THE UNIVERSITY OF MICHIGAN

COMPUTING RESEARCH LABORATORY

AN OPTIMAL DIGRAPH CLUSTERING APPROACH TO DATABASE RECORD CLUSTERING

Wan P. Chiang
Toby J. Teorey

CRL-TR-31-84

JULY 1984

Room 1079, East Engineering Building
Ann Arbor, Michigan 48109
USA
Tel: (313) 763-8000

ensgn

UMR1191

An Optimal Digraph Clustering Approach to Database Record Clustering

by

Wan P. Chiang*

Toby J. Teorey**

June 1984

* Bell Laboratories, 1600 Osgood Street, North Andover, MA 1845

** Dept. of Electrical Engineering and Computer Science, The University of Michigan, Ann Arbor, MI 48109

ABSTRACT

The database record clustering problem is represented in terms of a directed graph, and solved by determining how to cut the graph into a set of nonvoid and disjoint subgraphs such that each subgraph satifies a set of database system constraints while an objective function of physical block accesses is optimized. It is shown that if the logical database structure can be represented as an out-tree or out-necklace, this problem can be solved in pseudopolynomial time although the general problem is NP-complete. Efficient algorithms are given to compute the optimal solution for the general cases where the logical database structure can be represented as an acyclic or general digraph. An example database design problem is solved to illustrate the approach.

Categories and Subject Descriptors: G.2.2 [Discrete Mathematics]: Graph Theory, H.2.2 [Database Management]: Physical Design - access methods.

General Terms: Performance, Theory.

Additional Keywords and Phrases: Physical database design, record clustering.

TABLE OF CONTENTS

1. Introduction

An important cost factor of a physical database design depends on the secondary storage search time that is required by the typical operational workload. The major component of that search time is the expected number of physical block accesses (PBAs). Thus, one important consideration in the initial physical database design, or a later reorganization of the database, is to map the logical database structure into a secondary storage in such a way so as to minimize PBAs [ChTe82]. One way to minimize the expected PBAs is to make use of the database usage pattern to group occurrences of those record types that are logically related and are frequently accessed together as clusters and store them into the same block so that intra-block PBAs can be maximized while inter-block PBAs can be minimized [Schk77, TeFr82, Marc83].

Many DBMSs (e.g., IDMS, IMS, DB2) allow users to specify such a request using a statement like CLUSTERED CHILD VIA PARENT CHILD RELATIONSHIP NEAR PARENT. For example, in Figure 1a, X is the parent record type of child record type Y. The logically related occurrences of X and Y are physically linked together like Figure 1b. Assume occurrences of X and Y are frequently accessed together. A user can specify CLUSTERED Y VIA X-Y NEAR X which will result in the placement of the logically related occurrences of X and Y by the DBMS as shown in Figure 1c. This placement has the advantage that any time x_i is read into memory, y_{i1}, y_{i2} and y_{i3} are also in memory. This reduces PBAs.

There are several restrictions to the way a database designer can specify the grouping of occurrences statement. They are:

1) each specification must be via a parent – child relationship;

2) each record type can only be clustered with one of its

parents;

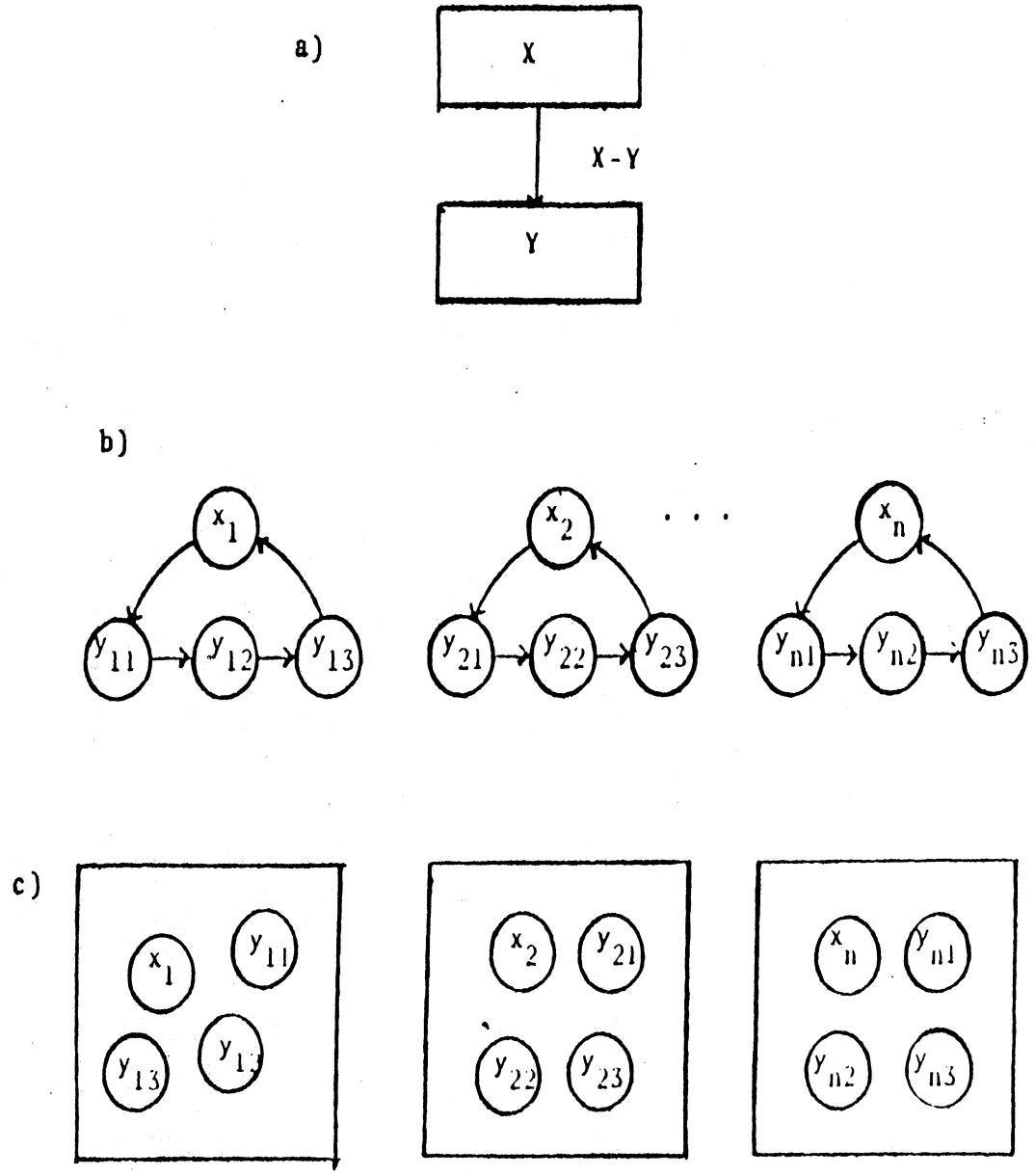
3) a record type may not be CLUSTERED VIA a PARENT CHILD RELATIONSHIP of which it is both child and parent type; and

4) the specification cannot form a cycle.

Let the logical database structure be represented by a digraph G=(V,E) where each vertex represents a record type and each arc a parent child relationship. The first restriction implies that on each arc there is one decision of value 0 or 1 to be made. A 1 value decision means the occurrences of the two record types which are related by the parent child relationship are going to be stored in the same cluster while a 0 value means otherwise. The second, third and fourth restrictions imply that each cluster(subgraph) resulting from the specification is an out-tree (see Appendix A for the basic definitions of graph theory terminology).

Schkolnick [Schk77] was the first to present a cost model and algorithm for the special case where the given logical database structure can be represented as an out-tree. But his algorithm runs in exponential time with respect to the outdegree of each vertex in the out-tree. Chiang & Teorey [ChTe82] extended Schkolnick's approach to deal with the case where the given logical database structure can be represented as a general digraph and obtained an exponential time algorithm also.

This paper presents a simpler cost model and a more efficient algorithm to Schkolnick's and Chiang & Teorey's work. We also consider the case where the given logical database structure can be represented as an out-necklace or acyclic digraph in addition to out-tree and general digraph. The approach is to represent the logical database structure as a digraph and formalize the database record clustering problem as four Optimal Digraph Clustering (ODC)



A rectangle represents a record type.

A square represents a block.

A circle represents a record occurrence.

Figure 1 Placements of record occurrences using CLUSTERED CHILD VIA PARENT CHILD RELATIONSHIP NEAR PARENT.

subproblems [Chia84]: $ODC_{t°t°}$, $ODC_{t°n°}$, $ODC_{t°a}$ and $ODC_{t°g}$. The first subscript of ODC indicates each cluster (subgraph) resulted from a specification must be an out-tree while the second subscript means the given digraph is either an out-tree($t°$), out-necklace($n°$), acyclic digraph(a) or general digraph(g) respectively.

