Implementing Hypertext Database Relationships through Aggregations and Exceptions

Yoshinori Hara **Arthur M. Keller** **Gio Wiederhold**

NEC Corporation and Stanford University
Advanced Decision Systems and Stanford University
Stanford University

Department of Computer Science
Stanford University
Stanford, CA 94305-2140, U.S.A.

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.

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. 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 [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. 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, technical manuals, or cooperative environment documents, 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 spatially capturing 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.

(2) Lack of design principles for associative links

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

However, the regularization of *associative links,* which are essential to information captured in hypertext as a network or a loop 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. 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]. 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, 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.

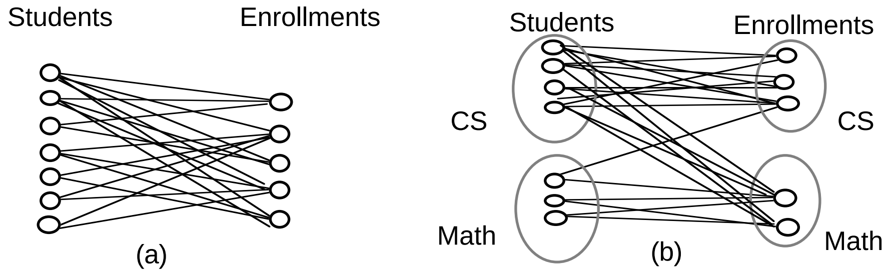


Figure 1. A Student-Enrollment Relationship

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.

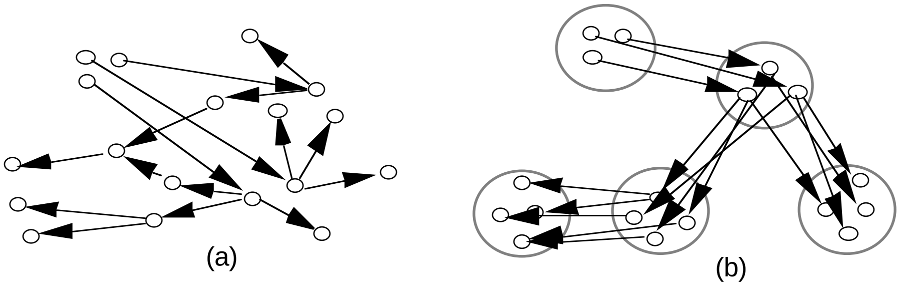


Figure 2. A Citation Hierarchy

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.

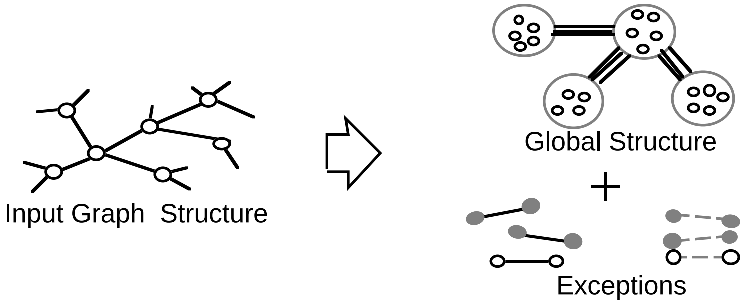
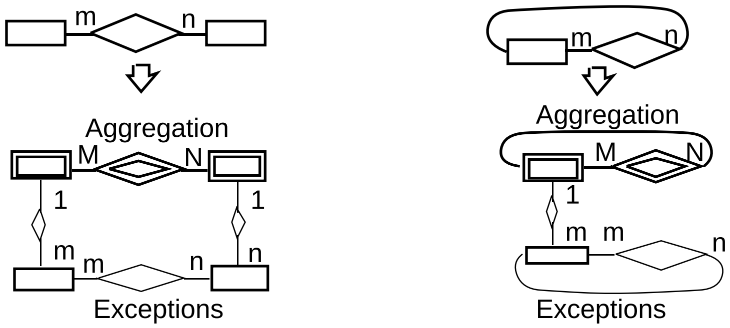


Figure 3. Aggregation Clustering with Exceptions

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.*



(a) M-to-M relationship aggregation (b) 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. 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 $V_A = \{V_{A_1}, ..., V_{A_s}\}$ is a partition of V. The set E_A contains the edges that connects the partition set 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 $G = (V, E)$
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:

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

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

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

As illustrated in Fig. 5, 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 components.

