Response Time Driven Multimedia Data Objects Allocation for Browsing Documents in Distributed Environments

Siu-Kai So, Ishfaq Ahmad, *Member*, *IEEE*, and Kamalakar Karlapalem, *Member*, *IEEE*

Abstract—Distributed information processing, in many world wide web applications, requires access, transfer, and synchronization of large multimedia data objects (MDOs) (such as, audio, video, and images) across the communication network. Moreover, the end users have started expecting very fast response times and high quality of service. Since the transfer of large MDOs across the communication network contributes to the response time observed by the end users, the problem of allocating these MDOs so as to minimize the response time becomes very challenging. This problem becomes more complex in the context of hypermedia documents (web pages), wherein the MDOs need to be synchronized during presentation to the end users. Note that the basic problem of data allocation in distributed database environments is NP-complete. Therefore, there is a need to pursue and evaluate solutions based on heuristics which generate near-optimal MDO allocation. In this paper, we address this problem by: 1) conceptualizing this problem by using a navigational model to represent hypermedia documents and their access behavior from end users, and by capturing the synchronization requirements on MDOs, 2) formulating the problem by developing a base case cost model for response time, and generalizing it to incorporate user interaction and buffer memory constraints, 3) designing two algorithms to find near-optimal solutions for allocating MDOs of the hypermedia documents while adhering to the synchronization requirements, and 4) evaluating the trade-off between the time complexity to get the solution and the quality of solution by comparing the solutions generated by the algorithms with the optimal solutions generated through an exhaustive search.

Index Terms—Data allocation, response time, multimedia data objects, hypermedia documents, distributed hypermedia document systems, navigational model.

1 Introduction

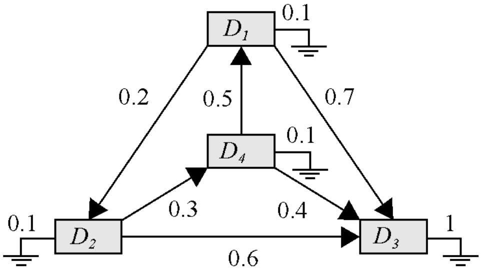
Distributed information processing has become the norm in recent years. Most of the Internet driven web based information access requires distributed processing. In many applications, this processing typically requires, access, transfer and synchronization of multimedia data objects (MDOs) (such as, audio, video, and images) [1], [5]. The quality of services provided in presenting these MDOs to end users has become an issue of paramount importance. End users have started expecting strict adherence to synchronization and response time constraints. Any application or system which cannot respond quickly and in a timely manner for presenting MDOs to end users is at a clear disadvantage. In order to manage and present large number of hypermedia documents and their MDOs distributed hypermedia database systems have been proposed [19]. In fact, a set of web servers can be treated as a distributed hypermedia database system. Therefore, the solutions and the approaches developed in this paper can also be applied in designing efficient web servers in intranet environments (wherein the organization has a complete control in placing the web pages at different internal web servers).

Distributed database systems have introduced a number of problems, such as the data fragmentation and data allocation problems, that do not exist in centralized database systems [12]. A good data allocation scheme is always highly desirable, even in single-media distributed database systems, since it can significantly reduce the response time of database queries. Due to the large variations in the sizes of *MDO*s such a data allocation scheme is even more urgently needed for distributed hypermedia database systems. These systems also cater to high performance applications wherein a set of end users can access multiple hypermedia documents in any order and expect good response time and quality of service. Since the hypermedia documents may not be located at the end users' sites, they need to be transferred across the communication network incurring delays (increasing response time) in presenting the MDOs of the hypermedia documents. Since end users at different sites may access the same hypermedia documents, the problem of hypermedia data allocation gets further complicated. Hence the allocation of the hypermedia documents and their MDOs govern the response time for the end users. Further, since the MDOs in a hypermedia document need to be synchronized, the allocation should also adhere to these synchronization constraints.

We model a hypermedia document as a directed graph with each node representing a hypermedia document with its MDOs, and each out going directed edge as a hyperlink to another hypermedia document. Because of the vagaries of the communication network, the MDOs are presented to

• *The authors are with the Department of Computer Science, Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong. E-mail: iahmad@cs.ust.hk.*

Manuscript received 15 November 1997; revised 7 July 1998.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number 107202.



$$Nav = \begin{array}{cc} & \begin{array}{cccc} D_1 & D_2 & D_3 & D_4 \end{array} \\ \begin{array}{c} D_1 \\ D_2 \\ D_3 \\ D_4 \end{array} & \begin{bmatrix} 0 & 0.2 & 0.7 & 0 \\ 0 & 0 & 0.6 & 0.3 \\ 0 & 0 & 0 & 0 \\ 0.5 & 0 & 0.4 & 0 \end{bmatrix} \end{array}$$

Fig. 1. Probability model of navigational links between hypermedia documents.

the end users only after they are buffered at the end-users' site. Therefore, the synchronization requirements may impose additional delays for presenting the MDOs to the end users [13], [26]. We also take into consideration end-user behavior in accessing the hypermedia documents from various sites, and the frequencies with which they access these hypermedia documents. We then develop a cost model which takes into consideration the synchronization constraints for calculating the response time for the end users for a given allocation scheme. We present two algorithms for finding the near-optimal data allocation of hypermedia documents. Subsequently, we illustrate our model and approach by a real-life example and evaluate the goodness of the proposed algorithms in terms solution quality (by comparing with the optimal solution) and time complexity in achieving this solution.

We design and evaluate data allocation algorithms so as to optimize the response time for a set of end users while adhering to the synchronization requirements of the MDOs presentation in distributed hypermedia database systems. In Section 2, we introduce different modeling specifications of multimedia documents. In the same section we propose a graphical notion to represent navigation in the hypermedia systems. In Section 3, we develop a cost model for the data allocation problem of distributed hypermedia systems. This cost model is used to evaluate an example hypermedia database system. In Section 4, we describe the proposed algorithms based on Hill-Climbing and probabilistic neighborhood search approaches. In Section 5, we include the experimental results. In Section 6, we provide an overview of existing related work as well as other issues related to our problem. Section 7 presents conclusions and possible extensions to this work.