Given a (connected) graph G=(V,E) with vertex set V and arc set E, a clustering K of G is defined as a cutting of G into a set of nonvoid and disjoint (connected) subgraphs by removing some arcs in G. Each of the (connected) subgraphs is called a cluster induced by the clustering K (see Figure 2b). Thus, a clustering K is an assignment function which assigns either 0 or 1 to each arc of E, i.e., K : E --> {0,1}. An assignment on arc (i,j) with value 0, denoted by $K_{ij}=0$, means the arc (i,j) is removed (or cut) while a 1 assignment on arc (i,j), denoted by $K_{ij}=1$, means the arc (i,j) is not removed (or cut) (see Figure 2a).

The database record clustering problem can be formalized as a maximization problem as follows:

Given:

1) a digraph G=(V,E) which can be an out-tree, out-necklace, acyclic digraph or general digraph;

2) two constraints: one requires each cluster to be an out-tree; one limits the length $L(C[v_i])$ of each cluster $C[v_i]$ rooted at v_i to the block length B:

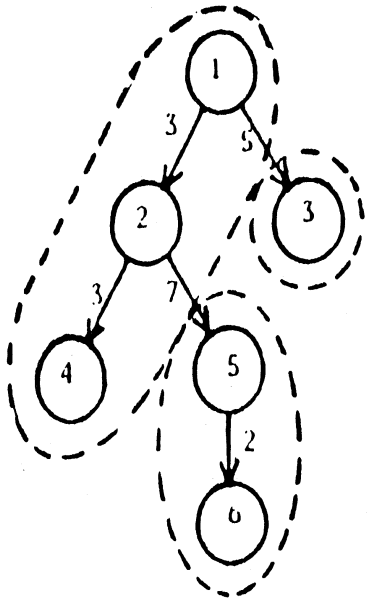
$$L(C[v_i]) = s_i + \overline{\sum_{\forall j(\overline{K_{ij}}=1)}} \lceil n_j/n_i \cdot L(C[v_j]) \rceil \leq B$$

where n_i and n_j are the expected number of occurrences of vertices v_i and v_j, s_i is the length of vertices v_i and $L(C[v_j])$ is the length of the complete subgraph in $C[v_i]$ rooted at v_j;

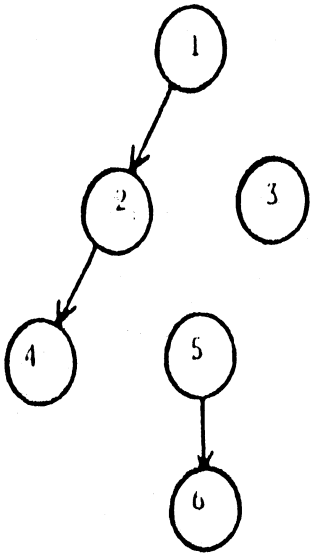
3) an objective function which is defined as

$$F(K) = \overline{\sum_{K_{ij}=1}} f_{ij}$$

a)



b)



B = 3

$\forall v_i$ (s=1) & (n_i=1)

Figure 2a) A clustering with $K_{13}=K_{25}=0$, $K_{12}=K_{24}=K_{56}=1$

Figure 2b) Three clusters induced by the clustering

where f_{ij} is the frequency of accesses between vertices v_i and v_j.

Find: A feasible clustering K such that $F(K)$ is maximized.

All ODC subproblems discussed in this paper are NP-complete problems [Chia84]. But this does not mean efficient and practical algorithms cannot be found to solve them. Because all of them are number problems, two pseudopolynomial algorithms are presented in this paper for $ODC_{t°t°}$ and $ODC_{t°n°}$. Two exponential algorithms for $ODC_{t°a}$ $ODC_{t°g}$ are also given which are useful when the given acyclic or general digraph is sparse. An algorithm for a problem is pseudopolynomial if it solves any instance *I* of the problem in time bounded by a polynomial function of the encoding length of *I* and the largest integer appearing in *I*. A polynomial algorithm runs in time bounded by a polynomial function of the the encoding length of *I* alone. One advantage of a pseudopolynomial algorithm for a problem is that it is practical (efficient) in many real life applications. This is because the largest integer in the problem is often bounded by a polynomial function of the encoding length of the problem and thus the pseudopolynomial algorithm becomes a polynomial algorithm. A second advantage is that even if the largest integer appearing in the problem is not bounded by a polynomial function of the encoding length of the problem, it can often be turned into fully polynomial time approximate schemes [GaJo79].

2. A Sequential Clustering Approach to $ODC_{t°t°}$

Given an out-tree G, a sequential clustering approach sequences the clustering process into a finite sequentially ordered stages. At each stage, a larger set of subgraphs of G, which is obtained from the immediate previous stage by adding a vertex or an arc, is considered. Because the set of sugraphs of G considered at one stage contains only one more

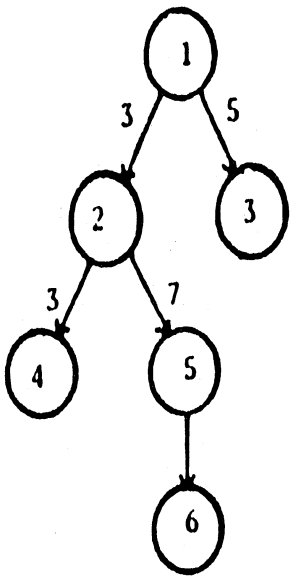
vertex or arc than its immediate previous stage, every feasible clustering of one stage can be obtained from the immediate previous stage by a simple operation (i.e., an operation that takes constant time to execute). The main thrust is to use the integer cluster length constraint and objective function to define an elimination rule such that at most a constant number of feasible clusterings (B) remains for each subgraph at each stage. Thus, the problem can be solved in pseudopolynomial time.

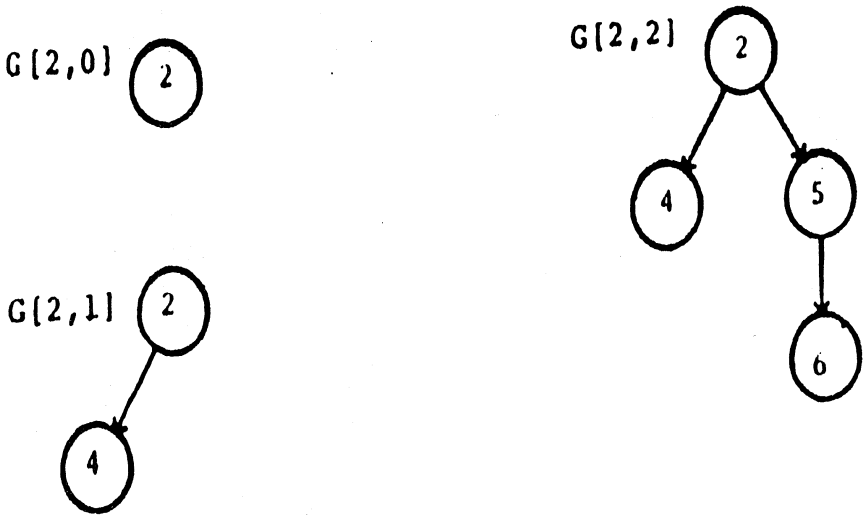
To describe how the clustering process is sequenced and to identify the simple operations needed in the sequential clustering approach, it is necessary to first define some terms and notations.

Terms and Notations:

Let the given out-tree $G=(V,E)$ have n vertices and m arcs and let v_{root} be the root of G. For each $v_r \in V$, $G[v_r]$ is any subgraph of G rooted at v_r. $od(v_r)$ is the outdegree of v_r where $od(v_r)=0$ if v_r is a leaf vertex. For each $v_r \in V$ and i, $0 \le i \le od(v_r)$, $G[v_r,i]$ is the subgraph of G induced by v_r, by its first i children (from left to right), and by all their descendants (see Figure 3). Note that $G[v_{root},od(v_{root})]=G$ and $G[v_r,0]$ is the subgraph of G consisting of only the vertex v_r. A subgraph $G[v_r,i]$ of G rooted at v_r is a <u>complete subgraph</u> if $i = od(v_r)$ and is a <u>partial subgraph</u> if $i < od(v_r)$. Observe that $G[v_r,i+1]$ is a larger subgraph of G than $G[v_r,i]$. It not only contains the partial subgraph $G[v_r,i]$ rooted at v_r but also the complete subgraph rooted at v_j, which is the i+1<u>th</u> child of v_r (denoted by $G[v_j,od(v_j)]$), and the arc (v_r,v_j) connecting the two subgraphs $G[v_r,i]$ and $G[v_j,od(v_j)]$.

The clustering process is sequenced in stages by visiting the vertices of G in <u>postorder traversal sequence</u> [Aho74]. At each vertex v_r, $G[v_r,0]$ is first considered and





B = 3

Vv_i (s=1) & (n_i=1)

Figure 3 An out-tree and its G[v_r,i]'s.

then $G[v_r,1]$, $G[v_r,2]$, ..., $G[v_r,od(v_r)]$. Since there are n vertices and m arcs, there are m+n stages. Figure 4 shows all eleven stages of the clustering process of the out-tree that has six vertices and five arcs given in Figure 3. Note that each vertex v_r is involved in $od(v_r)+1$ stages.

