# Implementing Hypertext Database Relationships through Aggregations and Exceptions

*Yoshinori Hara* [*]
*Arthur M. Keller* [†]
*Peter K. Rathmann* [‡]
*Gio Wiederhold* [§]

## Abstract

In order to combine hypertext with database facilities, we show how to extract an effective storage structure from given instance relationships. The schema of the structure recognizes clusters and exceptions. Extracting high-level structures is useful for providing a high performance browsing environment as well as efficient physical database design, especially when handling large amounts of data.

This paper focuses on a clustering method, *ACE,* which generates aggregations and exceptions from the original graph structure in order to capture high level relationships. The problem of minimizing the cost function is NP-complete. We use a heuristic approach based on an extended Kernighan-Lin algorithm.

We demonstrate our method on a hypertext application and on a standard random graph, compared with its analytical model. The storage reductions of input database size in main memory were 77.2% and 12.3%, respectively. It was also useful for secondary storage organization for efficient retrieval.

[*] e-mail address: hara@eclipse.stanford.edu
[†] e-mail address: ark@eclipse.stanford.edu
[‡] e-mail address: rathmann@eclipse.stanford.edu
[§] e-mail address: gio@cs.stanford.edu

**Keywords:** hypertext database, physical database design, database clustering, overview diagram, aggregation, exception

# 1 Introduction

Hypertext has been widely promoted as a medium for future information handling [YHMD88]. Its navigational interface, *browsing,* and its very simple structure, *nodes and links,* allow users to handle information easily. In some heterogeneous and cooperative environments, the strategy of using a bottom-up organization for combining multiple hypertext structures is quite natural.

When handling large amounts of data, however, several intrinsic problems occur in hypertext structure. Some of them are well known as the disorientation problem in the utilization phase [Conk87], and the lack of design principles for *associative links* in the authoring phase [MoBB90, HaKW91]. In addition, there exists a trade-off between the cost reduction of hypertext authoring/maintenance and the improvement of end-user benefits [HaKa90]. These problems lead to the decrease of information usability, in particular for usage over long periods of time.

In order to solve the above problems, this paper proposes a technique to combine hypertext structure with database models. The advantage of database models when handling large amounts of data is that the models specify a schema for the data. By using a database schema, users can utilize the facilities as a query language interface, views, and efficient storage structures.

This paper focuses on the the problem of extracting a global structure from given hypertext data, which may be treated as an induction method from the hypertext model into the relational/E-R model. In order to demonstrate the usefulness of this method, this paper also shows an optimum solution of the problem using an analytic model. Some experimental results are presented, one for a hypertext application, and one for a randomly generated graph.

Section 2 describes the problems and the requirements of navigational access for large information spaces. In Section 3, we propose a clustering model for relationships using aggregation and exceptions from a large hypertext structure, which is called ACE (Aggregation Clustering with Exceptions). Sections 4 and 5 present the formulation of ACE and a heuristic method to find the optimal solution, respectively. Section 6 gives the experimental results.

For the bulk of this paper, we will assume the hypertext structure to be a directed graph. This simplifies many of the algorithms and examples. Section 7 discusses the extension of this model into a more general hypertext/hypermedia structure.

# 2 Navigating Large Information Spaces

The major advantage of hypertext structure is that it provides navigational access for users. In practical systems, however, this advantage has only been proven in relatively small or very sparse structures. Here, we describe the problems and the requirements that arise when providing such navigation for large amounts of possibly dense data.

## 2.1 Problems of a large hypertext structure

When handling a large amount of hypertext data, such as an electronic encyclopedia [HaKa88], technical manuals [FrCo89], or cooperative environment documents [CoBe88], the following problems arise:

**(1) Disorientation problem**

Users cannot recognize where they are in the information space and they cannot find where they should go next [Conk87]. This common problem is caused by the difficulty in capturing spatially large amounts of complex structures, although spatial recognition is very useful to understand relatively small structures.

This problem can also arise when more than one user works on a shared hypertext. This is because each user has his/her own individual views, which may differ from the author's. Some standardization approaches have been proposed [MoBB90]; however, they are still insufficient for flexible and dynamic changes of view.

**(2) Lack of design principles for associative links**

The hypertext structure has two types of relationship links, namely, *organizational links* and *associative links*. The former are represented by the relationship between a document and its subdocuments, which form a hierarchy. The latter are represented by relationships among documents and may be organized as a network or a loop structure.

Some aggregation techniques have been applied to the regularization of organizational links using "is-a" and "is-part-of" link types. Their inheritance mechanisms enables system designers to capture more information efficiently.

However, the regularization of associative links, which are essential to information captured in hypertext structure, still remains a problem. A theoretical and analytical evaluation of this problem is necessary for both conceptual and physical designs.

**(3) Trade-off between organization costs and utilization benefits**

When a simple and sparse linking strategy is adopted, users are limited to browsing within very small portions of information spaces. On the other hand, when system designers attempt to provide a wide variety of information retrieval, costs in organizing and updating such systems rise accordingly. This trade-off exists in systems of any size, particularly in large systems [HaKa90].

## 2.2 Requirements for an effective and efficient navigational model

The following are requirements for improving the effectiveness and the efficiency of navigation in large information spaces.

**(1) Providing overview diagrams**

Graphical browsers, showing a subset of hypertext structure, are useful to users, just as maps are useful to tourists. Overview diagrams, i.e., graphical browsers using aggregation, are indispensable when the information space is too large for every node and link to be shown on a single map [Niel90].

Several improvements in browsing have been reported from human interface work. Guided tours [Trig88] and the fisheye view [Furn86] are typical examples. However, there are very few studies on such aggregations from the viewpoint of data structures. A structured hypertext mechanism [Fein88] has been proposed that uses the concept of link inheritance and a clustering operation. However, the applicable graph structure is restricted and no general operation has been proposed. This approach assumed that such clustering was done by hand.

### (2) Translating into set-oriented representation

When handling dense hypertext structures, set-oriented modeling such as relational algebra is indispensable. In other words, a link is connected to relevant set of nodes, instead of only one node. *Virtual link* [RoMN81], *Set-to-Set link* [HaKa90], and *Transient Hypergraph* [WaSh90] are based on this modeling. They are a type of query language interface and the target virtual node corresponds to the view in the relational model.

However, translation from a static hypertext structure into a set-oriented representation is not yet available. The translation of combining graph structure with E-R representation would make it possible to apply database maintenance techniques for large hypertext systems.

### (3) Providing high-level semantic structure

Relationship constraint techniques are useful to support hypertext link designs. They are similar to the constraint techniques in relational modeling or semantic data modeling. Once we combine hypertext structures with these database modelings, we can apply such techniques as functional dependencies among attributes and one-to-one/many-to-one constraints among relationships to hypertext modeling.

In addition, the view concept in the relational model is also useful to provide a flexible information handling for individual users. *Web* [UtYa89], a sub-graph structure specified by the attributes in associative links, can be treated as a view in the hypertext model. Therefore, there is good possibility of providing a general user-customized view in hypertext modeling.

### (4) Efficient indexing technique for many-to-many and dense relationships

*Single multiple-attribute index* [ClGa90] has been proposed as an efficient indexing for tree-structured databases. However, an appropriate indexing technique for network or loop structure is also necessary as an alternative to the tree-structured case.