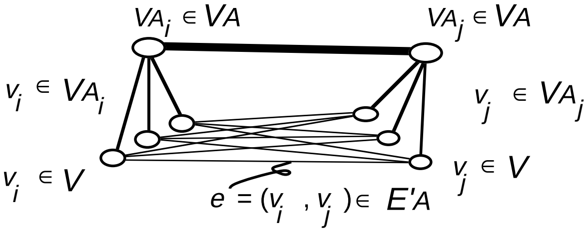
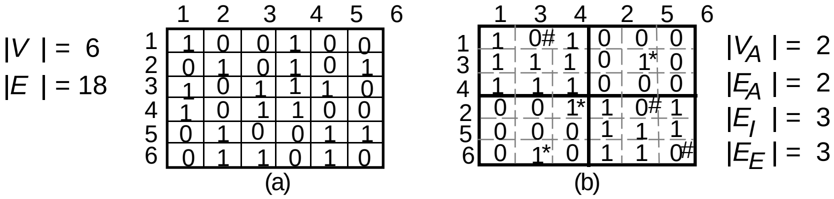


Figure 5. E'_A: A Set of Edges in the Complete Bipartite Graph

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

$$\text{Cost\_func}(G_A, G_I, G_E) = |V_A| + |E_A| + |E_I| + |E_E| \quad \rightarrow \quad \text{Min.} \tag{4}$$

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 the 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.



$|V| = 6$
$|E| = 18$

	1	2	3	4	5	6
1	1	0	0	1	0	0
2	0	1	0	1	0	1
3	1	0	1	1	1	0
4	1	0	1	1	0	0
5	0	1	0	0	1	1
6	0	1	1	0	1	0

(a)

	1	3	4	2	5	6
1	1	0#	1	0	0	0
3	1	1	1	0	1*	0
4	1	1	1	0	0	0
2	0	0	1*	1	0#	1
5	0	0	0	1	1	1
6	0	1*	0	1	1	0#

(b)

$|V_A| = 2$
$|E_A| = 2$
$|E_I| = 3$
$|E_E| = 3$

Figure 6. An Example of ACE

ACE is a clustering method of given relationships by allowing set difference and union operations in relational algebra. Since the clustering 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 [HKRW91]. 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.

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 fraction of the exclusive partitioned area is the same that of the inclusive partitioned area)
s ... the number of nodes (clusters) in the aggregation graph G_A, that is, $|V_A|$ (By assumption (1), we assume that each cluster size is the same)
r ... the number of links in the aggregation graph
α ... 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:

$$0 \leq q < p \leq 1 \tag{5}$$

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:

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

(6)

Therefore,

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

(7)

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,

$$S = n + r + qn^2$$
(8)

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 α^2, we can set out the following condition if the distribution of the links is a uniform distribution and n is relatively large:

$$q = p \cdot (1 - \alpha^2)$$
(9)

By applying equations (7), (9), and the storage function α, equation (8) can be written as follows:

$$\begin{aligned} S &= n + \frac{p\alpha^2}{1 - 2p(1 - \alpha)^2} \cdot n^2\alpha^2 + pn^2(1 - \alpha^2) \\ &= n + pn^2 \cdot \left(\frac{(1 - 2p)\alpha^4 - (1 - 4p)\alpha^2 + 1 - 2p}{2p\alpha^2 - 2p + 1} \right) \end{aligned}$$
(10)

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

$$\alpha^2_{opt} = \frac{\sqrt{1 - 2p} - (1 - 2p)}{2p}$$
(11)