In Figure 4, because the vertices are visited in postorder traversal sequence, at each stage of the clustering process, the graph considered is a set of subgraphs of G (not necessarily a single connected subgraph). For example, at stage 6, there are two subgraphs: (V = {2,4}, E = {(2,4)}) and (V = {5,6}, E = {(5,6)}). Except for at most one, each subgraph in the set is a complete subgraph of G. For example, at stage 6, subgraph (V = {5,6}, E = {(5,6)}) is a complete subgraph rooted at 5 while (V = {2,4}, E = {(2,4)}) is a partial subgraph rooted at 2. At stage 7, there is only one subgraph rooted at 2 and it is a complete subgraph. There are two patterns of changes in the set of subgraphs considered from stage to stage:

1) If all subgraphs in stage k are complete subgraphs, then in stage k+1 the set of subgraphs considered will contain a new vertex as a separate subgraph. This stage k+1 is called a new vertex stage (e.g., stages 1,2,3,5,8,and 9 in Figure 4).

2) If one of the subgraphs in stage k is a partial subgraph, then in stage k+1 the set of subgraphs considered will contain a new arc which connects the root of this partial subgraph with the root of a complete subgraph existing in the set of subgraphs considered in stage k. This stage k+1 is called a new arc stage (e.g., stages 4,6,7,10,and 11 in Figure 4).

Since the set of subgraphs of G considered at each new stage only contains one more new vertex or arc than the immediate previous stage, there is no need to form all possible clusterings from scratch for each new stage. This means

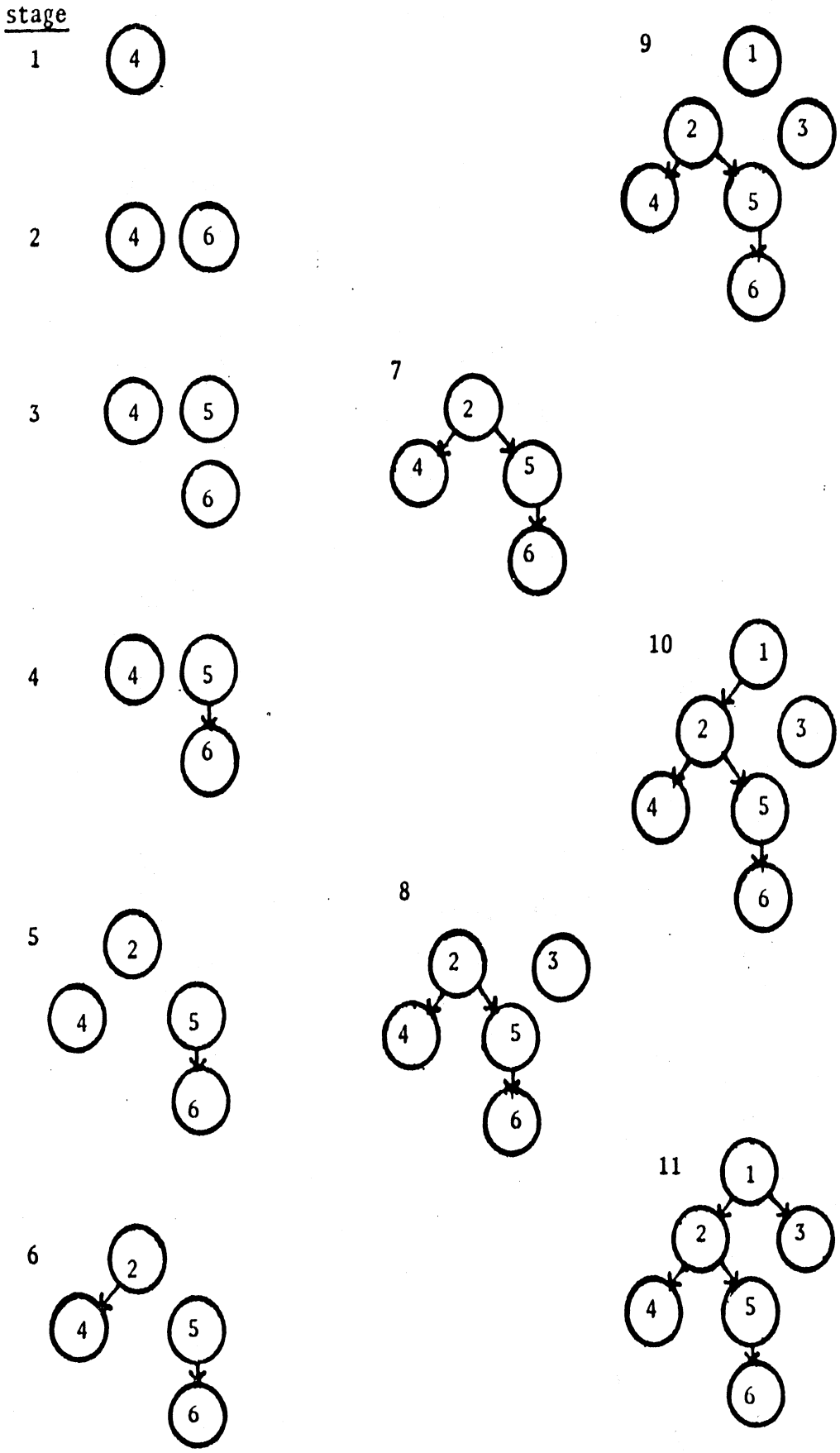


Figure 4 The eleven stages of the clustering process of the out-tree in Figure 3.

that if the set of all possible clusterings PK_k of stage k is known, the set of all possible clusterings PK_{k+1} of the next stage k+1 can be obtained by some simple operations on PK_k.

The set of all possible clusterings PK_k formed at stage k can be represented by s subsets of clusterings if there are s disjoint subgraphs considered at that stage. Each subset of clusterings represents the set of all possible clusterings for each subgraph. Let each clustering be represented by the set of clusters it induces. For example, in Figure 4, {(V={4},E=Ø)},{(V={5,6},E=Ø), (V={5,6},E={(5,6)})} is all possible clusterings of the two subgraphs in stage 4. To obtain all possible clusterings of stage 5, a new vertex stage, an operation which "inserts" vertex 2 as a separate cluster into the set of possible clusterings is needed. Thus, the resulting clusterings of stage 5 is {(V={2},E=Ø)},{(V={4},E=Ø)},{(V={5,6}, E={(5,6)}),(V={5,6},E=Ø)}. Consider stage 6, a new arc stage, since vertices 2 and 4 are connected by arc (2,4), there is a possibility that (V={2},E=Ø) and (V={4},E=Ø) could be "connected" together to form one cluster after clustering (i.e, assignment K_{24}=1) or not be connected together but simply "put" next to each other (i.e., assignment K_{24}=0). Thus, the new set of possible clusterings in stage 6 is
{(V={2,4},E={(2,4)}),(V={2,4},E=Ø)},
{(V={5,6},E={(5,6)}),(V={5,6},E=Ø)}.

It is now obvious that the necessary operations needed in the sequential clustering approach are an insertion operation in the new vertex stage and connect/put operations in the new arc stage. To formalize these operations, let $\{G[v_1,od(v_1)]$, $G[v_2,od(v_2)]$, ..., $G[v_j,od(v_j)]$, ..., $G[v_r,i]\}$ be the set of mutually disjoint subgraphs rooted at $v_1,v_2,\ldots,v_j,\ldots,v_r$ that are being considered in stage k, with $G[v_j,od(v_j)]$ as the i+1<u>th</u> complete child subgraph of

v_r. Let PK_k be the set of all possible clusterings formed at stage k. PK_k includes $PK(G[v_1,od(v_1)]$, $PK(G[v_2,od(v_2)])$, ..., $PK(G[v_j,od(v_j)]$, ..., and $PK(G[v_r,i])$ because there are r disjoint subgraphs considered at stage k. The operations needed in the two types of stages are:

New vertex stage:

At stage k+1, the set of subgraphs considered is $\{G[v_1,od(v_1)]$, $G[v_2,od(v_2)]$, ..., $G[v_j,od(v_j)]$, ..., $G[v_r,i]$, $G[v_p,0]\}$ where $G[v_p,0]$ represents the new vertex v_p that is introduced. There is only one way to obtain a clustering of a single vertex digraph $G[v_p,0]$, which is a cluster containing only the vertex v_p. This clustering is denoted by $PK[v_p,0]$. Thus, the set of all possible clusterings for stage k+1 can be obtained by taking the union of $PK[v_p,0]$ and PK_k, i.e.,

$$PK_{k+1} = PK_k \cup PK[v_p,0] \quad \text{where}$$

$$PK[v_p,0] = \{K[v_p,0] \text{ is formed by INSERT}\}$$

INSERT operation:

$$K[v_p,0] = \{(V=\{v_p\},E=\emptyset)\}.$$

New arc stage:

At stage k+1, the set of subgraphs considered is $\{G[v_1,od(v_1)]$, $G[v_2,od(v_2)]$, ..., $G[v_{j-1},od(v_{j-1})]$, $G[v_{j+1},od(v_{j+1})]$, ..., $G[v_{r-1},od(v_{r-1})]$, $G[v_r,i+1]\}$ where $G[v_r,i+1]$ is a subgraph formed by connecting $G[v_j,od(v_j)]$ and $G[v_r,i]$ that exist in stage k with an arc (v_r,v_j). Let the set of possible clusterings of $G[v_j,od(v_j)]$ and $G[v_r,i]$ be denoted by $PK[v_j,od(v_j)]$ and $PK[v_r,i]$ respectively. Thus,

$$PK_{k+1} = PK_k - PK[v_r,i] - PK[v_j,od(v_j)] \cup PK[v_r,i+1].$$

Let $K[v_r,i]$ and $K[v_j,od(v_j)]$ be any two clusterings of $PK[v_r,i]$ and $PK[v_j,od(v_j)]$ respectively. Let $K[v_r,i] = \{C_1,C_2,\ldots,C_q\}$, and $K[v_j,od(v_j)] = \{C_1',C_2',\ldots,C_p'\}$, and

assume $C_1=(V_1,E_1)$ and $C_1'=(V_1',E_1')$ are clusters that contain the root of $G[v_r,i]$ and $G[v_j,od(v_j)]$ respectively.

