrightarrow u_k \leq u_j, \quad \text{all } k \text{ and } J$$

where u_i is a measure of the urgency, probability, or other priority of application A_i, for each i.

The nature of the $\{A_i\}$ in practice suggests an amended version of the ACRP, where only a selection of the applications, the most important, need have the ACRP.

Definition 7

A set of applications $\{A_i\}$, $1 \leq i \leq m$, has the *truncated consecutive retrieval property of order n*, *TCRP(n)*, $n < m$, with respect to tuple collection $\{T_i\}$ if and only if $\{A_i\}$, $1 \leq i \leq n$, possess the ACRP w.r.t. $\{T_i\}$.

Clearly a similar truncated property can be defined for queries and the definitions of placement and retrieval properties of both this section and the preceding section may be rewritten correspondingly.

THE LAYERED PLACEMENT PROBLEM

The primary objective of this study of placement is to consider the problems defined next.

Problem 7

Given m tuples, $t_1, \ldots, t_m$ stored on n pages $p_1, \ldots, p_n$, let $d(ijz)$ be the retrieval unit cost to tuple t_i on page p_j when the previously accessed tuple was on page p_z. Tuples may be replicated between pages, but not within pages. We require to find an optimal placement (assignment) of tuples to pages as follows: we seek a minimal collection of pages $M \subset \{p_i\}$, $1 \leq i \leq n$ to store the m tuples so that the expected cost of retrieval of pairs of consecutively-accessed tuples is minimized and there is no duplication of tuples. The problem is called the *covering placement problem*, *CP-problem*.

The CP-problem is NP-complete

The proof of this result is similar to a proof of a result by Eswaran[13] for a network of computers, which uses a mapping from the set covering problem.[5] Thus this problem is not solvable in polynomial time or else many other difficult problems would be solvable in polynomial time by a deterministic Turing machine.

The CP-problem assumes an initial allocation of tuples to pages, possibly with replication between pages, and requires a non-redundant minimum cover of these pages. A very similar problem, the *data placement problem*, *DP-problem*, where the tuples are not initially assigned to pages but are to be optimally assigned during the solution, is also known to be NP-complete.

Problem 8

Given m tuples $t_1, \ldots, t_m$ stored on n_i occurrences of storage units su_i, where each i corresponds to a level of proximity, assume a fixed cost k of locating a tuple on an su_1 occurrence. Let d_i be the retrieval unit cost to a tuple on an su_i occurrence when the immediately preceding access was to a tuple on a different su_i occurrence, but not on a different su_j occurrence, any $j > i$. If $i = 1$, $d_1(xyz)$ represents the cost of retrieving tuple t_x on su_1 occurrence y when its immediate predecessor was on a different su_1 occurrence z.

Assume that there is a cost differential between each level of storage—i.e. for each i there is a factor f_i such that

$$d_i = f_i d_{i-1}, f_i > 1$$

Tuples may be replicated between su_1 occurrences but not within su_1 occurrences.

Assume that each su_i has a constant finite capacity c_i such that c_i is a number of tuples when $i = 1$ and a number of su_{i-1} occurrences for all other i.

We require to find an optimal placement of tuples to su_i units—i.e. to minimize the total cost of requested tuple-pair retrievals, while eliminating the duplication of tuples. This problem is called the *layered covering placement problem*, *LCP-problem*.

Theorem 6

The LCP-problem is NP-complete.

This result follows easily, *a fortiori*, from Theorem 5.

Again the analogy of the LCP-problem, called the *layered data placement problem*, *LDP-problem*, is NP-complete following from the complexity of the DP-problem.

These results show the difficulty of the problem of minimizing retrieval costs when there are multiple levels of proximity corresponding to 'same page', 'same cylinder', 'same disk', etc. (the su_i of Problem 8). Thus for example in addition to getting a minimum cover in

terms of pages, a minimum cover in terms of pages within a corresponding cylinder has to be obtained. In modern DBS the levels of the storage hierarchy which lead to the data staging problem may extend up through layers such as associative disk memories[14] to the level where the RSDs are on differing geographic sites. Clustering at all levels is worth while because of the cost differentials. The multi-level version of the DP-problem is the one given most attention when establishing techniques for placement in Ref. 15.

The results of this section therefore give formal warning that it is likely to be fruitless to look for solutions that solve the DP-problem or LDP-problem in polynomial time. The emphasis should be placed on heuristic approaches rather than deterministic approaches.

SUMMARY

In this paper the problems of data placement have been modelled at various increasing levels of realism, and it is shown that many of these problems are extremely difficult. The problems are all concerned with the determination of whether or not a given arrangement of tuples on physical storage is possible. They are not explicitly concerned with the problem of finding such arrangements. However it is clear that, because the 'existence' problems are NP-complete, the problem of finding the arrangements must be at least as difficult.

Hence it is proved to be inadvisable for database designers to seek algorithms to arrange data in the patterns described, and other systematic approaches, possible with less ambitious goals, are preferable. Some such approaches to placement problems are introduced and studied by Bell.[15]

Acknowledgements

This work was supported by funding from SERC. Useful comments were made by colleagues on the PRECI collaboration, especially Dr S. M. Deen of University of Aberdeen.

REFERENCES

1. D. A. Bell and S. M. Deen, Key space compression and hashing in PRECI. *The Computer Journal* **25** (4), 486–492 (1982).
2. S. P. Ghosh, File organisation: the consecutive retrieval property. *CACM* **15**, 802–808 (1972).
3. S. P. Ghosh, On the theory of consecutive storage of relevant records. *Information Science* **6**, 1–9 (1973).
4. S. A. Cook, The complexity of theorem proving procedures. *Proc 3rd ACM Conference on Theory of Computing,* May (1970).
5. R. M. Karp, *Reducibility Among Combinational Problems, complexity of Computations,* Plenum Press (1972).
6. K. P. Eswaran, Consecutive retrieval information systems. *PhD Thesis,* University of California, Berkeley (1973).
7. T. A. Nanako, Characterisation of intervals: the consecutive property. Comment. *Math Univ St Paul* **22**, 49–59 (1973).
8. D. R. Fulkerson and O. A. Gross, Incidence matrices and interval graphs. *Pacific J. Math* **15**, 835 (1965).
9. L. T. Kou, Polynomial complete consecutive information retrieval problems. *SIAM J. Computing* **6**, 67–75 (1977).
10. P. Brucker, On the complexity of clustering problems. *Optimierung and operations Research, Lecture Notes in Economics and Mathematical Systems,* Springer, Berlin.
11. W. Lipsky, Jr., One more polynomial complete consecutive retrieval problem. *Information Processing Letters* **6**, 91–93 (1977).
12. S. P. Ghosh, File organisation: consecutive storage of relevant records on drum-type storage. *Information and Control* **25** (2), 145–165 (1974).
13. K. P. Eswaran, Placement of records in a file and file allocation in a computer network. *Proc. Conf. Information Processing 74,* North Holland, 305–307 (1974).
14. E. A. Ozkarahan, S. A. Schuster and K. A. Smith, RAP—associative processor for databases management. *AFIPS Comp Proc,* 379–388 (1975).
15. D. A. Bell, Physical record clustering in Databases. *PRECI/up—W8203* (to be published—*Kybernetes*) (1984).
16. M. R. Garey and D. S. Johnson, *Computers and Intractability,* W. H. Freeman & Co (1979).

Received June 1983