In addition, there is also a requirement for a compact representation of dense graphs. Several compact representations for transitive closure have been proposed [AgBJ89, CrNo89]. However, the compact representation for non-transitively closed dense graphs is still required for large hypertexts.

# 3 Hypertext Relationship Aggregation with Exception

We propose a model of hypertext relationship aggregation in order to satisfy the above requirements. After showing the examples, we propose a clustering method for relationships using aggregation and exceptions.

## 3.1 Example 1 : A student-enrollment relationship

Let us consider a simple example of a relationship between students and the classes they take. This relationship is a many-to-many connection, and the number of links is on the order of $m * n$, where $m$ is the number of students and $n$ is the number of classes, as illustrated in Fig. 1(a). The more classes the students take, the more dense the connection becomes.

However, since many actual links in practical systems are correlated, we can often simplify the links by appropriate clustering. Figure 1(b) shows one possible clustering of each node. Most of the classes are taken by the students who belong to the same department as the classes. In some cases, some basic classes are taken by the students who belong to more applied side; e.g., Computer Science students take Mathematics classes as in Fig. 1(b).

Of course, some students do not take all the same classes as the students in the same category. When the number of such exceptions is relatively few, we can provide a simple expression useful both for physical database design and for overview diagrams.

## 3.2 Example 2 : A citation hierarchy

Consider a citation hierarchy in technical papers. As illustrated in Fig. 2(a), papers may refer to other papers. Some refer to more basic papers and organize as a hierarchy. Some refer mutually.

By fragmenting appropriately, the structure can be simplified, since related papers would quote similar papers. The global reference flow can also be emphasized, as shown in Fig. 2(b). Note that there is a category in which most papers do not quote mutually, however, citing or cited categories are the same.

Figure 1: A Student-Enrollment Relationship

Figure 2: A Citation Hierarchy

Figure 3: Aggregation Clustering with Exceptions

### 3.3 ACE : Aggregation Clustering with Exceptions

Our idea is to consider an effective clustering by allowing exceptions, in order to recognize a global relationship, as illustrated in Fig. 3. Compared with similar clustering techniques such as block diagonal methods, the main characteristic of the proposed method, ACE (Aggregation Clustering with Exceptions), is that it emphasizes the induction of aggregated connections from the actual hypertext graph. The purpose of existing clustering methods is to minimize the number of cut connections, which are the sum of the connections among clusters; therefore, the resulting connections may be unclear.

Taking into account both aggregations and exceptions provides a more natural and informative high-level structure. We must balance the scope of the clusters with the number of remaining exceptions. This graph compaction leads to the efficient representation of a physical structure. In addition, the aggregation is easy to understand for users and satisfy the requirements for a human interface representation, when used as the basic structure of an effective overview diagram.

Figure 4 shows an E-R diagram of ACE. There are two types of ACEs, depending on whether the related element sets are disjoint or not. Example 1 is a disjunctive case, i.e., *many-to-many relationship aggregation.* Example 2 is a conjunctive case, i.e., *self relationship aggregation.*

Figure 4: E-R Diagram of ACE

# 4 Formulation of ACE

In this section, we define the ACE problem and present an example. We only consider associative hypertext structures and denote them as directed graphs. A practical algorithm for ACE is presented in Section 5.

## 4.1 Problem definition

The input to ACE is a directed graph $G = (V, E)$ where vertices $V$ are nodes and edges $E$ are links in a hypertext structure. The target output graphs are *an aggregation graph* $G_A = (V_A, E_A)$ and *an exception graph* $G_X$, which together can be used to compute the input graph.

The aggregation graph $G_A$ is a hypergraph which has hypervertices $V_A$ and hyperedges $E_A$. The set $V_A = \{V_{A_1}, \ldots, V_{A_S}\}$ is a partition of $V$. The set $E_A$ contains the edges that connects the partition sets $V_A$, that is, $E_A \subseteq V_A \times V_A$.

The exception graph $G_X$ represents the difference between the simplified aggregation graph $G_A$ and the actual input graph $G$. There are two types of exceptions. One, called *an inclusive link,* is a minor input edge that has no corresponding hyperedge, and the graph $G_I = (V, E_I)$ is called *an inclusive graph.* The other, called *an exclusive link,* is used when there is no input edge between the two nodes, but there is a corresponding hyperedge. The graph $G_E = (V, E_E)$ is called *an exclusive graph.*

Whether a hyperedge is created or not depends on the majority of relationships between the two corresponding hypervertices. If there are more than some threshold level of input edges between them (e.g., threshold level = 0.5), the hyperedge is created.

The strategy to extract such an aggregation graph and an exception graph so as to make these graphs as simple as possible. The criteria for simplicity in this case may be the size of a graph, such as the number of vertices and the number of edges, or the number of edges only. Translating the source graph to such simple graphs is useful for the grasp of original data as well as for the compaction of the data.

The problem definition of extracting the global schema structure is described as follows:

Input: Input graph $G = (V, E)$<br>
Output: Aggregation graph $G_A = (V_A, E_A)$,
Inclusive graph $G_I = (V, E_I)$, and
Exclusive graph $G_E = (V, E_E)$

In the following, we denote $V_{A_i}$ both as an aggregation node and as a set of input nodes. Its meaning is understandable from the context. The input and output satisfy these constraints:

$$V = \bigcup_i V_{A_i}; \quad \forall i, j,\ i \neq j : \ V_{A_i} \cap V_{A_j} = \phi \tag{1}$$

$$E = (\ E'_A \ - \ E_E) \ \cup \ E_I \tag{2}$$

$$\begin{aligned} E'_A = \{(v_i, v_j) \mid v_i, v_j \in V,\ \ \exists V_{A_i}, V_{A_j} \in V_A, \\ v_i \in V_{A_i},\ v_j \in V_{A_j},\ \ (V_{A_i}, V_{A_j}) \in E_A\} \end{aligned} \tag{3}$$

As illustrated in Fig. 5, the set $E'_A$ is a set of edges in the complete bipartite graph generated by the aggregation graph $G_A$. The objective of ACE is how effectively we can extract the set $E'_A$, i.e., complete bipartite parts.

Figure 5: $E'_A$: A Set of Edges in the Complete Bipartite Graph

For the aggregation process to be most effective we wish to minimize

$$Cost\_func(G_A, G_I, G_E) \;=\; \mid V_A \mid + \mid E_A \mid + \mid E_I \mid + \mid E_E \mid \quad \rightarrow \; Min. (4)$$

Figure 6(a) shows sample input data, which is written as an binary adjacency matrix. The matrix size is equal to the number of vertices and the element $x_{ij} = 1$ means an edge from vertex $i$ to $j$ exists, and 0 means it does not exist.

By applying appropriate clustering on the condition that cost function is minimized, we can get the clustered matrix as in Fig. 6(b). Note that $0^{\#}$ and $1^{*}$ represent an exclusive link and an inclusive link, respectively. These links are exceptions, which describe minor relationships among the clusters.