That is, α_{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 $\sqrt{1 - 2p} \approx (1 - \mathrm{p})$, and $\alpha^2{}_{opt}$ is approximately 0.5. Also, $\alpha^2{}_{opt}$ 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 $\sqrt{1 - 2p} \approx (1 - \mathrm{p})$, the following optimum condition is ensured:

$$|E_I|_{opt} = \frac{1}{2} \cdot pn^2$$
(12)

$$|E_A|_{opt} + |E_E|_{opt} \approx \frac{1}{4} \cdot pn^2$$
(13)

Since the input database size is pn^2, the database size will be reduced by about 25% using ACE. Figure 7 shows the comparison of the database size of the output graph with that of the input graph according to α^2. Of course, the result of optimum solutions depends on the condition (9), the relationship between q and α. 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.

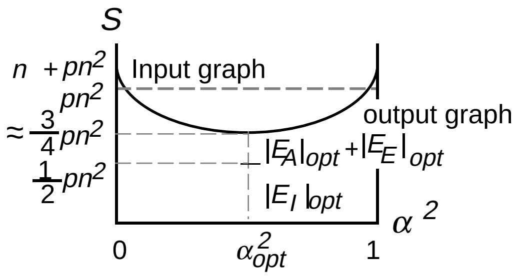


Figure 7. Comparison of the Database Size S with the Storage Function α

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 shown in Fig. 8, 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. In the usual case, there are no exceptions, so no other input-output operations are required. Otherwise, one additional input-output operation will obtain all the exceptions for this hyperedge. In Sections 6.1 and 6.2, we will discuss how infrequently this additional operation is required.

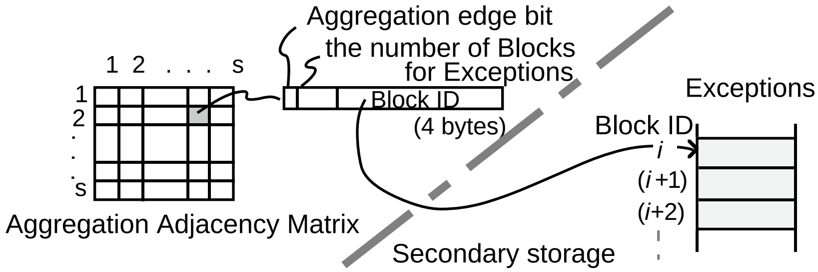


Figure 8. A Secondary Storage Organization for ACE

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: $k = 1$)

(Step 2) Repeated 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 $n \times k$ alternative cases
            end
            Find $l$ $(0 \leq l \leq n)$, $s.t. \sum \mathrm{Cost}_l \rightarrow \max$
            Perform the translations $(v_1, C_j^{(1)}), (v_2, C_j^{(2)}), ..., (v_l, C_j^{(l)})$; that is,
                move $v_i$ into cluster $C_j^{(i)}$, $(i = 1, 2, ..., l)$
            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 from a published testbed for graph algorithms.

6.1 Hypertext on Hypertext

"Hypertext on Hypertext" [Hype89] is one of the typical examples of hypertext structures. 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 × 26 × 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 × 313 / (512 × 8)) and there were two possible alternative: all blocks should be in main memory, or otherwise one input-output operation is always required.

	HT on HT	**Random Graph**
The number of nodes : n	313	124
The number of edges : e	8461	2542
The fraction of input links : p	8.6%	15.5% ($\tilde{p}$ = 16%)
Aggregation nodes : $\|VA\|$	26	58
Aggregation links : $\|EA\|$	138	254
Exclusive links : $\|EE\|$	34	110
Inclusive links : $\|EI\|$	1440	1742
Average fraction of exceptions : q	1.5%	12.0%
The fraction of non-exception areas	30.8%	40.4%
DB size $S = n + \|EA\| + \|EE\| + \|EI\|$	1925	2230
The reduction gain : 1 - (S / e)	77.2%	12.3%

Table 1. Experimental Results

Figure 9 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.