2 Modeling Hypermedia Documents

A hypermedia document is a directed graph $DG(H, E)$ where $H = \{D_1, D_2, ..., D_n\}$ is the set of vertices, each D_p representing a hypermedia document, and each directed edge from D_p to $D_{p'}$ is a link denoting access of document $D_{p'}$ from document D_p. Therefore, a user can start browsing the documents from (say) document D_p and then proceed to access document $D_{p'}$ etc. Further, each hypermedia document has a set of MDOs which need to be presented to the end users accessing this document. Since the end users can access the hypermedia documents in any order and browse through them, we have a label attached to each directed edge from D_p to $D_{p'}$ giving the probability of end users accessing document $D_{p'}$ from document D_p. These probabilities can be generated by gathering statistics (about document access, and browsing through logs of users browsing activity) about end-user behavior over a period of time. Further, since a user may end browsing after accessing any hypermedia document, the probabilities of outgoing edges from a vertex do not add up to 1.0, and the difference is the probability of ending the browsing at document D_p, and is shown by an edge connecting to the ground (see Fig. 1). An $n \times n$ matrix Nav is used to capture this information.

Example 1. Suppose we have four hypermedia documents D_1-D_4, Fig. 1 shows the links between these documents and the probabilities of access from one document to another. Further, we also show the probability of ending the browsing session at each hypermedia document. For instance, there is a probability of 0.1 that browsing ends after accessing document D_1. The corresponding Nav matrix is shown on the right-hand side (of Fig. 1).

From the above navigational model, we can calculate the cumulative long run probabilities of accessing a hypermedia document $D_{p'}$ from document D_p. This is done by considering all possible paths from document D_p to $D_{p'}$, and calculating the probability of accessing $D_{p'}$ from document D_p for each path, and taking the maximum of all these probabilities. Note that we assume each document access and browsing from one document to another to be independent events. Therefore, for a path with t edges from document D_p to document $D_{p'}$, the probability of this path is the product of t probabilities for the edges. Since there can be potentially infinitely long paths, we limit the length of the path by limiting the value of the cumulative probability given by the path to be greater than a parameter value (bpl). Let R be the $n \times n$ matrix, with each element $r_{pp'}$ giving the cumulative long run probability of accessing document $D_{p'}$ from document D_p.

Example 1 (continued). From the navigational model, we can construct a tree for each document representing the possible navigation path for each session starting from that document. These are given in Fig. 2. We set the bpl value to be 0.01. Notice that we do not need to further expand a node if the document represented by that node is the same as that of the root. (This happens in the first tree in Fig. 2). Therefore, if we start navigating the hypermedia system from document D_1, we have probability 0.2 that we browse

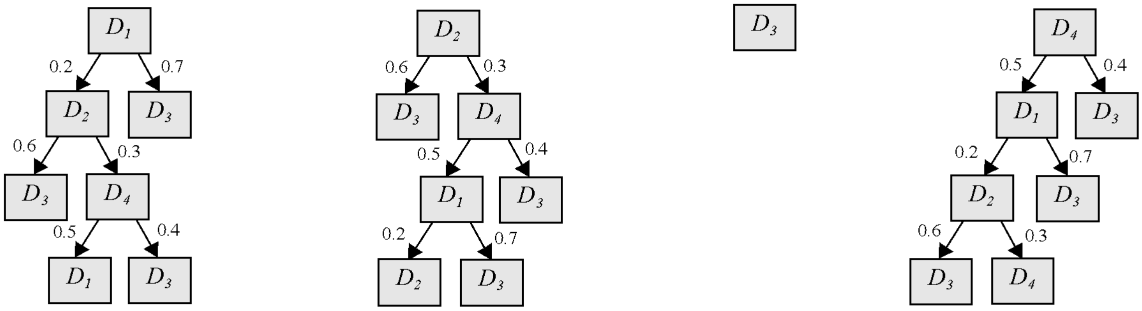


Fig. 2. Navigation paths starting from each hypermedia document (*bpl* is set as 0.01).

document D_2. For document D_3, if we follow the right path from D_1, the probability is equal to 0.7. But if we follow the path $D_1 \rightarrow D_2 \rightarrow D_3$, the probability is equal to $0.2 \times 0.6 = 0.12$. In this case, we use the greater probability to represent the long run probability of browsing D_3 from D_1 as 0.7.

Similarly other cumulative probability values are calculated. Therefore, the matrix R is

$$R = \begin{array}{c} \\ D_1 \\ D_2 \\ D_3 \\ D_4 \end{array} \begin{array}{c} \begin{array}{cccc} D_1 & D_2 & D_3 & D_4 \end{array} \\ \begin{bmatrix} 1 & 0.20 & 0.70 & 0.06 \\ 0.15 & 1 & 0.60 & 0.30 \\ 0 & 0 & 1 & 0 \\ 0.50 & 0.10 & 0.40 & 1 \end{bmatrix} \end{array}.$$

We use the *O*bject *C*omposition *P*etri *N*ets (OCPNs) [13] for modeling the synchronization constraints among the MDOs in a hypermedia document. Petri nets are powerful tools for representing objects that must be synchronized. In addition, they have the advantage of generating database schema as well as extracting spatio-temporal and content semantics [5]. In contrast to *HyTime* [6], Petri nets use a graphical notation for representing synchronization constraints. The hyperlinks are associated with the transitions [21] of the Petri net. With this mechanism, we can easily maintain a hyperlink by storing the address of the destination document in the database [10] and we will no longer be concerned about invalid links when the document addresses are changed. Further, *OCPN* simplifies the Petri nets by restricting the number of outgoing edges from each transition to two and enhances them by introducing the duration and the addressing scheme for each place. This enhancement makes *OCPN* suitable for modeling synchronization constrains among MDOs of hypermedia documents. We can transverse a transition (called as firing) if all places pointing to this transition have a token and are in an unlocked state. When the transition fires, the places that the transition is pointing to will become active (a token is added to these places) and locked. Places will become unlocked when their durations have expired. All *OCPN* models can be mapped to a corresponding *HyTime* model [3]. In Fig. 3, the following synchronization constraint is represented: MDO *A* has to be shown exactly 10 time units after the start of the presentation of the hypermedia document, and after another 30 time units MDO *B* must be shown.