In order to consider the meaning of ACE, the following equations using relational algebra are satisfied between input and output relations.

$$(R - R_I) \cup R_E = \prod_{R_{M^1.N,}\ R_{M^2.N}} (R_{M^1} \bowtie_{R_{M^1.A} = R_{A.S}} R_A \bowtie_{R_{A.E} = R_{M^2.A}} R_{M^2}) \qquad (5)$$

such that

$$\mid V_A \mid \; + \; \mid R_A \mid \; + \; \mid R_I \mid \; + \; \mid R_E \mid \rightarrow \; Min. \qquad (6)$$

where

$R$ ; the input relation<br>
$R_I$ ; the inclusive exception relation<br>
$R_E$ ; the exclusive exception relation

Figure 6: An Example of ACE

$R_M \equiv R_{M^1} = R_{M^2}$ ; the membership (clustering) relation
$R_A$ ; the aggregation relation
$\mid V_A \mid$ ; the number of values of the membership relation

As shown in Fig. 7, ACE is a clustering method of given relationships by allowing set difference and union operations. Since the membership relation is an extracted attribute of the nodes in the input graph, such clustering as ACE can be treated as a translation method of relationship information into some attribute values of the nodes.

## 4.2 Complexity of ACE

Finding an optimum solution of ACE is NP-complete. That is, ACE is in NP, and there is a polynomial transformation from a known NP-complete problem to ACE. Therefore, efficient heuristic algorithms are indispensable in solving the problem.

**Statement:** Given $G = (V, E)$, integer $K$, are there $G_A$, $G_I$ and $G_E$ representing $G$ such that $|E_A| + |E_I| + |E_E| \leq K$?

First we shall show that ACE is in NP by outlining a nondeterministic polynomial time algorithm to solve ACE. The first step is to nondeterministically guess a partition of the vertices of G into aggregation nodes. This can be accomplished in (nondeterministic) linear time. Then, given such an

Figure 7: A Formulation of ACE using Relational Algebra

aggregation partition, the best possible choices for $G_A$, $G_I$ and $G_E$ can be determined by the rule that if more than half the possible edges between a pair of aggregations actually exist in $G$, then an aggregation edge should be added to $G_A$, and the exceptions added to $G_E$; otherwise no aggregation edge should be added, and the edges of $G$ between the two partitions should be included in $G_I$. Once the graphs $G_A$, $G_I$, and $G_E$ are determined, they can be used to compute the cost function and compare the actual cost to the maximum allowed cost $K$.

To show completeness we reduce from vertex cover (see e.g., [GaJo79], page 46).

Vertex cover may be stated: "Given a graph $G = (V, E)$, and a positive integer $K \leq |V|$, does there exist a vertex cover of size $K$ or less for $G$, that is, a subset $V' \subseteq V$ such that $|V'| \leq K$ and for each edge $\{u, v\} \in E$, at least one of $u$ and $v$ belongs to $V'$?

Given an instance of the vertex cover problem, we will now construct an instance of ACE which will have the same answer. Discussion of the construction, and a proof that it guarantees the same answer are delayed until later. In the construction, we will use two integer parameters $m$ and $n$, which we will instantiate later.

For every vertex of the original graph $v \in V$ we create in $G'$, (the graph for the ACE instance) a set of nodes of size $mn$. When we need to refer to this set collectively, we shall call it the *supernode* derived from $v$. Each such supernode is composed of $n$ cliques each of size $m$. The $nm^2$ edges needed are termed *internal* edges, since they are used to give structure to the supernode.

For each $\{u, v\}$ in the original graph $G$, is represented in $G'$ by the set of all the possible $2m^2n^2$ arcs from nodes in the supernode derived from $u$ to nodes in the supernode derived from v, and also in the reverse direction. We call this collection a *superedge*. Note that the original graph $G$ is undirected, while $G'$ is a directed graph, and that we model this with arcs going both ways.

We then ask the question, is there an ACE representation for $G'$ which has cost function less than or equal to $n|V| + 2n|E| + Kn(m^2 - 1)$? As we shall see, the answer to this question is yes if and only if there is a vertex cover for $G$ of size no greater than $K$.

**Discussion and proof:** First we calculate the cost of some possible ACE representations for $G'$, and use these calculations to show that if $G$ has a vertex cover of size $K$, it can be used to construct an ACE representation for $G'$ with cost no greater than $n|V| + 2n|E| + Kn(m^2 - 1)$.

First, let us consider some possible ACE representations of a cluster of nodes in $G'$.

Let us consider three possible representations of a cluster, which differ in their degree of aggregation. In the first representation, (hereinafter termed "expanded"), there is no aggregation. $G_A$ and $G_E$ are empty, so the cost consists entirely of the $nm^2$ edges in $G_I$. A more compact representation aggregates the cliques within the cluster, yielding $n$ nodes in $G_A$. This representation costs $n$, since each node in $G_A$ has a self loop to indicate that it represents a clique. We shall call this the "normal" state for a cluster, since it results in the lowest possible cost for the node. Although, the normal state results in the lowest possible local cost, further aggregation may sometimes be advantageous. In the "collapsed" state, all the vertices in the cluster are aggregated into a single node. The cost of this collapsed representation is $nm^2$, again for the edges of $G_I$ since we can not represent any of the edges in the cluster with edges in $G_A$. (Adding a self-loop would increase the cost for $n > 1$, since it would add many exceptions.) We summarize these costs in Figure 8.

In general, the graph $G'$ does not consist entirely of isolated clusters. There will also be edges between the clusters. The cost contributed to the

| State | Local Cost |
|---|---|
| expanded | $nm^2$ |
| normal | $n$ |
| collapsed | $nm^2$ |

Figure 8: Internal Cost of Different Configurations of a Cluster.

| State | External Cost |
|---|---|
| EE | $2(nm)^2$ |
| EN | $2n^2m$ |
| EC | $2nm$ |
| NN | $2n^2$ |
| NC | $2n$ |
| CC | $2$ |

Figure 9: External cost of an edge

ACE representation of $G'$ by an edge depends on the representation of the two clusters the edge connects. Figure 9 gives the cost to represent an edge, depending on whether the endpoints are expanded, collapsed, or normal. As we see in Figure 9, the cost of an edge is reduced with greater aggregation of the endpoints.

Thus, we see from figures 8 and 9, that the collapsed is more costly than the normal state for a node viewed in isolation, but the cost of an edge is lessened if at least one of the nodes it connects is collapsed. The expanded state is always more more costly for both edges and nodes. The total cost to represent an edge and its two endpoints is given in Figure 10. Since the expanded state is always disadvantageous, that state is not included in Figure 10. Note that as long as $1 < m < \sqrt{2n}$, the minimum overall cost is found when exactly one of the nodes is collapsed. In general, we will find that the lowest overall cost for the graph will be found when each edge between clusters has at least one of its endpoints in a collapsed state.

So, if $G$ has a vertex cover of size $K$, we use it to construct an ACE representation in which the nodes of the cover are collapsed, and those not

| State | Internal Cost | External Cost | Total Cost |
|---|---|---|---|
| NN | $2n$ | $2n^2$ | $2(n+n^2)$ |
| NC | $n+nm^2$ | $2n$ | $2n+nm^2$ |
| CC | $2nm^2$ | $2$ | $2nm^2+n$ |

Figure 10: Total cost of a graph with two vertices and an edge

in the cover are in the normal state. The total cost of such a configuration is $n(|V|-K)$ for the internal cost of the normal nodes, plus $nm^2K$ for the internal cost of the collapsed nodes, plus between $2E$ and $2n|E|$ for the edges. Taking the larger of the two possible costs for the edges, we arrive at a bound for the total cost for the ACE representation of the graph of $n|V|+2n|E|+Kn(m^2-1)$. Thus, if such a vertex cover exists there is a solution to the ACE problem with the required cost function.