The set of all possible clusterings for $PK[v_r,i+1]$ can be obtained from all possible combinations of $K[v_r,i]$ in $PK[v_r,i]$ and $K[v_j,od(v_j)]$ in $PK[v_j,od(v_j)]$ for assignment $K_{ij}=$ 0 (denoted by $PK^0[v_r,i+1]$) and 1 (denoted by $PK^1[v_r,i+1]$) on arc (v_r,v_j), i.e.,

$PK[v_r,i+1] = PK^0[v_r,i+1] \cup PK^1[v_r,i+1]$;

$PK^0[v_r,i+1] = \{K^0[v_r,i+1]$ is formed using PUT for every pair of $K[v_r,i]$ and $K[v_j,od(v_j)]\}$;

$PK^1[v_r,i+1] = \{K^1[v_r,i+1]$ is formed using CONNECT for every pair of $K[v_r,i]$ and $K[v_j,od(v_j)]\}$, where

PUT operation:

$K^0[v_r,i+1] = K[v_r,i] \cup K[v_j,od(v_j)]$;

CONNECT operation:

$K^1[v_r,i+1] = \{C_2,C_3,\ldots,C_p,C_2',C_3',\ldots,C_q'\} \cup \{C_1''\}$

where $C_1'' = (V_1 \cup V_1', E_1 \cup E_1' \cup \{(v_r,v_j)\})$.

To obtain all feasible clusterings (i.e., clusterings that satisfy all constraints) instead of just all possible clusterings, the out-tree cluster and cluster length constraints need to be enforced at each stage. This can be easily done because:

1) Any cluster formed in the sequential clustering approach is a subgraph of an out-tree and is also an out-tree itself. Thus, the out-tree cluster constraint is automatically enforced in the process.

2) For each pair of clusterings from $PK[v_r,i]$ and $PK[v_j,od(i+1)]$, only the CONNECT operation forms a new cluster C_1'' which has a new higher cluster length value. To make sure every cluster formed is less than B at each new arc stage, the CONNECT operation can be

made conditional as follows:

IF $L(C_1) + [n_1'/n_1 \cdot L(C_1')] \leq B$ THEN

$K^1[v_r,i+1] = \{C_2,C_3,\ldots,C_p,C_2',C_3',\ldots,C_q'\} \cup \{C_1''\}$.

Thus, the set of all feasible clusterings formed at stage k+1 (denoted PK^f_{k+1}) can be easily obtained by applying either INSERT or PUT/conditional CONNECT operations on PK^f_k.

For computational purpose, let each feasible clustering K of a subgraph be represented by a triplet (F(K),L(K),CF(K)) that contains the following three components:

1) A value component F(K) which stores the sum of the cluster values of all the clusters induced by the clustering.

2) A length component L(K) which stores the cluster length of the cluster that contains the root of the subgraph.

3) A configuration component CF(K) which stores the clusters induced by this clustering of the subgraph and each cluster is represented by the vertices in it enclosed by "< >".

At each stage of the clustering process, if there are s subgraphs considered at a stage, there are s sets of feasible triplets formed. Each of these s sets of feasible triplets represents all feasible clusterings of a subgraph considered at that stage.

Notation:

Let PK^f_k and PK^f_{k+1} now represent all feasible triplets produced at stages k and k+1, and let $PK^f[v_r,i+1]$, $PK^f[v_r,i]$, and $PK^f[v_j,od(v_j)]$ now represent all feasible triplets of the subgraph $G[v_r,i+1]$, $G[v_r,i]$ and $G[v_j,od(v_j)]$ respectively. Let $K[v_r,i+1]$, $K[v_r,i]$ and $K[v_j,od(v_j)]$ be a feasible triplet in $PK^f[v_r,i+1]$,

$PK^f[v_r,i]$ and $PK^f[v_j,od(v_j)]$ respectively. Let $CF(K[v_r,i]) = \{C_1,C_2,\ldots,C_q\}$ and $CF(K[v_j,od(v_j)]) = \{C_1', C_2',\ldots C_p'\}$.

The three operations can now be redefined in terms of triplet formation:

INSERT operation:

form $K[v_p,0]$ with

$F(K[v_p,0] = 0$,

$L(K[v_p,0] = s_p$, and

$CF(K[v_p,0] = \{<v_p>\}$.

PUT operation:

form $K^o[v_r,i+1]$ with

$F(K[v_r,i+1]) = F(K[v_r,i]) + F(K[v_j,od(v_j)])$,

$L(K[v_r,i+1]) = L(K[v_r,i])$, and

$CF(K[v_r,i+1]) = CF(K[v_r,i]) \cup CF(K[v_j,od(v_j)])$.

Conditional CONNECT operation:

IF $L(C_1) + L(C_1') \leq B$ THEN

form $K^1[v_r,i+1]$ with

$F(K[v_r,i+1]) = F(K[v_r,i]) + F(K[v_j,od(v_j)]) + f_{rj}$,

$L(K[v_r,i+1]) = L(C_1) + [n_1'/n_1 \cdot L(C_1')]$

$CF(K[v_r,i+1]) = \{C_2,C_3,\ldots,C_p,C_2', C_3',\ldots,C_q'\}$

$\cup \{C_1 \cup C_1'\}$.

For example, let $\{(0,1,<4>)\},\{(0,1,<5><6>),(2,2,<5,6>)\}$ be the set of all feasible triplets formed at stage 4 in Figure 4. The feasible triplet sets formed at stage 5 using the INSERT operation are $\{(0,1,<2>)\},\{(0,1,<4>)\},\{(0,1,<5><6>),(2,2,<5,6>)\}$. The

feasible triplet sets formed at stages 6 and 7 using the PUT and conditional CONNECT are: (Assume B=3 and $\forall v_p$ (s_p=1 and n_p=1).)

stage 6:

{(0,1,<2><4>),(3,2,<2,4>)},

{(0,1,<5><6>),(2,2,<5,6>)};

stage 7:

{(0,1,<2><4><5><6>),(2,1,<2><4><5,6>),

(3,2,<2,4><5><6>),(5,2,<2,4><5,6>),(7,2,<2,5><4><6>),

(9,3,<2,5,6><4>),(10,3,<2,4,5><6>)}.

In the above example, every new vertex stage forms one additional new triplet using INSERT while every new arc stage can double the number of triplets formed in the immediate previous stage using PUT/conditional CONNECT. Thus, unless reduction of the number of feasible triplets can be done at the end of each new arc stage, the total computation time can still be exponential. Fortunately, some triplets formed at each new arc stage can be eliminated because they can never be extended to the optimal clustering of the out-tree.

The idea is to define a dominance relation on the set of newly formed feasible triplets at each new arc stage. Let K_1 and K_2 be any two feasible triplets formed at the same new arc stage. K_1 dominates K_2 if $F(K_1) \geq F(K_2)$ and $L(K_1) \leq L(K_2)$.

Proposition 2.1: The dominated triplet K_2 cannot be extended to the optimal clustering of the out-tree.

Proof: See Appendix B. ***Q.E.D.***

Proposition 2.1 implies that at the end of each new arc stage, all dominated triplets can be eliminated. The elimination can be done in two steps by decomposing $L(K_1) \leq$

$L(K_2)$ into

$L(K_1) = L(K_2)$ and $L(K_1) < L(K_2)$:

step 1) $L(K_1) = L(K_2)$ and $F(K_1) \geq F(K_2)$:

For triplets that have the same length, retain the one with the highest value and eliminate the rest. Thus, only one triplet remains for each integer length value.

step 2) $L(K_1) < L(K_2)$ and $F(K_1) \geq F(K_2)$:

For every remaining triplet K_2, if it has higher length than another triplet K_1 but have lower or equal value, then eliminate K_2.

