Programming Techniques
S.L. Graham, R.L. Rivest Editors

Covering Edges by Cliques with Regard to Keyword Conflicts and Intersection Graphs

L.T. Kou, L.J. Stockmeyer, and C.K.Wong
IBM Thomas J. Watson Research Center

Kellerman has presented a method for determining keyword conflicts and described a heuristic algorithm which solves a certain combinatorial optimization problem in connection with this method. This optimization problem is here shown to be equivalent to the problem of covering the edges of a graph by complete subgraphs with the objective of minimizing the number of complete subgraphs. A relationship between this edge-clique-cover problem and the graph coloring problem is established which allows algorithms for either one of these problems to be constructed from algorithms for the other. As consequences of this relationship, the keyword conflict problem and the edge-clique-cover problem are shown to be NP-complete, and if $P \neq NP$ then they do not admit polynomial-time approximation algorithms which always produce solutions within a factor less than 2 from the optimum.

Key Words and Phrases: keyword conflicts, intersection graphs, node clique cover, edge clique cover, computational complexity, NP-complete problems, polynomial-time heuristics

CR Categories: 4.12, 5.25, 5.32

Author's address: IBM Thomas J. Watson Research Center, PO Box 218, Yorktown Heights, NY 10598.

1. Introduction

A recent paper of Kellerman [8] discusses the following combinatorial optimization problem. Let $P = [p_{ij}]$ be an $n \times n$ matrix of zeroes and ones such that $p_{ij} = p_{ji}$ and $p_{ii} = 1$ for all i and j. We say that the $n \times k$ zero-one matrix $Q = [q_{ij}]$ has the *intersection property* with respect to P provided that $(q_{s1}q_{t1}, q_{s2}q_{t2}, \ldots, q_{sk}q_{tk}) = (0, 0, \ldots, 0)$ if and only if $p_{st} = 0$ for all s and t. In other words, row s and row t of Q have no 1's in the same position if and only if $p_{st} = 0$. Given P, the problem is to find such a matrix Q with k as small as possible. Since, in Kellerman's case, P is the conflict matrix for a set of n keywords, we refer to this problem as the *keyword conflict problem*. In [8], a heuristic for constructing Q with the intersection property is given, but the heuristic does not necessarily achieve the minimum k. See [8] for further discussion and motivation of the keyword conflict problem.

This problem is exactly the intersection graph problem described in [3, 4]: Given a graph G, find a set S and a family F of nonempty subsets of S such that G is the intersection graph of the family F; the objective is to minimize the cardinality of S. (G is the intersection graph of the family $F_1, \ldots, F_n \subseteq S$ if the nodes of G can be numbered $1, 2, 3, \ldots, n$ in such a way that node i and node j ($i \neq j$) are connected by an edge in G if and only if $F_i \cap F_j \neq \phi$.) Some worst-case bounds on the cardinality of S are given in [3, 4], but no algorithm for constructing S and F is discussed.

The purpose of the present paper is to first show that the keyword conflict problem (and hence the intersection graph problem) is equivalent to the following *edge-clique-cover* (ECC) problem: Given a graph G, find a set of complete subgraphs (cliques) which includes every edge of G; the objective is to minimize the number of cliques. Having demonstrated this equivalence, we then in Section 3 investigate the computational complexity of the ECC problem. The problem of covering the edges of a graph by the minimum number of cliques is shown to be NP-complete.[1]

[1] The reader who is unfamiliar with the concept of NP-completeness is referred to [1,7] for an introduction. The key fact is that either all or none of the NP-complete problems can be solved by polynomial-time algorithms. (An algorithm runs in polynomial-time if it always terminates within a number of steps which is bounded above by some polynomial in the size of the input.) Although it is not known which of these two possibilities holds, the answer is "none" if and only if $P \neq NP$ where P (NP) is the class of sets which can be recognized by deterministic (nondeterministic) Turing machines within polynomial-time. Since the class of NP-complete problems is known to include such notoriously difficult problems as the traveling salesman problem and 0-1 integer programming, it is believed that $P \neq NP$.

However, proving a problem to be NP-complete does not preclude the possibility of finding polynomial-time approximation algorithms for the problem. Such algorithms, while not always producing optimal solutions, are guaranteed to find solutions "close" to optimal (say, within a constant factor of the optimum). By using a result of Garey and Johnson [2], we are able to bound the performance of polynomial-time approximation algorithms for the ECC problem, under the assumption that $P \neq NP$. If $P \neq NP$ and if c and d are constants with $c < 2$, then there is no polynomial-time algorithm for the ECC problem which always produces $cE_0 + d$ or fewer cliques, where E_0 is the number of cliques in an optimal solution.