Now we must prove the other direction, that if no vertex cover exists of size $K$, then there will be no sufficiently low-cost solution to the ACE problem. As a first step towards seeing this, suppose that we attempt to construct an ACE solution based on a vertex cover of size $K+1$. This means that $G'$ will have a vertex cost of $(K+1)nm^2$ for the collapsed nodes plus $(|V|-K-1)n$ for the normal nodes. The edges must cost at least $2E$, taking the lower bound in which every edge connects collapsed nodes. Thus, the total cost of the ACE representation must be at least $(K+1)nm^2+(|V|-K-1)+2|E|$. This is greater than the allowed cost if

$$(K+1)nm^2+(|V|-K-1)+2|E| > n|V|+2n|E|+Kn(m^2-1),$$

which simplifies to

$$|E|(2n-2) < (n-1)m^2.$$

Setting $m=|E|$, for example, satisfies this requirement. We then choose $n$, mindful of the requirement that $\sqrt{2n} > m$. A choice of $n=|E|^3$ suffices. Smaller values for $n$ and $m$ would work, but these are simple to work with.

With these choices for $n$ and $m$ $G'$ has $|V|nm = |V||E|^4$ vertices and $|E|2(nm)^2 = 2|E|^9$ edges. This means that $G'$ is substantially expanded from $G$, but that expansion factor is polynomial and the reformulation can be effected in polynomial time.

We still need to show that *no* possible ACE representation of $G'$ will meet the cost requirements if $G$ has no vertex cover of size $K$. We do this by showing that the ACE representation generated by the smallest vertex cover must be close to optimal, and that if the smallest vertex cover has size $K+1$, no ACE representation will have cost less than the cost of the construction for a cover of size $K$.

We now show some simple conditions which govern any optimal ACE representation. Recall that $G'$ has a three level structure. Cliques are grouped together to form supernodes which together with the superedges form the larger graph $G'$.

The first condition is that the nodes within any one clique will always appear in the same aggregate node of $G_A$. To see this, let us assume we have an ACE representation in which nodes $a$ and $b$ belong to the same clique, but belong to (different) nodes $A$ and $B$ in $G_A$. Let $i$ be the number of edges of $G_I$ and $G_E$ which have $a$ as an endpoint, and let $j$ be the number of such edges of which have $b$ as an endpoint. Now if $i < j$, in some sense $a$ "fits better" in its aggregate than $b$ does. Therefore let us move $b$ into the node containing $a$. Note that by the way $G'$ was defined, whenever $a$ is the endpoint of an edge $\{a, x\}$, $b$ is the endpoint of an edge $\{b, x\}$ to the same node. Thus, if we don't change anything else, we will save the $j$ inclusion and exclusion edges needed for $b$ in its original location, at the cost of duplicating the $i$ edges needed for $a$. Since $i < j$, this move results in a net savings. If $i > j$, we do the reverse move, also for a net cost savings. Finally, if $i$ and $j$ are equal, we move either way, for no net change in cost. Thus, if we are looking for lowest cost ACE representations, it is sufficient to look only at those representations in which all the nodes of a clique are always in the same aggregate node.

Second, let us look at the cost of representing a given superedge in $G'$ which connects supernodes $A$ and $B$. One reasonable way of allocating the cost of such an edge is

$$\frac{1}{k}cost(A) + \frac{1}{l}cost(B) + cost(edge)$$

where $k$ and $l$ are the degrees of supernodes $A$ and $B$ respectively, $cost(A)$ refers to how many edges of $G_A$, $G_I$ and $G_E$ are required to represent the internal structure of $A$, and $cost$(edge) is how much of the cost of the ACE representation is caused by representing the superedge from $A$ to $B$.

Each supernode will have some number of "free" cliques, i.e., those cliques not further aggregated. The cost of adding a clique of $A$ into an existing aggregate is $2m^2$, for the edges which will now have to be represented as exceptions. This cost is more than compensated for if the number of free cliques in $B$ is greater than $m^2$, since fewer edges will be needed to represent the superedge between $A$ and $B$. This is certainly the case if there are few cliques aggregated, since each supernode can have as many as $n > m^2$ free cliques. Let us assume that the total number of free cliques in the supernodes $A$ and $B$ is to be held fixed, and we wish to allocate between them how much aggregation is to take place. A look at the function in which such a balancing is to take place shows that there are no local minima, only local maxima, and that any minimal state must occur at an extreme, i.e, with all of the aggregating done in one or the other node. This is not to say that by combining independently arrived at local minima, we generate a global maximum. However, the other direction is true. Since the vertex-cover based construction simultaneously each edge pair at a local minimum, it must therefore produce a global minimum.

Finally, one more detail must be established. With the above construction, we are assuming that no aggregation takes place at a scale larger than a single supernode. Such aggregation may actually be advantageous, but can not make enough of a difference to make a make a sufficiently inexpensive ACE representation in the absence of a vertex cover of size $K$. To see this, note that any such higher aggregation benefit would have to come from a reduced cost of representing superedges. The *total* cost of all superedges in a Vertex-Cover derived representation is $2|E|^4$ . Even if this cost is eliminated entirely, it cannot make up for the $(|E|^3 - 1)|E|^2$ cost of fully collapsing even one additional supernode.

Thus, a sufficiently inexpensive ACE representation exists *only* if the original graph had a vertex cover of size $K$, and this proves that ACE is NP-complete. □

## 4.3 An analytic model

In this subsection, we will show an analytic model of ACE, in order to estimate the several parameters and the effective solution of this clustering.

### 4.3.1 Optimum storage cost for main memory DB

When we consider the physical database design in the main memory, storage demands are important. Here, we will estimate for the optimum solution the database size, i.e., the total number of tuples of the output relation.

For the purpose of this discussion, we will set out the following simplifying assumptions:

(1) The number of nodes in each cluster is the same. That is, the input nodes are equally partitioned into clusters of the same size.
(2) The probability of exclusive links in the corresponding exclusive partitioned area is the same probability of inclusive links in the corresponding inclusive partitioned area.

The following parameters are used to characterize this model.