Assume that every vertex has vertex length 1. Since the length of a triplet which represents a feasible clustering of a subgraph is an integer less than or equals to B, the maximum number of triplets remaining for a subgraph after each stage is B. If the vertex has length higher than 1, and suppose Z is the maximum number of vertices that can be put in a cluster with cluster length less than or equals to B, then the maximum number of triplets remaining for a subgraph after each stage is Z (<B).

A complete algorithm called $ODC_{t^0t^0}$-1 can now be specified:

ALGORITHM: $ODC_{t^0t^0}$-1

INPUT: an out-tree G;

OUTPUT: an optimal out-tree clustering of G and its value;

PROCESS:

I) FOR each v_r visited in postorder traversal sequence DO:

1) FOR j=0 to $od(v_r)$ DO:

1.1) IF j=0 THEN form a new PK^f_{k+1} from PK^f_k

using INSERT; ELSE

1.2) form a new PK^f_{k+1} from PK^f_k using PUT and conditional CONNECT;

eliminate all dominated triplets;

END;

II) find the optimal value clustering in PK^f_{m+n} and RETURN the optimal value F(K) and the optimal clustering configuration CF(K)

END.

Proposition 2.2: The running time of algorithm $ODC_{t^\circ t^\circ}$-1 is $O(n \cdot Z^2)$ (or $O(n \cdot B^2)$ if every vertex has vertex length 1).

Proof: Every new vertex stage takes constant time while every new arc stage takes $2 \cdot Z^2$ time because of the PUT and conditional CONNECT. Thus, step I) takes $O(n \cdot Z^2)$ time and Step II) takes O(Z) time. The time complexity of $ODC_{t^\circ t^\circ}$ is therefore $O(n \cdot Z^2)$. ***Q.E.D.***

Example: Applying $ODC_{t^\circ t^\circ}$-1 to the example given in Figure 2-2, the following 24 triplets are formed during the clustering process:

formed triplets F(K),L(K),CF(K)	formed by INSERT (stage #)	formed by P/C (stage #)	elimi. by P/C (stage #)	elim. by Dominate rel. (stage #)
(0,1,<4>)	1		6	
(0,1,<6>)	2		4	
(0,1,<5>)	3		4	
(0,1,<5><6>)		4	7	
(2,2,<5,6>)		4	7	
(0,1,<2>)	5		6	
(0,1,<2><4>)		6	7	
(3,2,<2,4>)		6	7	
(0,1,<2><4><5><6>)		7		7
(2,1,<2><4><5,6>)		7	10	
(7,2,<2,5><4><6>)		7	10	

(9,3,<2,5,6><4>)		7		7
(3,2,<2,4><5><6>)		7		7
(5,2,<2,4><5,6>)		7		7
(10,3,<2,4,5><6>)		7	10	
(0,1,<3>)	8		11	
(0,1,<1>)	9		10	
(2,1,<1><2><4><5,6>)		10		10
(7,1,<1><2,5><4><6>)		10		10
(10,1,<1><2,4,5><6>)		10	11	
(5,2,<1,2><4><5,6>)		10		10
(10,3,<1,2,5><4><6>)		10		10
(10,1,<1><2,4,5><6><3>)		11		
(15,2,<1,3><2,4,5><6>)		11		

The optimal value is 15 and the optimal clustering configuration is <1,3><2,4,5><6>.

Further improvements on $ODC_{t°t°}$-1 can be made, but since the time complexity will not change, we will not discuss them here.

3. Maximal Clusterings Enumeration Approach to $ODC_{t°n°}$, $ODC_{t°a}$ and $ODC_{t°g}$

Let K_i and K_j be any two clusterings of a digraph G. $K_i \leq K_j$ if every cluster induced by K_i is a subgraph of some cluster induced by K_j. For example, in Figure 5, each K_3, K_4, K_5, K_8, K_9, K_{10} and $K_{12} \leq K_1$. Note that if $K_i \leq K_j$, then K_i can be obtained from K_j by removing some arcs in K_j. The $\leq$ relation is reflexive, transitive and anti-symmetric. It induces a partial ordering on the set of all possible clusterings PK of G. Let [PK,$\leq$] denote the partly ordered set (poset) that consists of a set PK and a partial ordering relation $\leq$ on PK.

For any two elements K_i and K_j in PK, an element K_k in PK is called the greatest lower bound, or meet of K_i and K_j, denoted by $K_i \cdot K_j$ if

$K_k \leq K_i$, $K_k \leq K_j$, and

$\forall K_l$ in PK such that $K_l \leq K_i$ and $K_l \leq K_j$ imply $K_l \leq K_k$.

Dually, an element $K_m \in$ PK is called the least upper bound, or join of K_i and K_j, denoted by $K_i + K_j$, if

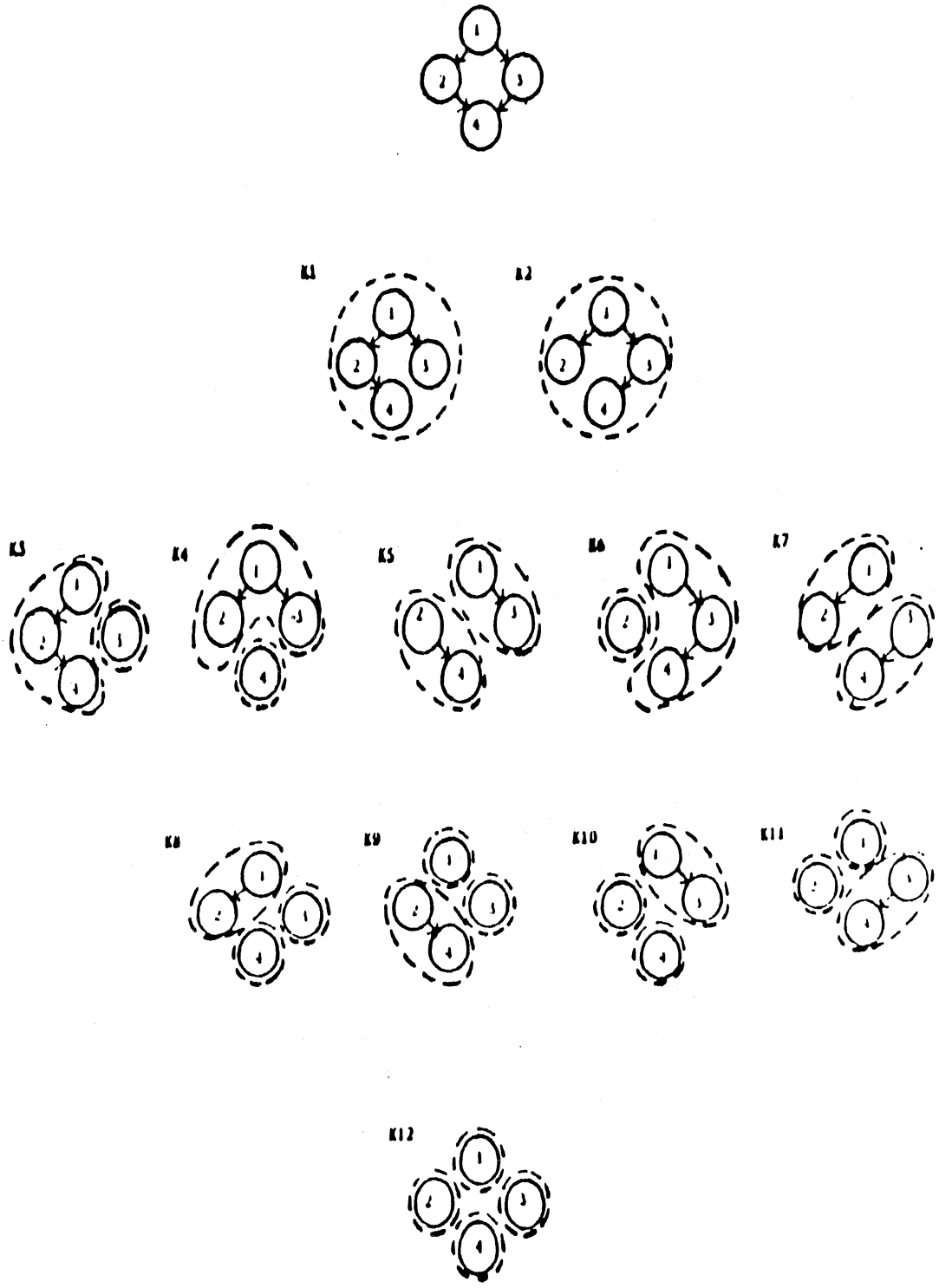


Figure 5 All possible out-tree clusterings of an acyclic digraph.

$K_i \leq K_m$ and $K_j \leq K_m$ and

$\forall$ K_l in PK such that $K_i \leq K_l$ and $K_j \leq K_l$ imply $K_m \leq K_l$.

A lattice is a partly ordered set which has a least upper bound and a greatest lower bound for every pair of elememts. Let x and y be any of the five digraph classes. Define the set of all x clusterings of a specific y as the subset of all possible clusterings of y which satisfies the constraint that each of these clusterings induces cluster of class x.

Proposition 3.1: The set of all out-tree clusterings of an out-tree together with the $\leq$ relation form a lattice.