In Section 4 the heuristic proposed in [8] is slightly improved. It is easily seen to be still polynomial-time. Finally, we construct a family of graphs to demonstrate that E/E_0 is arbitrarily large, where E is the number of cliques produced by the improved heuristic and E_0 is the optimal number. Of course the exhibition of this possibly anomalous behavior is not intended as a comment on the performance of the algorithm in practice. In light of the results of Section 3, however, the paper would not be complete without a demonstration that the ratio E/E_0 for this existing heuristic algorithm is bounded by no constant (in particular, by no constant less than 2).

2. Equivalence of the Keyword Conflict Problem and the Edge-Clique-Cover Problem

The edge-clique-cover (ECC) problem can be stated as follows: Given a graph G, find a set of cliques which includes every edge of G (i.e. find a set of complete subgraphs such that every pair of connected nodes belongs to some complete subgraph in the set). Note that the usual clique cover problem is to find a set of cliques which covers the nodes; this problem will be referred to as the node-clique-cover (NCC) problem. If G is a graph, we let $ECC(G)$ ($NCC(G)$) denote the minimum number of cliques which are sufficient to cover the edges (nodes) of G. If A is an algorithm which solves the ECC (NCC) problem, then $A(G)$ denotes the number of cliques produced by A when applied to the graph G.

We first show that the keyword conflict problem and the ECC problem are equivalent in the strong sense that, if we view the matrix P as the incidence matrix of a graph G where each keyword corresponds to a node, then a solution Q to the keyword conflict problem immediately yields a solution to the ECC problem, and vice versa. For the sake of simplicity in what follows, we assume that zero-one matrices P contain no isolated keywords; that is, for each i there is a j with $i \neq j$ and $p_{ij} = 1$. Similarly, we assume that graphs contain no isolated nodes; that is, every node is connected to some other node. Isolated keywords (isolated nodes) can easily be detected and eliminated from further consideration. If P is an $n \times n$ matrix as described in the Introduction, then we let $G(P)$ denote that graph whose incidence matrix is P (i.e. G has nodes $\{1, 2, \ldots, n\}$ and, for all i and j with $i \neq j$, there is an edge connecting node i to node j iff $p_{ij} = 1$). If Q is an $n \times k$ zero-one matrix, then we let $C(Q)$ denote the family of sets $\{C_1, \ldots, C_k\}$ where $C_j = \{i \mid q_{ij} = 1\}$ for $i \leq j \leq k$.

PROPOSITION 1. *Let P be an $n \times n$ zero-one matrix and let Q be an $n \times k$ zero-one matrix. Q has the intersection property with respect to P if and only if $C(Q)$ is an edge-clique-cover of $G(P)$.*

PROOF. *If.* Say that $C(Q)$ is an edge-clique-cover of $G(P)$. Let $(q_{s1}, q_{s2}, \ldots, q_{sk})$ and $(q_{t1}, q_{t2}, \ldots, q_{tk})$ be row s and t of Q. Then $(q_{s1}q_{t1}, \ldots, q_{sk}q_{tk}) = (0, 0, \ldots, 0)$ iff node s and node t do not belong to the same clique in $C(Q)$. Moreover, it follows from definitions that node s and node t do not belong to the same clique iff s and t are not connected in $G(P)$ iff $p_{st} = 0$. Therefore Q has the intersection property.

Only if. Say that Q has the intersection property and let $C(Q) = \{C_1, \ldots, C_k\}$. For an arbitrary j with $1 \leq j \leq k$, say that $C_j = \{s_1, s_2, \ldots, s_l\}$. By the intersection property of Q, the (s_α, s_β) = entry in P is 1 for all $1 \leq \alpha, \beta \leq l$. Thus C_j is a clique in $G(P)$. Suppose that node s and node t are connected in $G(P)$; then $p_{st} = 1$. Consequently $q_{sj}\, q_{tj} = 1$ for some j. So $q_{sj} = q_{tj} = 1$, and hence node s and node t belong to the same clique C_j. Therefore $\{C_1, \ldots, C_k\}$ is an edge-clique-cover of G. □

3. Complexity of the ECC Problem

It is well known that the NCC problem is merely a restatement of the graph coloring problem (by considering the complementary graph).[2] Garey and Johnson [2] have bounded the performance of polynomial-time graph coloring algorithms, under the assumption that $P \neq NP$. Translating their result to the NCC problem, it states that if $P \neq NP$ and if there are constants c and d and a polynomial-time algorithm A' for the NCC problem such that $A'(G) \leq c \cdot NCC(G) + d$ for all graphs G, then $c \geq 2$. A similar result for the ECC problem will follow from Proposition 2, which establishes a relationship between heuristic algorithms for the NCC and ECC problems.