$n$ ...the number of nodes in the input graph
$p$ ...the fraction of the input links compared with the number of possible links
(i.e., $pn^2$ is the number of links in the input graph)
$q$ ...the average fraction of the exception links in the corresponding partitioned area
(From assumption (2), we derive that the average fractions of the exclusive partitioned area and of the inclusive partitioned area are the same.)
$s$ ...the number of nodes (clusters) in the aggregation graph $G_A$, that is, $\mid V_A \mid$
(By assumption (1), we assume that each cluster size is the same)
$r$ ...the number of links in the aggregation graph
$\alpha$ ...the ratio of $s$ to $n$, that is, $\alpha = s/n$, one over the average number of input nodes per output cluster

Since the objective of this clustering is to reduce the rate of the exceptions in each partitioned area, we propose the following condition:

$$0 \leq q < p \leq 1 \tag{7}$$

Since the total number of input links is equal to the total number of links which are composed of aggregated links plus the total number of inclusive links, the following condition is satisfied:

$$pn^2 = r\,(1-q)\left(\frac{n}{s}\right)^2 \;+\; (s^2 - r)\,q\left(\frac{n}{s}\right)^2 \tag{8}$$

Therefore,

$$r = \frac{p-q}{1-2q} \cdot s^2 \tag{9}$$

The database size, $S$, is calculated by $n$, the number of tuples in the clustering relation plus $(r + qn^2)$, total number of tuples (i.e., links) in the output relations. Therefore,

$$\begin{aligned} S &= \mid R_M \mid + \mid R_A \mid + (\mid R_I \mid + \mid R_E \mid) \\ &= n \;+\; r \;+\; qn^2 \end{aligned} \tag{10}$$

In order to estimate the optimum database size, we have to consider the relationship between $q$, the average fraction of the exception links, and $s$, the number of nodes in the aggregation graph. When $s$ equals to 1, $q$ is equal to $p$. And, when $s$ equals to $n$, there are no exceptions in the output graph, i.e., $q = 0$. Therefore, there is a negative correlation between $q$ and $s$.

Since $q$ is on the order of $s^2$, or $\alpha^2$, we can set out the following condition if the distribution of the links is a uniform distribution and $n$ is relatively large:

$$q = p\,(1 - \alpha^2) \tag{11}$$

By applying equations (9), (11), and the storage function, $\alpha$, equation (10) can be written as follows:

$$\begin{aligned} S &= n + \frac{p\alpha^2}{1 - 2p\,(1-\alpha)^2} \cdot n^2\alpha^2 \;+\; pn^2\,(1-\alpha^2) \\ &= n + \; pn^2 \cdot \left( \frac{(1-2p)\alpha^4 - (1-4p)\alpha^2 + 1 - 2p}{2p\alpha^2 \;-\; 2p \;+\; 1} \right) \end{aligned} \tag{12}$$

By the condition of $\partial S / \partial \alpha = 0$, the optimum solution of $\alpha$ is calculated as follows:

$$\alpha^2_{opt} \;=\; \frac{\sqrt{1-2p} \;-\; (1-2p)}{2p} \tag{13}$$

Figure 11: Comparison of the Database Size $S$ with the Storage Function $\alpha$

That is, $\alpha_{opt}$ is the function of $p$, the fraction of input links, and is independent of $n$, the number of nodes in the input graph. If the value of $p$ is much less than 1, ($p \ll 1$), then $\sqrt{1-2p} \approx (1-p)$, and $\alpha_{opt}^2$ is approximately 0.5. Also, $\alpha_{opt}^2$ approaches zero as $p$ approaches 0.5.

When the input graph is treated as a relatively large graph, i.e., $n \ll pn^2$, and $\sqrt{1-2p} \approx (1-p)$, the following optimum condition is ensured:

$$\begin{aligned} \mid R_I \mid_{opt} &= \frac{1}{2} \cdot pn^2 && (14) \\ \mid R_A \mid_{opt} + \mid R_E \mid_{opt} &\approx \frac{1}{4} \cdot pn^2 && (15) \end{aligned}$$

Since the input database size is $pn^2$, the database size will be reduced by about 25% using ACE. Figure 11 shows the comparison of the database size of the output graph with that of the input graph according to $\alpha^2$.

Of course, the result of optimum solutions depends on the condition (11), the relationship between $q$ and $\alpha$. If the input graph is heavily correlated, we can obtain a much more compact output graph. On the contrary, if the input graph is more random, the reduction gain is small. We will illustrate them with the practical data in Section 6.

Figure 12: A Secondary Storage Organization for ACE

### 4.3.2 A secondary storage organization for efficient retrieval

When we consider physical database design using secondary storage, retrieval cost often dominates, with most of that cost due to page fetching. We show an efficient organization of secondary storage that reduces the input-output cost.

As illustrated in Fig. 12, we can store the aggregation adjacency matrix as a two dimensional array. Each array element contains an aggregation edge bit, a count of the number of blocks used to store the aggregation exceptions for this aggregation edge, and the block ID of the first such block. The exception blocks for one aggregation edge are stored sequentially and can all be fetched in one input-output operation. Each aggregation edge entry can be stored in 4 bytes.

If the aggregation index array and the clustering relation can be stored in the main memory, the average input-output cost will be reduced. Otherwise one input-output operation is required for the aggregation edge entry. Only if there are exceptions to this edge is another input-output operation required to obtain all the exceptions. In Sections 6.1 and 6.2, we will discuss how often this additional operation is required.

## 5 A Heuristic Algorithm for ACE

In order to find an optimal solution for ACE, we developed a heuristic algorithm based on the Kernighan-Lin algorithm [KeLi70]. Kernighan-Lin algorithm is a well-known graph partition algorithm. An input data of a weighted, undirected graph with an even number of nodes can be partitioned into two equal-sized node sets with the condition that the cut, that is, the sum of the weights of all edges that connect nodes in both sets, is minimal.

We modified the Kernighan-Lin algorithm to apply it to the problem of ACE. The differences in our adaptation are: the number of clustered nodes may be more than two and the size of the clusters do not have to all be equal. Our modification uses an extra dummy cluster in each step of the original algorithm. Then, we rearrange clusters. The resulting algorithm is as follows:

### An Extended Kernighan-Lin Algorithm for ACE

```
(Step1) Initial clustering step
            Make all nodes belong to one cluster $C_1$  (the number of clusters:
              $k = 1$)

(Step2) Repeating step
            While no updating occurs do
              Make $C_{k+1}$ as a dummy cluster
              for $i := 1$ to $n$ do  (the number of nodes: n)
                Choose some unselected node and call it $v_i$
                Let $j_c$ be the cluster of $v_i$ (i.e., $v_i \in C_{j_c}$)
                for $j := 1$ to $k + 1$ and $j \neq j_c$ do
                  Calculate the cost when $v_i$ moves into $C_j$
                  Select the pair of $(v_i, C_j)$ if the movement makes the best benefit (i.e., largest decrease in cost)
                end
                Add $(v_i, C_j^{(i)})$ to the list of movement with the best benefit for
                  this group of $n * k$ alternative cases
              end
```

```
    Find $l$ $(0 \le l \le n)$, $s.t. \sum cost_l \to max$
    Perform the translations $(v_1, C_j^{(1)}), (v_2, C_j^{(2)}), \ldots, (v_l, C_j^{(l)})$ ;
      that is, move $v_i$ into cluster $C_j^{(i)}$
    Rearrange clusters ($k$ may change to $(k-1)$ or $(k+1)$)
end
```

# 6 Experimental Results

We applied our method to two types of data. One is a practical hypertext structure having some degree of correlation among its nodes. The other is a randomly generated graph, a published test bed for graph algorithms.

## 6.1 Hypertext on Hypertext

"Hypertext on Hypertext" [Hype89] is one of the typical examples of hypertext structures, and is available on HyperCard [Good87]. Eight articles [CACM88] devoted to hypertext/hypermedia are transformed into a hypertext. The basic structure consists of 313 cards of articles and the links among those cards with their indexing information. We applied our method to "Hypertext on Hypertext" using the following rules:

(1) The cards which have the same keyword are to be connected with each other.