Proof: This is immediate from the fact that every subgraph of an out-tree is an out-tree. Thus, every possible clustering is an out-tree clustering and the set forms a lattice. ***Q.E.D.***

But the set of out-tree clusterings of a general digraph, acyclic digraph or out-necklace is not a lattice because it does not always have a least upper bound for each pair of elements. For example, in Figure 5, $K_1 \cdot K_2 = K_4$ and $K_3 + K_{10} = K_1$, but $K_3 + K_7$ is undefined.

A poset [PK,$\leq$] is a join-semilattice (meet-semilattice) if for any two elements K_i and K_j in the poset PK, $K_i + K_j$ ($K_i \cdot K_j$) exists.

Proposition 3.2: The set of all out-tree clusterings of a general digraph/acyclic digraph/out-necklace together with the $\leq$ relation form a meet-semilattice.

Proof: For each cluster C_i induced by K_i and C_j by K_j and $C = C_i \cap C_j$, form three clusters (where $\cap$ represents graph intersection):

1) $C_i - C$.

2) $C_j - C$.

3) C.

The set of all clusters so formed for each pair of clusters from K_i and K_j, denoted by K_k, is also a clustering of PK because each cluster is an out-tree. Obviously, $K_k \leq K_i$ and $K_k \leq K_j$.

Now, assume that there is a K_l in PK such that $K_l \leq K_i$, $K_l \leq K_j$ and $K_k \leq K_l$. Let $C = C_i \cap C_j \neq \emptyset$ be an induced cluster in K_k. By $K_k \leq K_l$, C must also be a subgraph of some C' induced by K_l. Assume $C'-C \neq \emptyset$, then C' cannot be a subgraph of both K_i and K_j. Thus, C' is not a subgraph of both K_i and K_j. This contradicts the assumption that $K_l \leq K_i$ and $K_l \leq K_j$. Therefore, either C=C' or $C \subset C'$ is true and $K_l \leq K_k$. This means that $\forall K_l$ in PK such that $K_l \leq K_i$ and $K_l \leq K_j$ imply $K_l \leq K_k$. Thus, the set of all out-tree clusterings of a general digraph/acyclic digraph/out-necklace together with $\leq$ forms a meet-semilattice. ***Q.E.D.***

An element $K_i \in PK$ is <u>maximal</u> when there is no other element K_j in PK where $K_i \leq K_j$ (a <u>minimal</u> element is similarly defined). For a lattice, there is always one minimal element and one maximal element. In a meet-semilattice, we note that there is always one minimal element (e.g., K_{12} in Figure 5) and many maximal elements (e.g., K_1 and K_2 in Figure 5).

Proposition 3.3: The poset formed by the set of out-tree clusterings of an out-tree and the $\leq$ relation has only <u>one maximal out-tree clustering</u>.

Proof: Immediate from the set of all out-tree clusterings of an out-tree which together with the $\leq$ relation form a lattice (from Proposition 3.1). ***Q.E.D.***

Proposition 3.4: The poset formed by the set of out-tree clusterings of an out-necklace and the $\leq$ relation has <u>K maximal out-tree clusterings</u> if the number of arcs in the only cycle in the out-necklace is K.

Proof: Each vertex in an out-necklace has indegree 1, while each vertex in an out-tree also has indegree 1 except for the root which has indegree 0. Thus, a maximal out-tree clustering can be obtained by removing a minimum of one arc from an out-necklace. But not all sets of clusterings generated by removing one arc from the out-necklace are out-tree clusterings. It is only those that are formed by removing an arc in the cycle of the out-necklace are out-tree clusterings. Thus, if there are K arcs in the out-necklace, there are K maximal out-tree clusterings. *Q.E.D.*

Proposition 3.5: The poset formed by the set of out-tree clusterings of an acyclic digraph and the $\leq$ relation has

$$L = \prod_{i=1}^{p} a_i \text{ maximal out-tree clusterings}$$

if the acyclic digraph has p m-indegree vertices where each has indegree a_i.

Proof: Let the indegree of the set of m-indegree vertices in the given acyclic digraph be $a_1, a_2, \ldots, a_p$. To form a maximal out-tree clustering from the acyclic digraph requires the removal of the "redundant" arcs that exist in each of the m-indegree vertices so that every m-indegree vertex can be reduced to have indegree 1. There are

$$\binom{a_1}{a_1-1} \cdot \binom{a_2}{a_2-1} \cdot \ldots \cdot \binom{a_p}{a_p-1}$$

$$= a_1 \cdot a_2 \cdot \ldots \cdot a_p \text{ combinations.}$$

Thus, there are $\prod_{i=1}^{p} a_i$ maximal out-tree clusterings. *Q.E.D.*

Proposition 3.6: The poset formed by the set of out-necklace clusterings of a general digraph and the $\leq$ relation has at most

$$L = \prod_{i=1}^{p} a_i \text{ maximal out-necklace clusterings}$$

if the general digraph has p m-indegree vertices where each has indegree a_i.

Proof: Similar to the proof given for Proposition 3.5. *Q.E.D.*

The $ODC_{t°g}$, $ODC_{t°a}$ and $ODC_{t°n°}$ problems are to find a set of bounded out-tree clusters from a general digraph, acyclic digraph and out-necklace respectively such that the total value of the set is optimized.

To solve $ODC_{t°a}$ and $ODC_{t°n°}$, the maximal out-tree clustering concept and the $ODC_{t°t°}$-1 discussed in section 2 are used. A maximal out-tree clustering induces a set of largest out-tree clusters that can be obtained from an acyclic digraph or out-necklace by removing minimal number of arcs from the m-indegree vertices. Since any possible out-tree clustering, including the optimal one, ≤ one of the maximal out-tree clustering (or can be obtained by additional arc removals of the induced clusters), the optimal solution can thus be found in one of the maximal out-tree clusterings. That can be done by using $ODC_{t°t°}$-1. The optimal out-tree clustering of an acyclic digraph or out-necklace is therefore the maximumm value one among the set of optimal out-tree clusterings obtained from all maximal out-tree clusterings. An algorithm for these two problems can be formulated as follows:

ALGORITHM: $ODC_{t°a}$-1/$ODC_{t°n°}$-1

INPUT: an acyclic digraph/out-necklace G;

OUTPUT: an optimal out-tree clustering and its value;

PROCESS:

1) FOR each maximal out-tree clustering of G which induces one or more out-tree clusters DO:

   1.1) apply algorithm $ODC_{t°t°}$-1 to obtain an optimal solution for each out-tree cluster;

1.2) IF the sum of the optimal out-tree clustering values of these out-tree clusters is the largest value so far, keep the solution;

END;

2) RETURN the last solution kept;

END.

An algorithm for $ODC_{t\circ g}$, named $ODC_{t\circ g}$-1, is similar to algorithm $ODC_{t\circ a}$-1 except that instead of applying $ODC_{t\circ t\circ}$-1 in step 1.1, $ODC_{t\circ n\circ}$-1 is used and all occurrences of maximal out-tree clusterings are changed to maximal out-necklace clusterings.

Proposition 3.7: The $ODC_{t\circ n\circ}$-1, $ODC_{t\circ a}$-1 and $ODC_{t\circ g}$-1 algorithms find

1) the optimal out-tree clustering of an out-necklace in $O(n\cdot Z^2\cdot K)$ time;

2) the optimal out-tree clustering of an acyclic digraph in $O(n\cdot Z^2\cdot L)$ time; and

3) the optimal out-tree clustering of a general digraph in

$O(n\cdot Z^2\cdot \bar{K}\cdot L)$ time.

Proof:

1) In step 1 of $ODC_{t\circ n\circ}$-1, a given maximal out-tree clustering induces only one out-tree cluster. Thus, steps 1.1 and 1.2 take $O(n\cdot Z^2)$ by Proposition 2.2. Since there are K maximal out-tree clusterings to search for (see Proposition 3.4) in the step, the time complexity is $O(n\cdot Z^2\cdot K)$.

2) In step 1 of $ODC_{t\circ a}$-1, let a given maximal out-tree clustering contain w out-trees with $n_1,n_2,\ldots,n_w$ vertices respectively in them. Steps 1.1 and 1.2 take $O(n_1\cdot Z^2)+O(n_2\cdot Z^2)+\ldots+O(n_w\cdot Z^2) = O(n\cdot Z^2)$ time to

find an optimal solution. Since there are L maximal out-tree clusterings to search for (see Proposition 3.5) in step 1, the time complexity is $O(n \cdot Z^2 \cdot L)$.

3) In step 1 of $ODC_{t \circ g}$-1, let a given maximal necklace clustering contain w necklaces with $n_1, n_2, \ldots, n_w$ vertices respectively in them. Steps 1.1 and 1.2 take

$$O(n_1 \cdot Z^2 \cdot K_1) + O(n_2 \cdot Z^2 \cdot K_2) + \ldots + O(n_w \cdot Z^2 \cdot K_w) = O(n \cdot Z^2 \cdot \bar{K})$$

where $\bar{K}$ is the maximum of all K_i, $1 \le i \le w$. Since there are L maximal out-necklace clusterings to search for, the time complexity is $O(n \cdot Z^2 \cdot \bar{K} \cdot L)$. *Q.E.D.*

Example: The general digraph shown in Figure 6a contains the two maximal out-necklace clusterings that are in Figures 6b and c.