In [13] the multimedia specification associates time with a place. However, in traditional timed-Petri net, time is associated with a transition [15]. The reason for associating time with a place is for compactness. By using the traditional method, we can model user interaction [21] during hypermedia document presentation. In Fig. 4 the presentation of MDO *D*, represented as a box, will continue as long as both of the two places pointing to it have tokens. The user can interrupt the presentation by pressing the button associated with the immediate transition, represented in Fig. 4 by a bold vertical line. This user interaction will fire the upper transition, removing the token in the middle place on the left.

EXAMPLE 1 (continued). The *OCPN* synchronization specifications of hypermedia documents D_1 to D_4 are shown in Fig. 5.

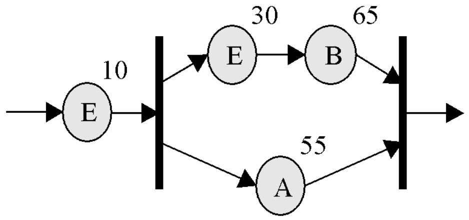


Fig. 3. The *OCPN* model of synchronizing MDOs A and B, where E represents some delay event.

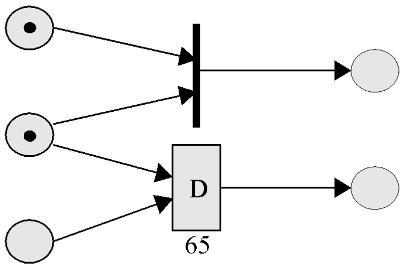


Fig. 4. Traditional timed-Petri net modeling of user interaction of presentation.

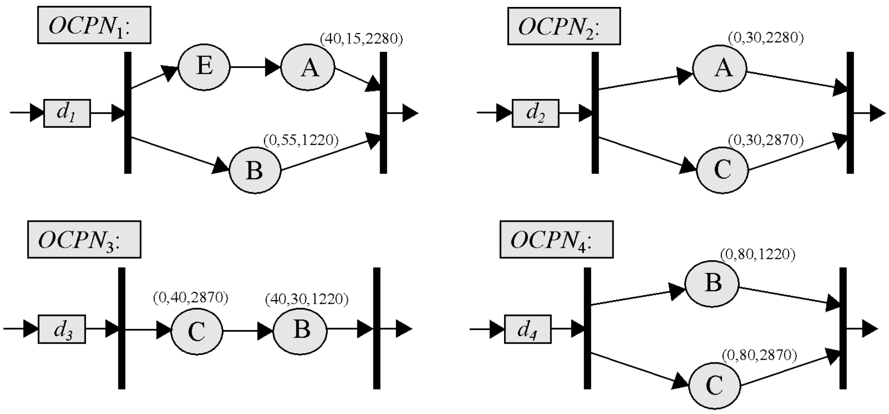


Fig. 5. The *OCPN* specification of each hypermedia document; the tuple is [start time, duration, media size in kilobytes].

3 Cost Model for Data Allocation Scheme

In order to reduce response time for the end users browsing activities, we need to develop a cost model for calculating the total response time observed. This response time depends on the location of the MDOs and the location of the end user. Further, it depends on the synchronization constraints among the MDOs of the hypermedia document browsed. The hypermedia document navigational model presented in Section 2 is used to estimate the number of accesses (times browsed) to each MDO from each site. This gives us the information regarding the affinity between the MDOs and the sites of the distributed environment. Typically, one would assign an MDO to a site which accesses it the most. But this may incur large delay for other sites that also need to access this MDO. Further, synchronization constraints may impose additional delays in transferring the MDO to the end-user site. This is done when two streams of MDOs need to simultaneously finish their presentation, and one of them is for shorter duration than the other. Since we buffer the MDOs at the user sites before the start of the presentation, the MDO allocation problem needs to minimize the additional delay incurred because of the synchronization constraints. We also take into consideration limited buffer space constraint at end user's site and user interaction during MDO presentation.

Table 1 lists a number of notations used throughout this paper.

3.1 Total Response Time Cost Function

Suppose there are m sites in the distributed hypermedia database system. Let S_i be the name of site i where $1 \le i \le m$. These m sites are connected by a communication network. A communication link between two sites S_i and $S_{i'}$ will have a positive integer $c_{ii'}$ associated with it giving the transmission speed from site i to site i'. Note that these values depend on the routing scheme of the network. If fixed routing is used, we can get the exact values. However, if dynamic routing is used, we can only obtain the expected values statistically. Let there be j hypermedia documents, called $\{D_1, D_2, ..., D_j\}$ accessing k *MDO*s, named $\{O_1, O_2, ..., O_k\}$.