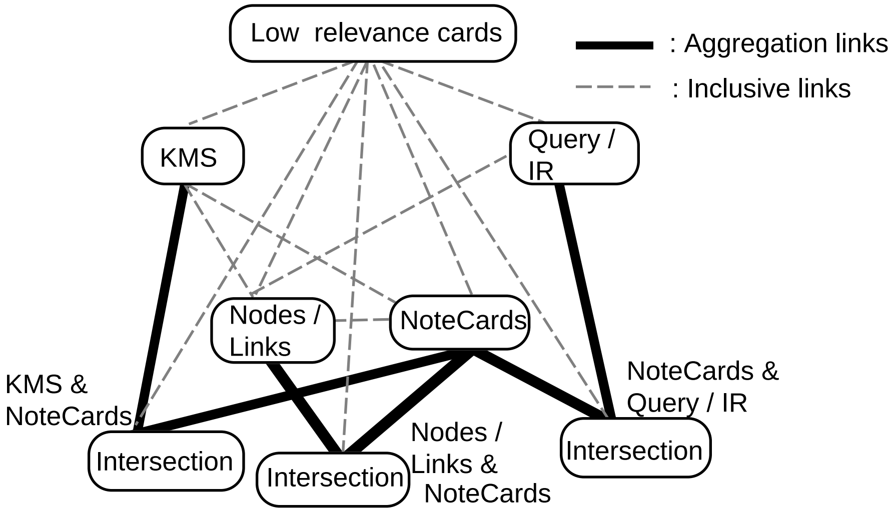


Figure 9. A Global Structure of "Hypertext on Hypertext" (Cluster size is more than five.)

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 Fig. 10. Thus, we see how effectively structures are compacted by ACE even for a random graph.

Figure 10. (a) The Input graph G (b) The Output $G_A + G_I$ (c) The Output G_A

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 input links and evaluate equation (9). There are two main causes for the difference between the analytic model and the experimental results. The first cause is that the distribution of each simulated graph was somewhat different from the assumption made for equation (9). 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 tuning issues for ACE and its extension into a more general hypertext/hypermedia structure.

7.1 Extension issues for the ACE model

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 relation to be equal to the number of input nodes, n, resulting in a simple output structure. In some special cases, however, a few extended methods provide more an elegant representation. One is to allow *overlapping* clusters, that is, to apply a many-to-many clustering relationship between clustered nodes and their elements. It provides a more compact aggregation when some input nodes correlate to nodes in both clusters. Another extension is to provide for *recursion*. That is, a sequence of aggregation graphs and of exception graphs is extracted by applying ACE recursively. It is useful when the same aggregated structure can be applied to multiple levels of aggregation in the recursive hierarchy.

Other extended issues such as combining with 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. 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 algorithm is based on a greedy algorithm. 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, it 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 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 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, Peter Rathmann, 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," **ACM SIGMOD'89**, 1989, pp. 253-262.

[CACM88] "Special Issue on Hypertext," **CACM**, Vol. 31, No. 7, 1988.

[ClGa90] Clifton, C., and Garcia-Molina, H. "Indexing in a Hypertext Database," **VLDB'91**, 1990, pp. 36-49.

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

[Fein88] Feiner, S. "Seeing the Forest for the Trees: Hierarchical Display of Hypertext Structure," **ACM Conf. on OIS**, 1986, pp. 205-212.

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

[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 Co.**, 1979.

[HaKa88] Hara, Y., and Kaneko, A. "A New Multimedia Electronic Book and Its Functional Capabilities," **RIAO'88**, 1988, pp. 114-123.

[HaKa90] Hara, Y., and Kasahara Y. "A Set-to-Set Linking Strategy for Hypertext Systems," **ACM Conf. on OIS,** 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** (to appear), 1991.

[HKRW91] Hara, Y., Keller, A. M., Rathmann, P. K., and Wiederhold, G. "Implementing Hypertext Database Relationships through Aggregations and Exceptions," **Stanford CS Technical Report** (to appear), 1991.

[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 Sys. J.**, Vol. 49, No. 2, 1970, pp. 291-307.

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

[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.

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

[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 IS**, 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 IS**, Vol. 8, No. 2, 1990, pp.77-102.

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