Applying $ODC_{t \circ g}$-1 to the general digraph shown in Figure 6a, we find the optimal solution in Figure 6b which has a total value of 16 and 2 clusters:

Cluster 1 contains vertices 1 and 3.

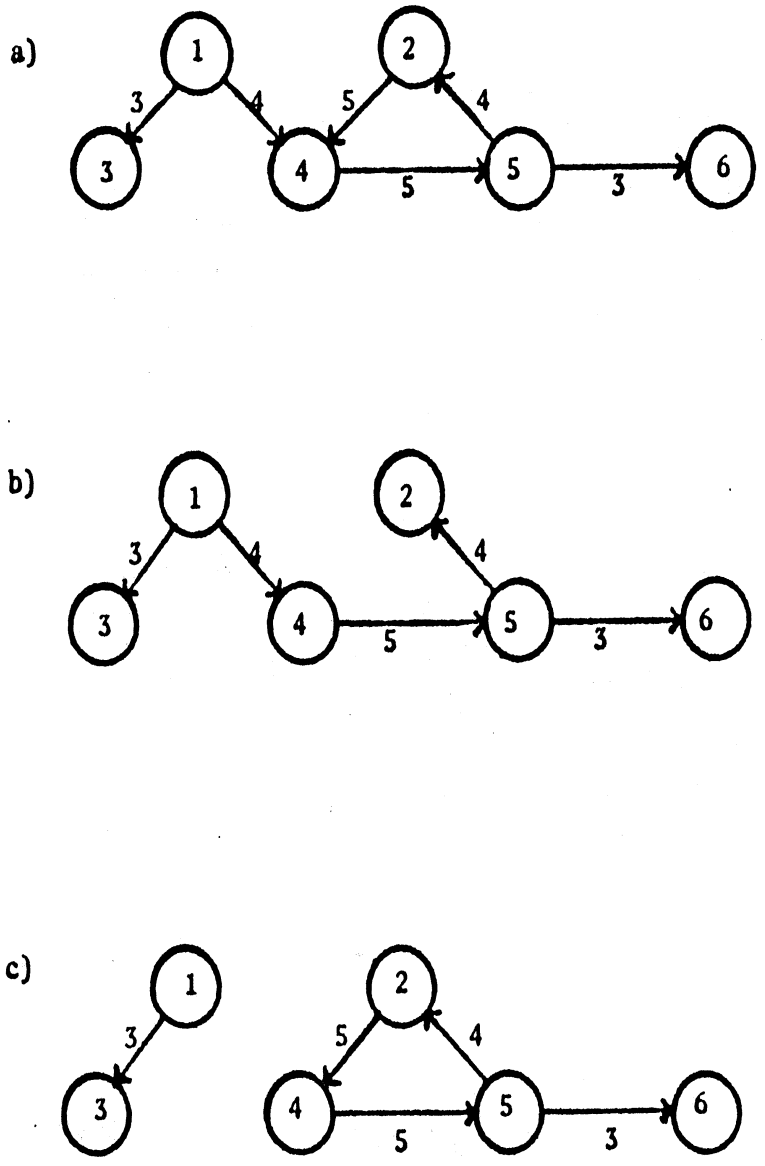
Cluster 2 contains vertices 2, 4, 5 and 6.

4. An Example Application to Database Design

Let the logical database structure given by Teorey and Fry [TeFr80] be represented by an acyclic digraph shown in Figure 7a. The record length (s_i) and the expected number of occurrences (n_i) of each record type v_i is shown in the following table:

record type	record length	expected no. of occurences
PLANT	60	50
JOB_TYPE	26	1K
WORK_TASK	13	200K
EMPLOYEE	80	100K
EMP_DATA	36	100K

The usage information which is characterized by the



B = 3

$\forall v_i$ (s=1) & (n_i=1)

Figure 6 A general digraph and its two maximal out-necklace clusterings.

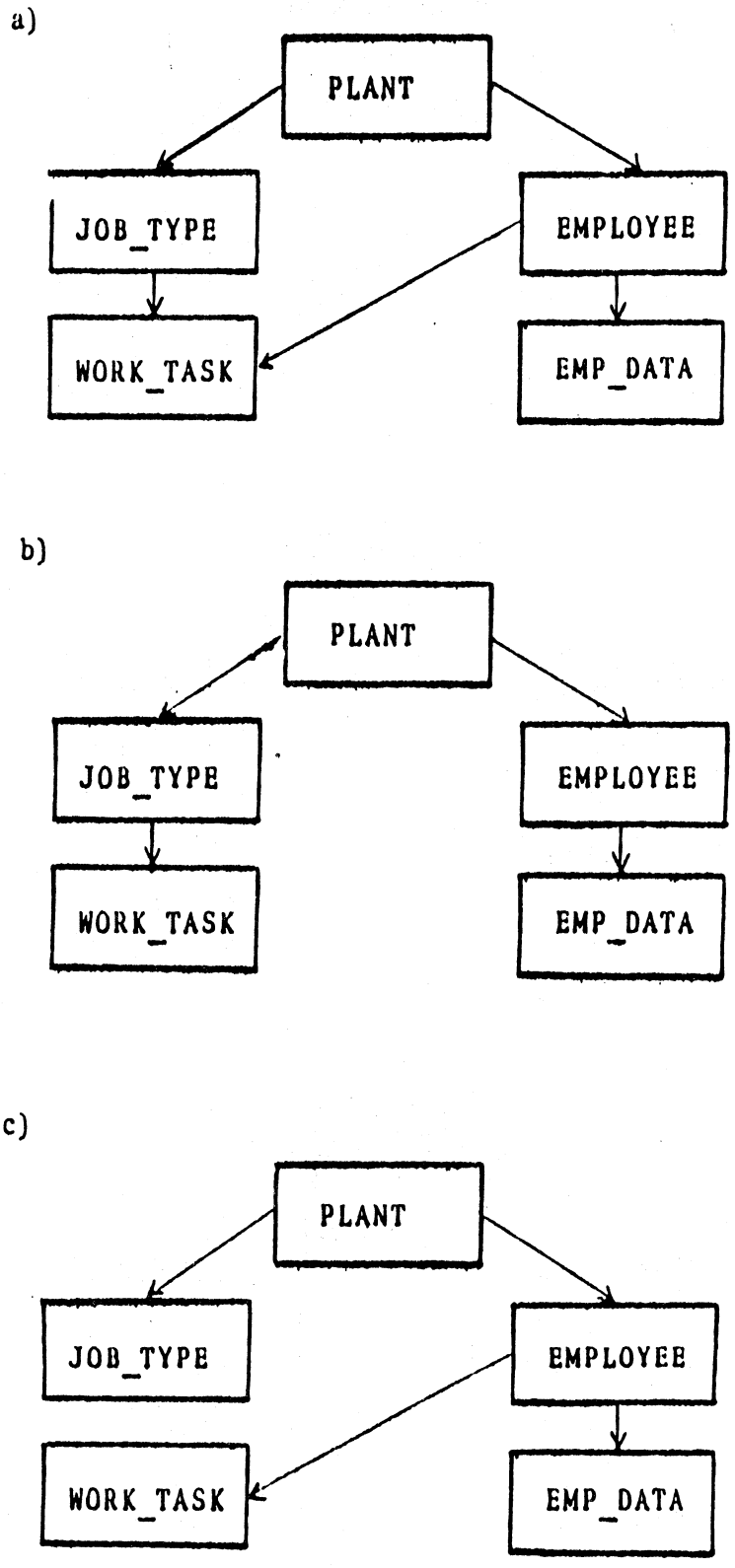


Figure 7 A database schema and its two maximal out-tree clusterings.

frequency and the number of logical record accesses (LRAs) to the database schema by each application is defined in the following table:

application	frequency	LRAs
T1	100/day	1 PLANT+1 EMPLOYEE+1 EMP_DATA
T2	100/day	1 PLANT+1 EMPLOYEE+1 EMP_DATA
T3	200k/day	1 EMPLOYEE+ 1 WORK_TASK+ 1 JOB_TYPE
T4	120k/year	1 EMPLOYEE
R1	Quarterly	100k EMPLOYEE +100k EMP_DATA
R2	Annually	50 PLANT + 100k EMPLOYEE
R3	Monthly	50 PLANT
R4	Monthly	50 PLANT + 50·20 JOB_TYPE
R5	Daily	50 PLANT + 100k EMPLOYEE + 200k WORK_TASK + 200k JOB_TYPE
R6	Monthly	50 PLANT+100k EMPLOYEE+100k EMP_DATA
R7	1000/day	1 EMPLOYEE

Let f_{ij} be the transition frequency between v_i and v_j. A table showing f_{ij} can be constructed using the frequency and LRA of each application as follows:

(We assume: 1 year=240 working days, 1 quarter=60 working days, 1 month= 20 working days.)