Example 1 (continued). Assume that the hypermedia database system for storing the *MDO*s is distributed in a network with three sites. The transmission speeds between the three sites can be represented as an $m \times m$ matrix C, with entry $c_{ii'}$ representing the transmission speed from S_i to $S_{i'}$.

$$C = \begin{array}{c} \\ S_1 \\ S_2 \\ S_3 \end{array} \begin{array}{c} \begin{array}{ccc} S_1 & S_2 & S_3 \end{array} \\ \begin{bmatrix} 0 & 38 & 41 \\ 38 & 0 & 35 \\ 41 & 35 & 0 \end{bmatrix} \end{array}.$$

As explained above, from the navigation model, we can construct j trees representing the navigation path of the session starting from each document. Since the height of these trees will typically be infinite, we must limit the level we will use for our cost model. We limit the height of the trees such that the cumulative probability of each path is greater than a threshold value *bpl*, say 0.001. These trees will give us cumulative long run probability $r_{jj'}$ of retrieving the document $D_{j'}$ if we start navigating from the document D_j.

For each site, we use an irreducible continuous-timed Markov process [23] to model the user behavior across browsing sessions as a stationary regular transition probability matrix, P^i, $1 \le i \le m$. These processes will converge in the long run and from these long run behaviors, we can estimate the probability of using each document as starting point for each browser session initiated in each site. These Markov chains will have $n + 1$ states representing the probabilities of using each of the documents as the starting point for browsing session (n, states), and probability of not browsing any of the documents ($(n + 1)th$ state, shown by row/column E in matrices P^i below). After analyzing the long run behavior of the Markov chain at each site, we will have the probabilities of using each document as initial browsing document and of not browsing at each site. As there is no delay when the user does not browse, we can eliminate the probability of not browsing. If we normalize the probabilities derived from long run behavior of Markov chain at each site and multiply them by a constant vector F (number of accesses to documents at each site), we get the expected frequencies of initiating browsing at each

TABLE 1
SYMBOLS AND THEIR MEANINGS

Symbol	Meaning
S_i	The ith site
D_j	The jth hypermedia document
O_k	The kth *MDO*
m	The number of sites in the network
n	The number of hypermedia documents in the database system
k	The number of *MDO*s in the database system
B	The navigation initial document frequencies matrix
b_{ij}	The frequency of using the jth document as initial point at the ith site
C	The transmission speed matrix of the network
$c_{ii'}$	The transmission speed from site i to site i'
P^i	The user navigation probability matrix of site i
$p^i_{jj'}$	The probability matrix modeling the user behavior at site i between browsing sessions. This value gives the probability of using document j' as initial browsing document in the next session if the initial browsing document is j in the current navigation session
A	The access frequencies matrix
a_{ij}	The access frequency of document j from site i
l	The allocation limit vector of the sites
l_i	The allocation limit of site i
R	The hypermedia document dependency matrix
$r_{jj'}$	The probability of retrieving document j' if browsing initial document is j
$OCPN_j$	The *OCPN* specification of document j
U	The usage matrix
u_{jk}	The boolean value of whether document j uses *MDO* k
dur_{jk}	The presentation duration of *MDO* k in document j
$start_{jk}$	The presentation starting time of *MDO* k in document j
$size_k$	The size of the kth *MDO*
bpl	The browsing probability limit
et_j	The expected number of times document j will be retrieved
D	The delay matrix
d_{ij}	The delay of presentation starting time of document j at site i
F	The access frequency vector
Nav	The navigation probability between documents within each browsing session
t	The total delay

document from each site in unit time. The resultant information is represented by an $m \times n$ matrix B.

We multiply this matrix to the $n \times n$ matrix R obtained from the hypermedia document trees to generate $m \times n$ matrix A with entries a_{ij} giving the expected number of times S_i needs to retrieve the *MDO*s in D_j.

EXAMPLE 1 (continued). Suppose the user navigation probabilities in the three sites are,

$$P^1 = \begin{array}{c} \\ D_1 \\ D_2 \\ D_3 \\ D_4 \\ E \end{array} \begin{array}{c} \begin{array}{ccccc} D_1 & D_2 & D_3 & D_4 & E \end{array} \\ \begin{bmatrix} 0.2 & 0.1 & 0.1 & 0.3 & 0.3 \\ 0.1 & 0.6 & 0.2 & 0.1 & 0 \\ 0 & 0.2 & 0.6 & 0.1 & 0.1 \\ 0.1 & 0.2 & 0.2 & 0.4 & 0.1 \\ 0.3 & 0.1 & 0.1 & 0.3 & 0.2 \end{bmatrix} \end{array},$$

$$P^2 = \begin{array}{c} \\ D_1 \\ D_2 \\ D_3 \\ D_4 \\ E \end{array} \begin{array}{c} \begin{array}{ccccc} D_1 & D_2 & D_3 & D_4 & E \end{array} \\ \begin{bmatrix} 0.2 & 0.4 & 0.2 & 0 & 0.2 \\ 0.1 & 0.6 & 0.1 & 0 & 0.2 \\ 0.4 & 0.2 & 0.2 & 0 & 0.2 \\ 0.3 & 0.3 & 0.3 & 0 & 0.1 \\ 0.2 & 0.2 & 0.4 & 0 & 0.2 \end{bmatrix} \end{array},$$

$$P^3 = \begin{array}{c} \\ D_1 \\ D_2 \\ D_3 \\ D_4 \\ E \end{array} \begin{array}{c} \begin{array}{ccccc} D_1 & D_2 & D_3 & D_4 & E \end{array} \\ \begin{bmatrix} 0.5 & 0 & 0 & 0.4 & 0.1 \\ 0.2 & 0.3 & 0.2 & 0.1 & 0.2 \\ 0.2 & 0.2 & 0.3 & 0.1 & 0.2 \\ 0.2 & 0.1 & 0.1 & 0.6 & 0 \\ 0.3 & 0.1 & 0.1 & 0.2 & 0.3 \end{bmatrix} \end{array}.$$