(2) The keywords "hypertext" and "hypermedia" are ignored since they appear in more than one third of the total cards.

Since the original system does not have set-oriented operations, index cards were added to avoid the excess links within the relevant cards. Therefore, some relevant cards could not be directly navigated. We used rule (1) to avoid this limitation.

We have adopted rule (2) in order to improve recall and precision rate because all articles refer to the topics of hypertext and hypermedia and little information can be extracted from such connections.

Table 1 shows the input data we used for the simulation and the results using our method. The results were remarkable for both storage reduction of database size and page fetching. Due to the tight connections among the cards, the storage reduction of input database size (i.e., total number of input links) was 77.2%.

The secondary storage organization also resulted in compression. When we consider a 4K block size (512 bytes), the aggregation index array takes up only 6 blocks ($26 \times 26 \times 4/512$). In addition, two-thirds of the elements were non-exception areas. Consequently, if the aggregation array is stored in memory, only one input-output operation is required for one-third of the time that exceptions exist.

On the other hand, the source graph occupies 24 blocks ($313 \times 313/(512 \times 8)$) and there were two possible alteratives: all blocks should be in main memory, or otherwise one input-output operation is always required.

Table 1: Experimental Results

Figure 13: A Global Structure of “Hypertext on Hypertext” (Cluster size is more than five.)

Figure 13 shows an extracted global structure of “Hypertext on Hypertext.” There are 8 major aggregated nodes, each of which has more than five cards. We assigned concrete meanings to each cluster. Therefore, ACE has much possibility to provide an effective overview diagram, as well as an efficient physical database design.

## 6.2 A standard random graph

The other simulation example is a standard random graph given by Johnson [JASM89]. The standard random graph is defined in terms of two parameters, $n$ and $\tilde{p}$. The parameter $n$ specifies the number of nodes in the graph; the parameter $\tilde{p}$, $0 < \tilde{p} < 1$, specifies the probability that any given pair of vertices constitutes an edge.

Table 1 also shows the input information and the simulation results. Even for a random graph, about 12.3% of the input database size was reduced. The inputs and outputs of the random graph are graphically represented in Figs. 14, 15, and 16. Thus, we see how effectively structures are compacted by ACE even for a random graph.

## 6.3 The comparison with the analytic model

In Section 4.3.1, we described that the reduction of database size is about 25% in the optimum solution when we assume the uniform distribution of