PROPOSITION 2. *Let c be a nonnegative constant. There is a polynomial time algorithm A for the ECC problem such that $A(G) \leq c \cdot ECC(G) + d$ for all G if and only if there is a polynomial-time algorithm A' for the NCC problem such that $A'(G) \leq c \cdot NCC(G) + d'$ for all G.*

[2] If G is a graph, its complement $\bar{G}$ is defined to have the same set of nodes as G, and two nodes are connected in $\bar{G}$ iff they are not connected in G. Note that if S is a subset of the nodes, then S is a clique in G iff the nodes in S can be colored the same color in $\bar{G}$ (i.e., S is an independent set in $\bar{G}$).

PROOF. *If.* Given an algorithm A' for the NCC problem we describe an algorithm A for the ECC problem. Let G be a given graph input for the ECC problem. Say that G has edges $\{e_1, e_2, \ldots, e_m\}$. Form a graph G' with nodes $\{n_1, n_2, \ldots, n_m\}$ and edges as follows. For $1 \leq i < j \leq m$, let I_{ij} denote the set of nodes in G upon which e_i and e_j are incident (so that I_{ij} contains either three or four nodes, depending on whether or not e_i and e_j share an endpoint). Join n_i and n_j by an edge in G' iff I_{ij} contains the nodes of a complete subgraph of G. It is easy to verify that $ECC(G) = NCC(G')$. The algorithm A constructs G' and applies A' to G'.

Only if. Let A be the given algorithm for the ECC problem. Let G be an input for the NCC problem and say that G has e edges. Consider t additional nodes $U = \{u_1, u_2, \ldots, u_t\}$, where $t = \lceil ce + d + 1 \rceil$. Join each u_i to all nodes in G. Let the new graph be G'. We first claim that

$$ECC(G') \leq t \cdot NCC(G) + e.$$

To prove this statement, let $\{C_1, C_2, \ldots, C_{NCC(G)}\}$ be a node-clique-cover for G. By construction, $\{u_i\} \cup C_j$ is a clique in G', and the set of all such cliques $\{u_i\} \cup C_j$ for $1 \leq i \leq t$ and $1 \leq j \leq NCC(G)$ covers all edges from U to G. In order to cover all edges in G, we need at most e more cliques. Thus the total is at most $t \cdot NCC(G) + e$. The polynomial-time algorithm A applied to G' produces a set S of cliques which cover the edges of G'. By hypothesis, the cardinality of S is

$$A(G') \leq c \cdot ECC(G') + d \leq c(t \cdot NCC(G) + e) + d.$$

For $1 \leq i \leq t$, let $S_i \subseteq S$ be the set of cliques which contain the node u_i, and let k_i be the cardinality of S_i. Note that $S_i \cap S_j = \phi$ for $i \neq j$ because u_i and u_j are not connected. For any i, the cliques in S_i with the node u_i deleted can clearly be used as a node-clique-cover for G. Thus within polynomial time we can extract from S a node-clique-cover for G with cardinality equal to

$$\min_{1 \leq i \leq t} k_i \leq \left\lfloor \frac{\sum_{i=1}^{t} k_i}{t} \right\rfloor \leq \left\lfloor \frac{A(G')}{t} \right\rfloor \leq c \cdot NCC(G) + 1. \quad \square$$

The following corollary is immediate from Proposition 2 (only if) and the result of Garey and Johnson mentioned above.