After the analyses of the long run behavior of these Markov chains, the expected starting document frequencies out of $F = \{900, 800, 900\}$ browsing sessions, matrix B is,

$$B = \begin{array}{c} \\ S_1 \\ S_2 \\ S_3 \end{array}\begin{array}{c} \begin{array}{cccc} D_1 & D_2 & D_3 & D_4 \end{array} \\ \begin{bmatrix} 100 & 300 & 300 & 200 \\ 200 & 400 & 200 & 0 \\ 300 & 100 & 100 & 400 \end{bmatrix} \end{array}.$$

Then the matrix A ($B \times R$) is,

$$A = \begin{array}{c} \\ S_1 \\ S_2 \\ S_3 \end{array}\begin{array}{c} \begin{array}{cccc} D_1 & D_2 & D_3 & D_4 \end{array} \\ \begin{bmatrix} 245 & 340 & 630 & 296 \\ 260 & 440 & 580 & 132 \\ 515 & 200 & 530 & 448 \end{bmatrix} \end{array}.$$

Further, we need the size, starting time, duration, and presentation rate (synchronization constraint specification) of each *MDO* in each hypermedia document. For the last three items, we only need any two of them; the remaining one can be derived from the other two. This information can be obtained from the OCPN specification of MDOs in a hypermedia document.

A box is added at the beginning of each *OCPN* which represents the delay in starting the presentation of the hypermedia document so as to adhere to the synchronization requirements. The duration of this delay box is related to the browsing site and the sites where the *MDO*s in the document are allocated. Thus, we use d_{ij} to represent the duration of the delay box when site S_i accesses the document D_j.

An *OCPN* representation provides the starting time, $start_{jk}$, and duration, dur_{jk}, of each O_k in each document D_j. In addition, the $n \times l$ usage matrix U is generated from the *OCPN* specifications (if document D_j uses MDO O_k, then $u_{jk} = 1$, otherwise, $u_{jk} = 0$). Then, by multiplying A by U, we can estimate the access frequencies of each *MDO* from each site. Let $size_k$ be the size of *MDO* O_k.

With this information, we can calculate d_{ij} as follows,

$$d_{ij} = max_{\forall k, u_{jk}=1}\left(\frac{size_k}{c_{site(k)\cdot i}} - dur_{jk} - start_{jk}\right) \tag{3.1}$$

where $site(k)$ represents the site where O_k is allocated.

We can calculate the values of all d_{ij}, $1 \le i \le m$, $1 \le j \le n$, by using the above formula. This formula means that the delay is equal to the maximum value of (transmission duration - presentation duration - presentation starting time) for each *MDO* in the document. When this value is negative, it implies that the transmission time is shorter than the presentation time, as we can start presenting the MDOs in the hypermedia document as soon as the MDOs arrive at the end-user site. When this value is positive, we must delay the presentation, otherwise the MDOs presentation cannot end at the synchronization time, and hence will not adhere to the synchronization constraints.

For example, suppose we need to present the document with OCPN specification shown in Fig. 6. If the MDO A is not in the site where this document is presented, we need to retrieve it from the network. If the transmission duration for the whole MDO A is greater than 30, we need to introduce some additional delay for fulfilling the synchronization requirement (to avoid jitter).

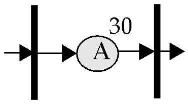


Fig. 6. *OCPN* representation of a simple MDO.

Therefore, the objective function is,

$$t = \sum_{i \in S}\sum_{j \in D} d_{ij} \cdot a_{ij} \tag{3.2}$$

By minimizing this value through the change of the function $site(k)$, we obtain the data allocation scheme that is optimal (the (delay incurred) response time is minimal), while adhering to the synchronization constraints.

EXAMPLE 1 (continued). There are three *MDO*s, namely, A, B and C (E is a delay state, so there is no associative actual *MDO*). If we allocate A at Site 2, B at Site 3, and C at site 1, then d_{11} is equal to,

$$d_{11} = max\left(\left(\frac{2{,}280}{38} - 15 - 40\right), \left(\frac{1{,}220}{41} - 55 - 0\right)\right) = 5.$$

Similarly, we can calculate the values of all d_{ij}, $1 \le i \le 3$, $1 \le j \le 3$, as we have the *MDO* allocation scheme. And the total response time delay will be,

$$11430 + 40650 + 29130 = 81210.$$

3.2 User Interactions and Buffer Space Constraints

The model presented above does not consider user interactions and buffer space constraints. It assumes that the user does not interrupt the presentation and the size of the local storage facility is large enough for storing any one of the hypermedia documents in the database system. The original *OCPN* model does not incorporate user interactions. As stated in Section 2, we can model these kinds of user interactions by using the traditional timed-Petri net. However, as there are many different kinds of interactions, this method alone is insufficient. The model must include the semantics for user interactions as well. There are a number of extensions to the Petri net that include these semantics [17], [20], [25]. However, in our paper, we only need to be concerned with the probabilities of each user interaction; therefore, we will not introduce these models here. Interested readers can refer to these papers for details. By including user interactions and buffer space constraints, there can be four different cases for hypermedia document allocation problem given below.

3.2.1 *No User Interaction and Unlimited Buffer Space*

This is essentially the best scenario, because we can retrieve all *MDO*s in a hypermedia document at the beginning since there is no storage limitation. As there is no user interaction, the data can be discarded immediately after use. The cost function for the response time for each hypermedia document, as presented in Section 3.1, is the maximum of the delays of the embedded *MDO*s for satisfying the synchronization requirements.