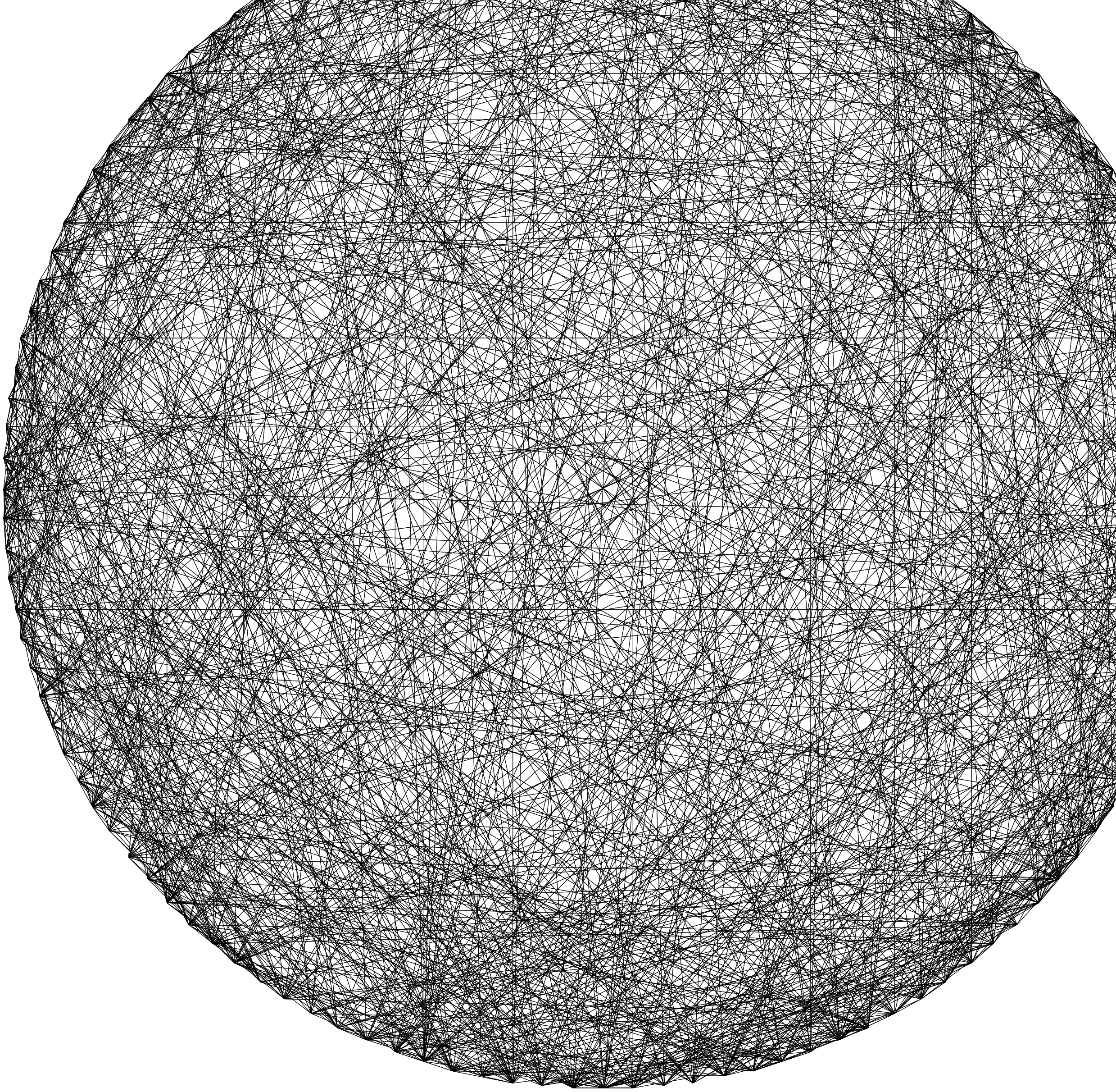

Figure 14: The Input Graph $G$

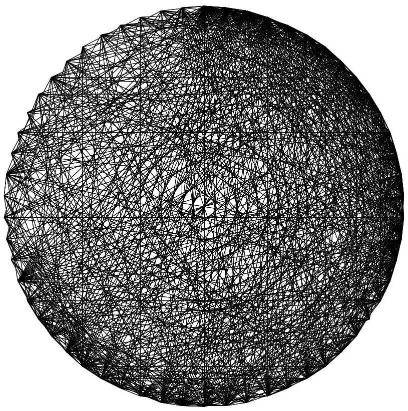

Figure 15: The Output Graph $G_A$ and $G_I$

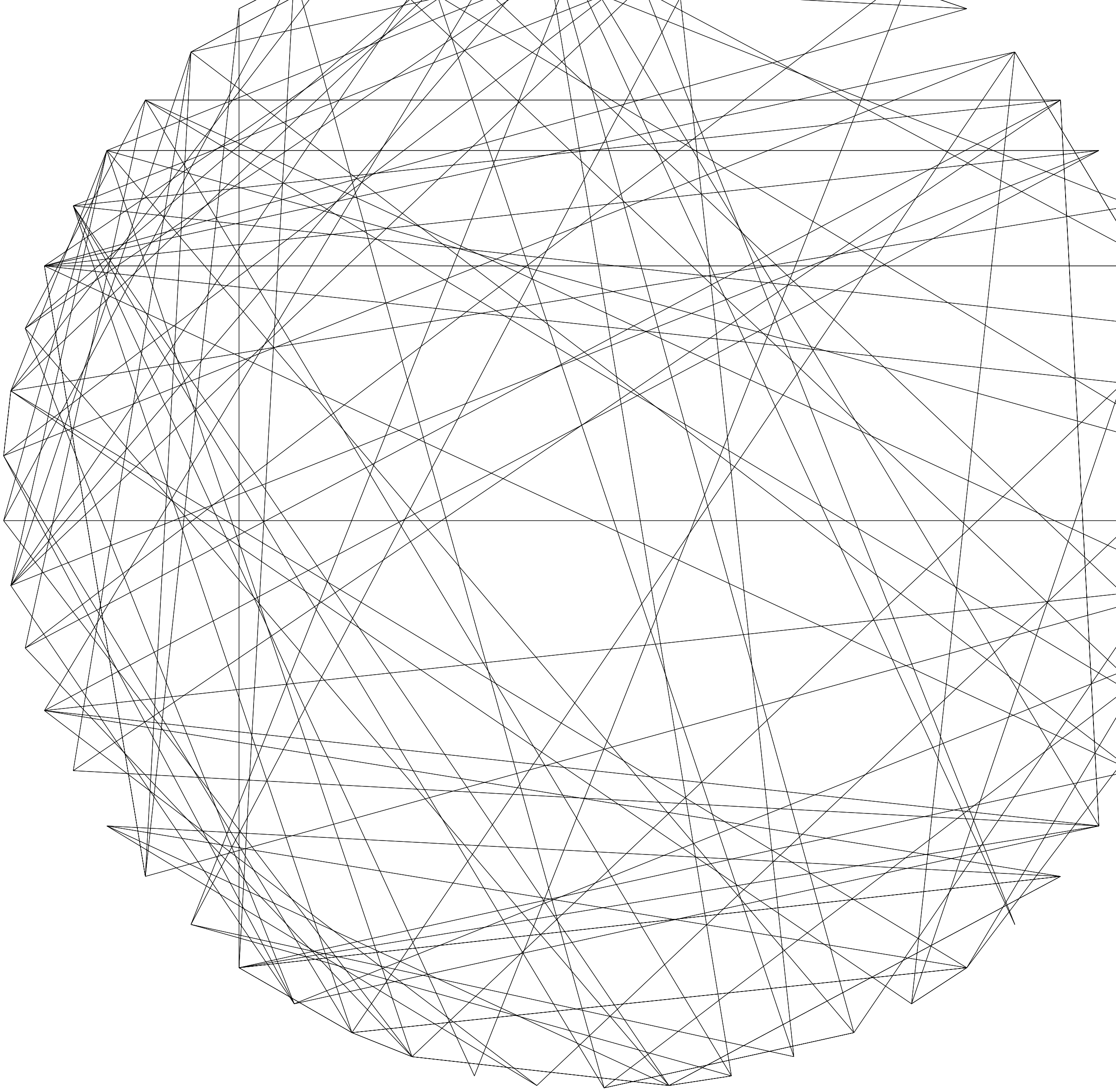

Figure 16: The Ouput Graph $G_A$

input links and evaluate equation (11). There are two main causes for the difference between the analytic model and the experimantal results. The first cause is that the distribution of each simulated graph was somewhat different from the assumption made for equation (11). When we consider $q = F(\alpha^k)$, the graph of "Hypertext on Hypertext" has a $k < 2$, and the random graph has a $k > 2$. The second cause is the difference between the sub-optimal solution of the heuristic algorithm and the optimum solution of the analytic model.

One of the interesting aspects from the comparison is the ratio of the number of links which are composed of aggregated links, to the total number of inclusive links. In the analytic model, the ratio is 1, i.e., balanced. In the experimental results, on the other hand, the ratio in a correlated case is greater than 1 and that in a random case is smaller than 1. This shows that the degree of reduction in ACE depends highly on the extraction of the exclusive areas.

# 7 Discussion and Future work

In this section, we discuss the tuning issues for ACE and the extension into a more general hypertext/hypermedia structure.

## 7.1 Overlapping and recursive clusterings for ACE

The clustering relation between input nodes and output cluster nodes is restricted to a many-to-one relationship in ACE. This limitation allows the size of the membership relation to be equal to the number of input nodes $n$, resulting in a simple output structure. This simple structure leads to efficient searching in secondary memories as well as to a compact representation in main memory.

In some special cases, however, a few extended methods provide more an elegant representation of hypertext relationship aggregations than the aggregation with exceptions only. One extension is to allow *overlapping* clusters, that is, to apply a many-to-many membership relationship between clustered nodes and their elements, instead of the many-to-one relationship. This approach provides a more compact aggregation when some input nodes correlate to nodes in both clusters.

Another extension is to provide for *recursion.* That is, we let our method apply to the data recursively. As a result, a sequence of aggregation graphs and of exception graphs is extracted. Recursive clustering is similar to Principal Components Analysis in multivariate analysis. Since the information of the aggregation graph is orthogonal to its corresponding exception graph, recursive clustering is treated as a decomposition technique of graph data. In particular, it is useful for physical data representation when the same aggregated structure can be applied to multiple levels of aggregation in the recursive hierarchy.

Other extended issues such as combining stochastic approaches [TsNa91], considering weighted links, and the quantitative evaluation for the cost function of ACE, are left for future study.

## 7.2 Other implementation algorithms

As presented in Section 4, the problem of ACE is similar to the graph partitioning problem. The heuristic methods of the graph partitioning problem have been proposed. Johnson, et al. have compared some of them and have evaluated them very precisely [JAMS89]. Simulated annealing may be an alternative implementation algorithm and produce results closer to the optimum, although it would take more time than other heuristic algorithms.

Another implementation algorithm is based on a greedy algorithm. The characteristic of this algorithm is that it is very fast, but it can also fail to find a good result. Therefore, the latter would be more applicable for creating overview diagrams in hypertext structure.

## 7.3 General hypertext structures

For application to arbitrary hypertext structures, we have to consider organizational links as well as associative links since ACE is focused on the latter. There are two alternative approaches to handle both link types uniformly. One approach emphasizes associative links by compressing organizational links. That is, by packing each of the organizational parts as a grouping node, ACE can apply to general hypertext structures. If the number of organizational links is relatively small, this approach may be useful.

The other approach handles all of the relationships hierarchically, by eliminating associative links. Namely, by rearranging the information of associative links into a hierarchy, several efficient methods for tree-structured database can apply to general hypertext structures. Since ACE can be treated as a translation method of associative links into attribute values of nodes, i.e., cluster names, ACE will contribute highly to the rearrangement.

We prefer the hierarchy approach for two reasons. One is that placing the data into a hierarchy not only permits many efficient algorithms, but also provides meaning for an effective view facilitating human recognition. The other is that we can provide a set-oriented modeling for the data and combine hypertext with database facilities in a relatively high level. Our approach will contribute to improving the usability of information.