COROLLARY 1. *If $P \neq NP$ and if there are nonnegative constants c and d and a polynomial-time algorithm A for the ECC problem such that $A(G) \leq c \cdot ECC(G) + d$ for all graphs G, then $c \geq 2$.*

We now note that the problem of finding a minimum edge-clique-cover is NP-complete. This means that $P = NP$ if and only if there is a polynomial-time algorithm for the ECC problem such that $A(G) = ECC(G)$ for all G. In order to conform to the framework used by Karp [7] in proving NP-completeness, we restate the ECC and NCC problems as set recognition problems. Let

$$SET\text{-}NCC = \{(G, k) | NCC(G) \leq k\},$$
$$SET\text{-}ECC = \{(G, k) | ECC(G) \leq k\}.$$

PROPOSITION 3. *SET-ECC is NP-complete.*

PROOF. As is customary with such proofs, we omit the trivial verification that *SET-ECC* belongs to *NP*. Since *SET-NCC* is known to be NP-complete [7], we need only show that *SET-NCC* is transformable to *SET-ECC* by a function computable in polynomial-time. Let (G, k) be a given graph-integer pair. Construct the graph G' as in the proof of Proposition 2 (only if) where $c = 1$ and $d = 0$, so that $t = e + 1$. Let $k' = k(e + 1) + e$. It follows as in the proof of Proposition 2 that

$$(G, k) \in SET\text{-}NCC \text{ iff } (G', k') \in SET\text{-}ECC,$$

which completes the proof. $\square$

4. Heuristic for the ECC Problem

In view of the results in the previous section, it seems unlikely that there is a polynomial-time algorithm A for the ECC problem with a ratio $A(G)/ECC(G)$ less than 2, and there is even cause to suspect that there is no such algorithm with bounded ratio. The literature contains a number of heuristic algorithms for the NCC problem (in the guise of graph coloring algorithms), for example [9, 10]; each such NCC algorithm yields an ECC algorithm by Proposition 2 (if). However, Johnson [5] has analyzed the behavior of several graph coloring algorithms and has shown the ratio $A(G)/NCC(G)$ to be unbounded for all those which he considered. In light of Proposition 2 (only if), this adds weight to the suspicion that there is no polynomial-time ECC algorithm with $A(G)/ECC(G)$ bounded.

Kellerman [8] has proposed a polynomial-time heuristic for the keyword conflict problem. It is an intuitively appealing heuristic and indeed works very well for small problems. However, we shall construct a family of cases such that even an improved version of the heuristic will have an arbitrarily large ratio.

When translated to the ECC problem, the heuristic in [8] can be stated as follows. Let G be a given graph input. We assume that the nodes of G have been labeled $1, 2, \ldots, n$. The algorithm forms cliques C_1, $C_2, \ldots$ by examining the nodes one by one. When a node i is examined, only edges connecting i to nodes with smaller label are considered. In the following description of the algorithm, i is the node currently being examined, and k is the number of cliques which have been created so far. The set W contains those nodes $j < i$ such that an edge connecting $\{i, j\}$ has yet to be covered.

1. Initialize $k \leftarrow 0$ and $i \leftarrow 0$.
2. Set $i \leftarrow i + 1$ and terminate the algorithm if $i > n$.
   Set $W \leftarrow \{j | j < i$ and $\{i, j\}$ are connected in $G\}$.
3. If $W = \phi$, then set $k \leftarrow k + 1$, create a new clique $C_k = \{i\}$, and go to 2.
4. Try to insert i into existing cliques:
   4.1. Set $m \leftarrow 1$ and $V \leftarrow \phi$.
   4.2. While $m \leq k$ and $V \neq W$ do
      4.2.1 If $C_m \subseteq W$, then set $C_m \leftarrow C_m \cup \{i\}$
         and $V \leftarrow V \cup C_m$.
      4.2.2 $m \leftarrow m + 1$.
5. Update W to account for those edges which were covered in step 4: $W \leftarrow W - V$.
6. If $W = \phi$, then go to 2. Otherwise new cliques must be added. Find an m, $1 \leq m \leq k$, such that the cardinality of $C_m \cap W$ is maximal; break ties by choosing the smallest such m. Set $k \leftarrow k + 1$, $C_k \leftarrow (C_m \cap W) \cup \{i\}$, $W \leftarrow W - C_m$, and go to 6.

(Note that step 3 cannot produce useless singleton cliques because we have assumed that G contains no isolated nodes. The cliques initialized in step 3 are subsequently grown in step 6.)

In the example in Figure 1, the resulting cliques are $C_1 = \{1, 3, 5\}$, $C_2 = \{2, 3, 4\}$, $C_3 = \{4, 5, 6\}$, $C_4 = \{3, 6\}$. Note that when considering node 5, we see it is connected to 1, 3, 4. Since $C_1 = \{1, 3\}$ at that moment, it includes 5 to form a new clique $C_1 = \{1, 3, 5\}$, and a new clique $C_3 = \{4, 5\}$ is created. As a result, for node 6, we need two cliques, $\{4, 5, 6\}$ and $\{3, 6\}$. Clearly, an optimal clique cover is $C_1 = \{1, 3, 5\}$, $C_2 = \{2, 3, 4\}$, $C_3 = \{3, 4, 5, 6\}$.