(v_i, v_j)	f_{ij}	from application
PLANT, EMPLOYEE	0	
EMPLOYEE, EMP_DATA	100+100+100k/60 +100k/20=6.86k	T_1, T_2, R_1, R_6
PLANT, JOB_TYPE	1k/20=0.05k	R_4
JOB_TYPE, WORK_TASK	200k+200k=400k	T_3, R_5
EMPLOYEE,WORK_TASK	200k+200k=400k	T_3, R_5

Let the block length B be 2,000 bytes. Since the given database schema can be represented as an acyclic digraph and the desired clusters are out-tree, the $ODC_{t \cdot a}$ presented in Section 3 is applied here. The following traces the results produced by each major steps:

step	action performed

1 Obtain the first maximal out-tree clustering (see Figure 7b).

1.1 Apply $ODC_{t^{\circ}t^{\circ}}$-1 to obtain 3 clusters:

cluster 1 contains: PLANT, JOB_TYPE

cluster 2 contains: WORK_TASK

cluster 3 contains: EMPLOYEE, EMP_DATA

with value $F(K)= 0.05k + 6.86k = 6.91k$.

1.2 Keep the solution (because this is the first result).

1 Obtain the second maximal out-tree clustering (see Figure 7c).

1.1 Apply $ODC_{t^{\circ}t^{\circ}}$-1 to obtain 2 clusters:

cluster 1 contains: PLANT, JOB_TYPE

cluster 2 contains: EMPLOYEE, WORK_TASK, EMP_DATA

with value $F(K)= 0.05k + 400k + 6.86k = 406.91k$.

1.2 Keep the solution (because 406.91k is greater than 6.91k).

2 RETURN the kept solution with an $F(K)=406.91k$.

Note: Without clustering, the number of inter-block PBAs is 0 + 6.86k + 0.05k + 400k + 400k = 806.91k, while with clustering, it is 806.91k - 406 = 400k. This is a savings of (806.91-400)/806.91 = 50.4%.

In conclusion, we have shown that efficient record clustering algorithms can be developed to optimize performance for database models represented as networks, hierarchies, or as separate relations. Globally optimal solutions are obtainable for typical database structures found in practice.

Appendix A. Definitions of Graph Theory Terminology

A graph G=(V,E) is a collection of vertices V and a collection of arcs E. If the arcs in E have directions, the graph is called a directed graph or digraph. If the arcs in E have no directions, the graph is called an undirected graph . An arc is denoted by the two vertices it connects. In a digraph, every arc has a direction from the first vertex to the second. The first vertex is called the tail and the second vertex the head of the arc. In an undirected graph, an arc has no direction and the order of the two vertices in an arc has no meaning.

A path is a sequence of arcs $(v_0,v_1),(v_1,v_2),\ldots,(v_{n-1},v_n)$ such that consecutive arcs in the sequence have a common vertex and each arc appears only once in the arc sequence. A cycle is a path for which the first vertex v_0 and the last vertex v_n in the sequence are the same. A graph is connected if every pair of vertices are joined by a path.

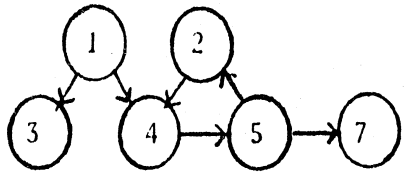
A subgraph of G is a graph G'=(V',E') such that V' is a subset of V and E' consists of arcs (v_r,v_j) in E such that both v_r and v_j are in V'.

For a digraph, the outdegree od(v) of a vertex v is the number of arcs which has v as the head; indegree id(v) of v is the number of arcs which has v as the tail. A vertex which has indegree/outdegree greater than one is caled a m-indegree/m-outdegree vertex. For an undirected graph, the degree of a vertex v is the number of arcs incident with v.

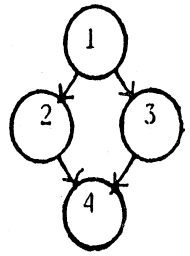
In this paper, ony four classes of graphs are being considered. They are: (see Figure 8)

general digraph (G_g) -- a digraph that may have m-indegree vertices, m-outdegree vertices and cycles in it;

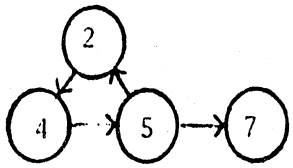
acyclic digraph (G_a) -- a digraph that may have m-



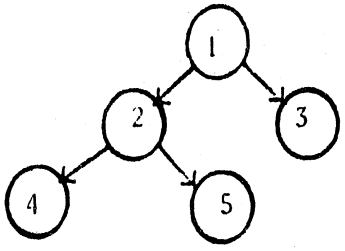
general digraph



acyclic digraph



out-necklace



out-tree

Figure 8 Four classes of digraphs.

indegree and m-outdegree vertices but no cycles;

out-neckace ($G_{n\circ}$) -- a digraph that may have m-outdegree vertices and one cycle in it, but no m-indegree vertices;

out-tree ($G_{t\circ}$) -- a digraph that may have m-outdegree vertices in it but no m-indegree vertices and no cycles;

Appendix B. Proof of Proposition 2.1

A clustering K is an assignment function which assigns either 0 or 1 to each arc e_i of E. Let a triplet formed in a new arc stage j be represented by a sequence of pairs:

$$K_j = [e_1,a_1],[e_2,a_2],\ldots,[e_j,a_j]$$

where the first entry of each pair represents the arc considered in the new arc stage and the second entry represents the assignment on that arc.

Let K_r and K_p be two feasible triplets formed in the rth and pth stages respectively. We define a derivation of a feasible triplet K_p from K_r as the sequence

$$[e_{r+1},a_{r+1}],[e_{r+2},a_{r+2}],\ldots,[e_p,a_p]$$

Let K_{k-1}' and K_{k-1}'' both be formed in the same new arc stage. Let $F(K_{k-1}') \geq F(K_{k-1}'')$ and $L(K_{k-1}') \leq L(K_{k-1}'')$. Assume that there exists a feasible triplet K_{m+n}'' derived from K_{k-1}'' that has a higher value than any feasible triplet derived from K_{k-1}'. We now show that this is impossible.

Let a derivation of K_{m+n}'' from K_{k-1}'' be

$$[e_k,a_k],[e_{k+1},a_{k+1}],\ldots,[e_{m+n},a_{m+n}].$$

a triplet k_{m+n}' can be formed from K_{k-1}' with the derivation $[e_k,a_k],[e_{k+1},a_{k+1}],\ldots,[e_{m+n},a_{m+n}]$ such that

1) the triplet K_{m+n}' is feasible, and

2) $F(K_{m+n}') \geq F(K_{m+n}'')$.

The triplet K_{m+n}' is feasible because in every new arc stage from s=k to m+n, the cluster containing the root of the subgraph in K_s' has equal or lesser length than that of K_s''. This is immediate from the fact that for each s=k to m+n, if a_s=0 then both K_s' and K_s'' have the same weight. If a_s=1 then

$$L(K_s') = L(C_1) + [n_1'/n_1 \cdot L(C_1')]$$

$$L(K_s'') = L(C_1) + [n_1'/n_1 \cdot L(C_1'')]$$

$$L(K_s") - L(K_s') = n_1'/n_1 \cdot [L(C_1)"-L(C_1')] \geq 0$$

$F(K_{m+n}') \geq F(K_{m+n}")$ is immediate from the fact that $F(K_{k-1}') \geq F(K_{k-1}")$ and the fact that k_{m+n}' and $K_{m+n}"$ is formed from K_{k-1}' and $K_{k-1}"$ with the same derivation sequence: $[e_k,a_k],[e_{k+1},a_{k+1}],\ldots ,[e_{m+n},a_{m+n}]$.

REFERENCES

Aho74 Aho, A.V. et al. The Design and Analysis of Computer Algorithms. Reading, Mass: Addision-Wesley; 1974.

ChTe82 Chiang, Wan P. and Teorey, Toby J. A Method for Database Record Clustering. Proceedings of the Conference on Information System; December 13-15, 1982: p. 57-73.

Chia84 Chiang, W.P. Optimal Graph Clustering Problems with Applications to Information System Design. Ph.D. Thesis, University of Michigan, Michigan; May 1984.

GaJo79 Garey, M.R. and Johnson, D.S. Computers and Intractability, a Guide to the Theory of NP-Completeness. San Francisco, Calif: Freeman; 1979.

Marc83 March, Salvatore T. Techniques for Structuring Database Records. Computing Surveys, Vol.15, No.1; March 1983.

Schk77 Schkolnick, M. A Clustering Algorithm for Hierarchical Structures. ACM Trans. Database Syst., Vol.2, No.1; 1977.

TeFr80 Teorey, T.J. and Fry, J.P. The Logical Record Access Approach to Database Design. ACM Computering Suveys, Vol.12, No.2; June 1980: p. 179-211.

TeFr82 Teorey, T.J. and Fry, J.P. Design of Database Structures. Englewood Cliffs, N.J.: Prentice-Hall; 1982.