# 8 Conclusion

We have presented the method of Aggregation Clustering with Exceptions in order to capture high level relationships from a given hypertext structure. We have evaluated the physical database issues of this method by using a heuristic algorithm based on the Kernighan-Lin algorithm and applied it to some practical data. In order to specify the effective solution of our method, we also discussed the estimation of boundary conditions and an optimum solution to the problem using an analytic model.

Our results demonstrate the usefulness of our clustering methodology for both storage reduction of database size and page fetching. The potential exists for creating overview diagrams based on our method. ACE can provide an efficient navigating environment for large information spaces for both physical database design and human interface design.

## Acknowledgements

We would like to thank David Johnson for allowing us to use his standard graph examples, and also thank Jeff Ullman, Hector Garcia-Molina, Shaibal Roy, Pierangela Samarati, Mark Frisse and Marianne Siroker for their helpful comments.

# References

[AgBJ89] Agrawal, R., Borgida, A., and Jagadish, H. V. "Efficient Management of Transitive Relationships in Large Data and Knowledge Bases," *Proc. of the ACM SIGMOD'89,* 1989, pp. 253-262.

[Borg85] Borgida, A. "Language Features for Flexible Handling of Exceptions in Information Systems," *ACM Trans. on Database Systems,* Vol. 10, No. 4, 1985, pp. 565-603.

[Bush45] Bush, V. "As We May Think," *Atlantic Monthly 176,* 1945, pp. 101-110.

[CACM88] "Special Issue on Hypertext", *Communications of the ACM,* Vol. 31, No. 7, 1988.

[ClGa90] Clifton, C., and Garcia-Molina, H. "Indexing in a Hypertext Database," *Proc. of the 16th VLDB Conference,* 1990, pp. 36-49.

[CoBe88] Conklin, J., and Begeman, M. L. "gIBIS: A Hypertext Tool for Exploratory Policy Discussion," *ACM Trans on. Office Information Systems,* Vol. 6, No. 4, 1988, pp. 303-331.

[Conk87] Conklin, J. "Hypertext: An Introduction and Survey," *IEEE Computer,* Vol. 20, No. 9, 1987, pp. 17-41.

[CrNo89] Cruz, I. F., and Norvell, T. S. "Aggregative Closure: An Extension of Transitive Closure," *Proc. of the 5th Int. Conference on Data Engineering,* 1989, pp. 189-204.

[Fein88] Feiner, S. "Seeing the Forest for the Trees: Hierarchical Display of Hypertext Structure," *Proc. of ACM Conference of Office Information Systems,"* 1986, pp. 205-212.

[FrCo89] Frisse, M. E., and Cousins, S. B. "Information Retrieval from Hypertext: Update on the Dynamic Medical Handbook Project," *Proc. of the Hypertext'89,* 1989, pp. 199-212.

[Furn86] Furnas, G. W. "Generalized Fisheye views," *Proc. of ACM CHI'86,* 1986, pp. 16-23.

[GaJS76] Garey, M. R., Johnson, D. S., and Stockmeyer, L. "Some Simplified NP-Complete Graph Problems," *Theory of Computer Science,* Vol. 1, 1976, pp. 237-267.

[GaJo79] Garey, M. R., and Johnson, D. S. "Computers and Intractability: A Guide to the Theory of NP-Completeness," *W. H. Freeman and Company,* 1979.

[Good87] Goodman, D. "The Complete HyperCard Handbook," *Bantam,* 1987.

[HaKa88] Hara, Y., and Kaneko, A. "A New Multimedia Electronic Book and Its Functional Capabilities," *User-oriented, Content-based, Text and Image Handling (RIAO),* 1988, pp. 114-123.

[HaKa90] Hara, Y., and Kasahara Y. "A Set-to-Set Linking Strategy for Hypertext Systems," *ACM Conference on Office Information Systems,* 1990, pp. 131-135.

[HaKW91] Hara, Y., Keller, A. M., and Wiederhold, G. "Relationship Abstractions for an Effective Hypertext Design: Augmentation and Globalization," *DEXA'91,* 1991, pp. 270-274.

[Hype89] "Hypertext on Hypertext," *ACM Press Database and Electronic Products Series,* 1989.

[JAMS89] Johnson, D. S., Aragon, C. R., et al. "Optimization By Simulated Annealing: An Experimental Evaluation; Part I, Graph Partitioning," *Operations Research,* Vol. 37, No. 6, 1989, pp. 865-892.

[KeLi70] Kernighan, B. W., and Lin, S. "An efficient heuristic procedure for partitioning graphs," *Bell Systems J.,* Vol. 49, No. 2, 1970, pp. 291-307.

[MoBB90] Moline, J., Benigni, D., et al. (eds.) "Proceedings of the Hypertext Standardization Workshop," *National Institute of Standards and Technology,* 1990.

[Niel90] Nielsen, J. "Hypertext and Hypermedia," *Academic Press,* 1990.

[PaSt82] Papadimitriou, C. H., and Steiglitz, K. "Combinational Optimization: Algorithms and Complexity," *Prentice-Hall,* 1982.

[RoMN81] Robertson, C. K., McCracken, D., et al. "The ZOG Approach to Man-Machine Communication," *Int. J. of Man-Machine Studies,* Vol.14, 1981, pp. 461-488.

[SmSm77] Smith, J. M., and Smith, D. C. P. "Database abstractions: aggregation and generalization," *ACM Trans. on Database Systems,* Vol. 2, No. 2, 1977, pp. 105-133.

[Tomp89] Tompa, F.W. "A Data Model for Flexible Hypertext Database Systems," *ACM Trans. on Information Systems,* Vol. 7, No. 1, 1989, pp. 85-100.

[TsNa91] Tsangaris, M. M., and Naughton, J. F. "A Stochastic Approach for Clustering in Object Bases," *Proc. of the ACM SIGMOD'91,* 1991, pp.12-21.

[Trig88] Trigg, R. H. "Guided Tours and Tabletops: Tools for Communicating in a Hypertext Environment," *ACM Trans. on Office Information Systems,* Vol. 6, No. 4, 1988, pp. 398-414.

[Ullm84] Ullman, J. D. "Computational Aspects of VLSI," *Computer Science Press,* 1984.

[UtYa89] Utting, K., and Yankelovich, N. "Context and Orientation in Hypermedia Networks," *ACM Trans. on Information Systems,* Vol. 7, No. 1, 1989, pp. 58-84.

[WaSh90] Watters, C., and Shepherd M. A. "A Transient Hypergraph-Based Model for Data Access," *ACM Trans. on Information Systems,* Vol. 8, No. 2, 1990, pp.77-102.

[Wied87] Wiederhold, G. "File Organization for Database Design," *McGraw-Hill,* 1987.

[WoKi87] Woelk, D., and Kim, W. "Multimedia Information Management in an Object-Oriented Database System," *Proc. of the 13th VLDB Conference,* 1987, pp. 319-329.

[YHMD88] Yankelovich, N., Haan, B. J., et al. "Intermedia: The concept and the construction of a seamless information environment," *IEEE Computer,* Vol. 20, No. 1, 1988, pp. 81-96.