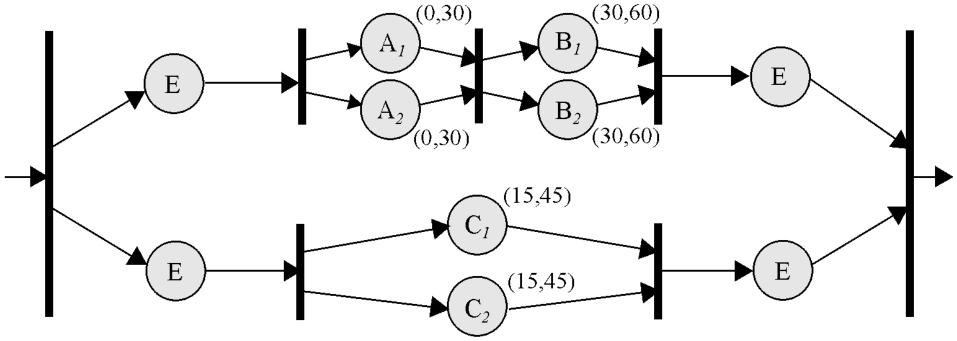


Fig. 7. The OCPN specification of synchronization requirements for document D_5.

3.2.2 User Interaction and Unlimited Buffer Space

By including user interactions, some of the *MDO*s in a hypermedia document may need to be presented multiple times (e.g., play in reverse or stop and resume later). However, as there is unlimited buffer space, the system can store all *MDO*s of a hypermedia document once they are retrieved. Therefore, the delay for handling the user interactions is due to some local processing that is irrelevant to the data allocation of the *MDO*s. The cost function is thus the same as that in the Section 3.1.

3.2.3 No User Interaction and Limited Buffer Space

In this scenario, the system cannot retrieve all the *MDO*s in advance. Instead, the system must retrieve the *MDO*s only when it needs to present these *MDO*s. Therefore, every synchronization point in the hypermedia document may cause some delay and the cost function in such a situation is the summation of these delays. Indeed, the model presented in Section 3.1 can be generalized to deal with this scenario.

First, we need to decompose each document into component subdocuments. From the *OCPN* specifications, we know the states representing the *MDO*s in each document. Denoting this set of states as S and for $\forall s, s \in S$, we can get the starting time and ending time of the state (i.e., presentation of the corresponding *MDO*) from the *OCPN* specifications. Then, we can use the algorithm DECOMPOSE_DOCUMENT(S) to decompose the document into its subdocuments. The algorithm is outlined on the next page.

EXAMPLE 2. Suppose we have a hypermedia document D_5 shown in Fig. 7. Then, we can decompose it into three subdocuments (at the synchronization points) as shown in Fig. 8.

Denote these subdocuments as D_A, D_B, and D_C. Calculating the delays of these subdocuments and summing them up will give the total delay of the original document D_1. The remaining problems of replacing D_1 by D_A, D_B, and D_C are the corresponding updates of the two matrices R and B. For the matrix R, as D_A is the starting subdocument of D_1, we replace the label of D_1 by D_A. After that, we add two columns and two rows for D_B and D_C. If we browse to D_A, both D_B and D_C will be retrieved. Therefore, set the values r_{AB} and r_{AC} to 1. Besides, set r_{BB} and r_{CC} to 1. Set the remaining values that are not yet specified to 0. This is because we will not need other documents if we are in D_B or D_C and we will not need to retrieve D_B or D_C in other documents.

The matrix B represents the probability of using each document as the initial browsing document. Thus, we only need to change the label of D_1 to D_A (the starting subdocument) and add two columns representing D_B and D_C. In these two columns, set all the entries to 0, as we cannot start browsing from D_B or D_C. Thus, by decomposing each hypermedia document at its synchronization points, the model presented in Section 3.1 incorporates the constraints of limited storage.

3.2.4 User Interaction and Limited Buffer Space

If we know the expected number of times each subdocument will be presented in each hypermedia document, we can calculate the expected response time of each document in each site, which is just the weighted sum of the delays of the subdocuments in the document. In the previous scenario, the expected number of times each subdocument is needed is 1, so the cost function is just the summation of the delays. After employing the algorithm DECOMPOSE_DOCUMENT(S) to decompose documents, we do not need to distinguish documents or subdocuments and we will use documents to represent both from now on.

To calculate the expected number of times each document is needed, we must know the probabilities of relevant user interactions. Relevant means that the interaction will need to retrieve the *MDO*s in the document again such as reverse playing or stop and resume. Other interactions that will not need the retrieval of the *MDO* again (for instance, termination of browsing) can be ignored. Once we have these probabilities, we can calculate the expected number of times each document is presented by employing the first step analysis method [23]. Note that these probabilities can be generated by observing user interaction over a period of time. Alternatively, we can use Markov chain to model the inter-*MDO* user interaction and obtain the required probabilities from long run behavior analysis.

For example, suppose the relevant probability of an *MDO* O_k in a document D_j is ip_{jk}. Assume that the expected

```
DECOMPOSE_DOCUMENT(S)
begin
   Construct state starting time list SSL
   Construct state ending time list SEL
   Construct transition time list TTL = SSL ∪ SEL
   Sort TTL and eliminate duplicate items from it
   Initiate the resultant document set RDS as empty set
   Initiate the current document set CDS as empty set
   counter = current_et = 1
   while TTL is not empty
      Remove the first item from TTL and store it in current_time
      if current_time ∈ SEL then
         for each s ∈ S with current_time as ending time
            for each document d ∈ CDS
               if s ∈ d then
                  Remove s from d
                  if d is empty then
                  Remove d from CDS
                  end if
               end if
            end for
         end for
      end if
      if current_time ∈ SSL then
         Combine ∀_s ∈ S with current_time as starting time into a
         new document D
         D.name = counter
         Calculate the value of et_counter
         if et_counter > current_et then
            current_et = et_counter
            for ∀d ∈ CDS
               et_d.name = current_et
            end for
         else
            et_counter = current_et
         end if
         Insert D into CDS and RDS
         Increment counter by 1
      end if
   end while
   return RDS
end.
```