Our proposed improvement of the above heuristic requires one additional pass after the original algorithm terminates in step 2.

7. Suppose $C_1, C_2, \ldots, C_k$ are the cliques produced by the previous heuristic. Examine the cliques one by one to see if the edges covered by a clique are a subset of the union of the edges covered by the remaining cliques. If a clique is subsumed by the union of the remaining cliques, then eliminate it.

This version is still polynomial-time, and it certainly will never produce more cliques than the original heuristic.

For example, in Figure 2 the original heuristic will produce cliques $C_1 = \{1, 2, 3\}$, $C_2 = \{1, 2, 4\}$, $C_3 = \{1, 3, 5\}$, $C_4 = \{2, 3, 6\}$. But the edges covered by C_1 are a subset of those covered by C_2, C_3, C_4; C_1 is thus eliminated by the modified version. In fact $\{C_2, C_3, C_4\}$ is an optimal edge-clique-cover.

It should be pointed out that, although a straightforward implementation of the clique elimination step 7 runs in polynomial time, this step might consume more time than the original heuristic. The increase could be, in the worst case, a factor proportional to the number of cliques found by the original heuristic.

We now construct a family of graphs $\{G_{m,t}\}$ which illustrate that even the improved heuristic can produce a number of cliques which is an arbitrarily large multiple of the optimal number. This family is very similar to one which Johnson [6] has constructed to illustrate the worst-case performance of the Welsh and Powell [9] coloring algorithm. Let $\{a_1, a_2, \ldots, a_m\}$ be the

Fig. 1.

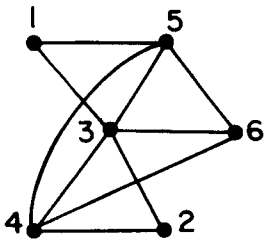
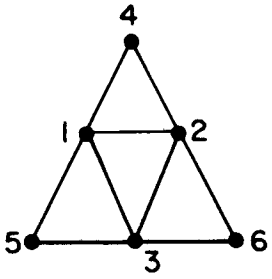


Fig. 2.



nodes of a complete graph A and let $\{b_1, b_2, \ldots, b_m\}$ be the nodes of a complete graph B. Join $\{a_i, b_i\}$ for $i = 1, 2, \ldots, m$. Let $\{u_1, u_2, \ldots, u_t\}$ be a set of disjoint nodes. Join each u_i to all nodes in A and B. The graph thus constructed is the required $G_{m,t}$.

We label the nodes of $G_{m,t}$ in the order $u_1, u_2, \ldots, u_t, a_1, b_1, a_2, b_2, \ldots, a_m, b_m$. Then steps 1-6 of the heuristic produce the following cliques:

$$C_i = \{a_1, b_1, u_i\}, \quad i = 1, 2, \ldots, t,$$
$$C_{t+1} = \{a_1, a_2, a_3, \ldots, a_m, u_1\},$$
$$C_{t+i} = \{a_2, b_2, u_i\}, \quad i = 2, 3, \ldots, t,$$
$$C_{2t+1} = \{b_1, b_2, b_3, \ldots, b_m, u_1\},$$
$$C_{(t-1)(j-1)+i+2} = \{a_j, b_j, u_i\}, \quad 2 \leq i \leq t, 3 \leq j \leq m.$$

For example, when b_2 is under consideration, $C_1, C_2, \ldots, C_{2t}$ are already in existence. b_2 is connected to $u_1, u_2, \ldots, u_t, a_2$, and b_1, so that b_2 can be inserted into the cliques $C_{t+2}, \ldots, C_{2t}$. This leaves only the edges $\{b_1, b_2\}$ and $\{u_1, b_2\}$ uncovered; so a new clique C_{2t+1} is created. Thereafter, when examining a_j for $j \geq 3$, step 4 of the algorithm inserts a_j into C_{t+1}. No other insertions are possible because all other existing cliques contain some node in B which is not connected to a_j. Therefore step 6 creates $t - 1$ new cliques to cover the edges from a_j to $u_2, u_3, \ldots, u_t$. Then b_j is inserted into C_{2t+1} and into the $t - 1$ cliques which were just created for a_j. Note that the number of cliques is $m(t - 1) + 3$.

Consider now the effect of the clique elimination step 7 when applied to this edge-clique-cover. C_1 can be eliminated, but each of the other cliques C_j for $j \geq 2$ covers an edge which is covered by no other clique. For example, C_j is the only clique which covers the edge connecting u_j to a_1, for $j = 2, 3, \ldots, t$. Letting

H denote the heuristic algorithm consisting of steps 1-7 above, we therefore have

$$H(G_{m,t}) = m(t - 1) + 2.$$

On the other hand, if we form cliques in the following manner:

$$C'_i = \{a_1, \ldots, a_m, u_i\}, \quad i = 1, 2, \ldots, t,$$
$$C'_{t+i} = \{b_1, \ldots, b_m, u_i\}, \quad i = 1, 2, \ldots, t,$$
$$C'_{2t+j} = \{a_j, b_j\}, \quad j = 1, 2, \ldots, m,$$

then $\{C'_1, \ldots, C'_{2t+m}\}$ is an edge-clique-cover for $G_{m,t}$. Therefore,

$$ECC(G_{m,t}) \leq 2t + m.$$

Choosing $m = t$, we have

$$H(G_{m,t})/ECC(G_{m,t}) \geq [m(t - 1) + 2]/(2t + m) = (t^2 - t + 2)/3t \to \infty \text{ as } t \to \infty.$$

Received May 1976; revised October 1976.

References

1. Aho, A.V., Hopcroft, J.E., and Ullman, J.D. *The Design and Analysis of Computer Algorithms*. Addison-Wesley, Reading, Mass., 1974, chap. 10.
2. Garey, M.R., and Johnson, D.S. The complexity of near-optimal graph coloring. *J.ACM 23*, 1 (Jan. 1976), 43-49.
3. Harary, F. *Graph Theory*. Addison-Wesley, Reading, Mass., 1969.
4. Harary, F. On the Intersection Number of a Graph. In *Proof Techniques in Graph Theory*, F. Harary, Ed., Academic Press, New York, 1969, pp. 71-72.
5. Johnson, D.S. Worst case behavior of graph coloring algorithms. Proc. Fifth Southeastern Conf. on Combinatorics, Graph Theory, and Computing, Utilitas Mathematica Pub., Winnipeg, Canada, 1974, pp. 513-528.
6. Johnson, D.S. Approximation algorithms for combinatorial problems. *J. Comptr. Syst. Sci. 9*(1974), 256-278.
7. Karp, R.M. On the computational complexity of combinatorial problems. *Networks 5* (1975), 45-68.
8. Kellerman, E. Determination of keyword conflict. *IBM Tech. Disclosure Bull. 16*, 2 (July 1973), 544-546.
9. Welsh, D.J.A., and Powell, M.B. An upper bound to the chromatic number of a graph and its application to time tabling problems. *Comptr. J. 10* (1967), 85-86.
10. Wood, D.C. A technique for coloring a graph applicable to large scale time-tabling problems. *Comptr. J. 12* (1969), 317-319.

Management Applications — H. Morgan Editor

B-trees Re-examined

Gerald Held, Tandem Computers, Inc.
and Michael Stonebraker,
University of California, Berkeley

The B-tree and its variants have, with increasing frequency, been proposed as a basic storage structure for multiuser database applications. Here, three potential problems which must be dealt with in such a structure that do not arise in more traditional static directory structures are indicated. One problem is a possible performance penalty.

Key Words and Phrases: B-tree, directory, static directory, dynamic directory, index sequential access method

CR Categories: 3.70, 3.73, 3.74, 4.33, 4.34

Introduction

The B-tree [1] has been receiving considerable attention as a storage structure for certain files on paged secondary storage devices. Such files include those consisting of a collection of records each of which has an identifying portion called a KEY. Access to the file is desired both randomly (by requesting the record corresponding to a given key) and sequentially (in collating sequence by key value).

In this paper we briefly explain a B-tree, several of its variants and an alternate static directory structure. Then we indicate some potential problems which B-tree implementations must overcome in a multiuser database environment that do not arise in static directory structures. These problems include a possible performance penalty.

Authors' addresses: Gerald Held, Tandem Computers, Inc., 20605 Valley Green Drive, Capertino, CA 95014; Michael Stonebraker, Department of Electrical Engineering and Computer Science, University of California, Berkeley, CA 95720.