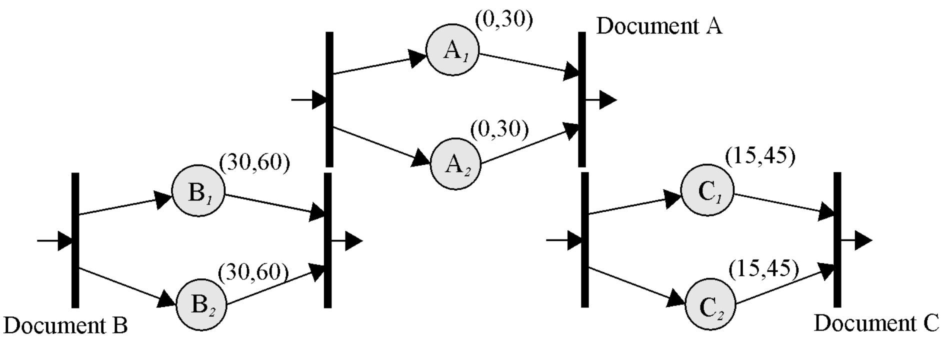


Fig. 8. Subdocuments OCPN specification of synchronization requirements for document D_5.

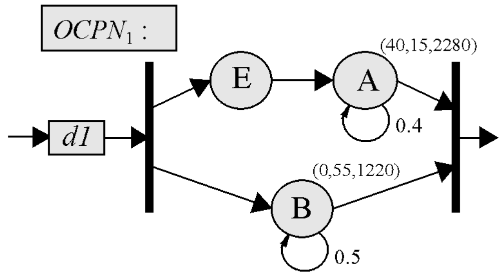


Fig. 9. The augmented OCPN by including user-interaction.

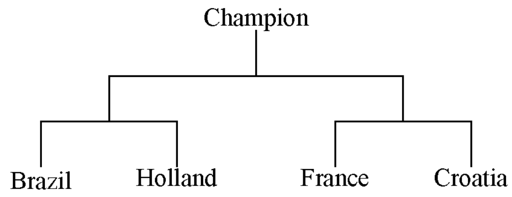


Fig. 10. The scenario for 1998 World Cup semifinal round.

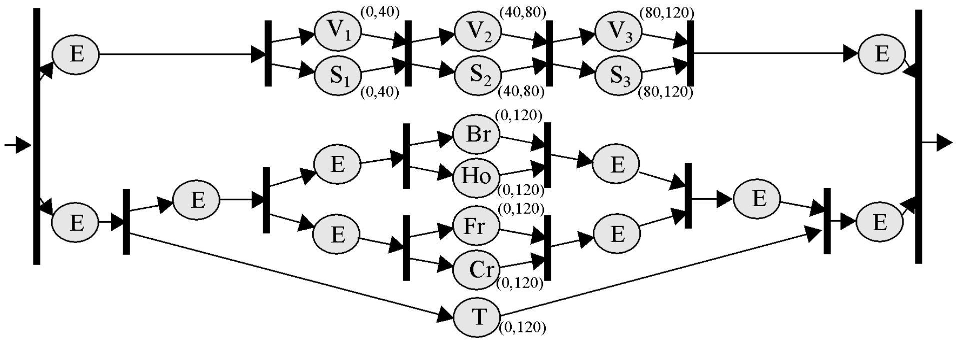


Fig. 11. The 1998 World Cup at semifinal round (D_1).

number of times this *MDO* is needed is mdo_et_{jk}. Then, we have,[1]

$$mdo\_et_{jk} = 1 + ip_{jk} \times mdo\_et_{jk},$$
$$mdo\_et_{jk} = (1 - ip_{jk})^{-1}. \quad (3.3)$$

Similarly, we can estimate the expected number of times other *MDO*s composing this document are needed. Then, the expected number of times this document is needed is just the maximum of these values. Denoting this value as et_j for document D_j, we have,

$$et_j = max(mdo\_et_{jk}), \text{ for } \forall k, u_{jk} = 1. \quad (3.4)$$

And the delays d_{ij} will become,

$$d_{ij} = \left(max\left(\frac{size_k}{C_{site(k) \cdot i}} - dur_{jk} - start_{jk} \right) \right) \cdot et_j, \quad (3.5)$$
$$for \ \forall k, \ u_{jk} = 1$$

We have made an assumption here that we can retrieve the *MDO*s from anywhere in the middle of it. If this assumption is violated, we can only get the whole *MDO* again even if the user just wants the last part of it. Otherwise, the assumption will introduce some overhead. With uniform distribution, the overhead will be half of the *MDO* presentation duration. That is,

$$overhead_{ij} = \frac{max(dur_{jk} + start_{jk}) \cdot (et_j - 1)}{2}. \quad (3.6)$$

And the cost function will become,

$$t = \sum_{i \in S} \sum_{j \in D} (d_{ij} + overhead_{ij}) \cdot a_{ij}. \quad (3.7)$$

Suppose we add user interactions and worst case buffer space constraints to the hypermedia database system in Example 1. After adding the probability of relevant user interruption to the *MDO*, the augmented *OCPN* of D_1 is shown in Fig. 9.

Thus, the expected number of times *MDO A* is needed when document D_1 is retrieved is given by:

$$mdo\_et_{1A} = 1 + 0.4 \times mdo\_et_{1A},$$
$$mdo\_et_{1A} = \frac{1}{1 - 0.4} = 1.667.$$

Similarly, the expected number of times *MDO B* is needed is given by:

$$mdo\_et_{1B} = 1 + 0.5 \times mdo\_et_{1B},$$
$$mdo\_et_{1B} = \frac{1}{1 - 0.5} = 2.$$

Note that when we need *B* again, *A* is also needed. Thus, $et_1 = max(1.667, 2) = 2$.

Since we have the worst-case buffer space constraints, the delay d_{11} will become

$$d_{11} = \left(max\left(\left(\frac{2,280}{38} - 15 - 40 \right), \left(\frac{1,220}{41} - 55 - 0 \right) \right) \right) \times 2 = 10.$$

3.3 A Real-World Example

The four semifinalists of the 1998 World Cup Football are Brazil, Holland, France, and Croatia as shown in Fig. 10.

The World Cup hypermedia documents are updated accordingly, there are now five hypermedia documents. One for the overview and four others (one for each team). The hypermedia database system contains three sites, one located in Europe, one in Asia and one in South America. Users will use the site nearest to them for retrieving the most up-to-date information.

The *OCPN* specifications of the D_1 and D_2 are shown in Fig. 11 and Fig. 12, respectively. D_3, D_4, and D_5 are similar to

1. Or $mdo\_et = 1 + ip_{jk} + ip_{jk}^2 + \ldots = (1 - ip_{jk})^{-1}$.

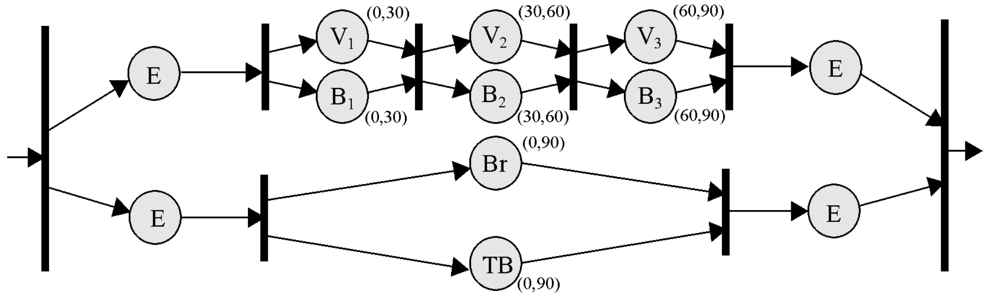


Fig. 12. Information about the Brazilian team (D_2).

TABLE 2
DESCRIPTION OF *MDO*S IN THE HYPERMEDIA DATABASE SYSTEM

Object	Description	Size (in Kilobytes)
V_1	Image files, an animation of a football drawn by hand.	401
V_2		1201
V_3		601
S_1	Audio files, recording the 1998 World Cup Song.	1501
S_2		1501
S_3		1501
Br	Photo of the Brazil football team.	301
Ho	Photo of the Holland football team.	301
Fr	Photo of the France football team.	301
Cr	Photo of the Croatia football team.	301
T	General information about the 1998 World Cup in text format.	101
B_1	The national anthem of Brazil.	1201
B_2		1201
B_3		1201
TB	Information of the Brazil team in text format.	201
H_1	The national anthem of Holland.	1201
H_2		1201
H_3		1201
TH	Information of the Holland team in text format.	201
F_1	The national anthem of France.	1201
F_2		1201
F_3		1201
TF	Information of the France team in text format.	201
C_1	The national anthem of Croatia.	1201
C_2		1201
C_3		1201
TC	Information of the Croatia team in text format.	201
E	Empty placeholder.	0

D_2 except that they contain other teams information, D_3 for Holland, D_4 for France, and D_5 for Croatia. The size and content of the *MDO*s are shown in Table 2.

The three sites of the hypermedia database system are fully connected, with the network transmission speed (in kilobytes per second) between them shown in Table 3.

TABLE 3
TRANSMISSION SPEED BETWEEN THE THREE SITES
(IN KILOBYTES PER SECOND)

	Asia	Europe	South America
Asia	0	15	18
Europe	15	0	10
South America	18	10	0

The probability matrix of browsing from a document to another document is,

$$\begin{array}{c} \\ D_1 \\ D_2 \\ D_3 \\ D_4 \\ D_5 \\ D_6 \end{array} \begin{array}{c} \begin{array}{cccccc} D_1 & D_2 & D_3 & D_4 & D_5 & D_6 \end{array} \\ \begin{bmatrix} 0 & 0.2 & 0.2 & 0.2 & 0.2 & 0.2 \\ 0.2 & 0 & 0.5 & 0.1 & 0.1 & 0.1 \\ 0.2 & 0.5 & 0 & 0.1 & 0.1 & 0.1 \\ 0.2 & 0.1 & 0.1 & 0 & 0.5 & 0.1 \\ 0.2 & 0.1 & 0.1 & 0.5 & 0 & 0.1 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \end{array},$$

where D_6 represents the end of the browsing session.

Using the methodology developed in Section 3 with *bpl* of the trees limited to 0.01, we have the following 5×5 matrix, R, representing the probability of retrieving *MDO*s in a document if we start browsing from a specific document,

$$\begin{array}{c} \\ D_1 \\ D_2 \\ D_3 \\ D_4 \\ D_5 \end{array} \begin{array}{c} \begin{array}{ccccc} D_1 & D_2 & D_3 & D_4 & D_5 \end{array} \\ \begin{bmatrix} 1 & 0.2 & 0.2 & 0.2 & 0.2 \\ 0.2 & 1 & 0.5 & 0.1 & 0.1 \\ 0.2 & 0.5 & 1 & 0.1 & 0.1 \\ 0.2 & 0.1 & 0.1 & 1 & 0.5 \\ 0.2 & 0.1 & 0.1 & 0.5 & 1 \end{bmatrix} \end{array}.$$

The users behavior in the three sites are different. The European and South American users are more interested in the teams representing their region. The Asian users do not have such a bias. However, Brazil is the favorite team in Asia and D_2 is accessed more often than other documents. The following gives the transition matrices for the initial browsing document for the three sites.

In Asia (P^1),

$$\begin{array}{c} \\ D_1 \\ D_2 \\ D_3 \\ D_4 \\ D_5 \\ N \end{array} \begin{array}{c} \begin{array}{cccccc} D_1 & D_2 & D_3 & D_4 & D_5 & N \end{array} \\ \begin{bmatrix} 0.3 & 0.2 & 0.1 & 0.1 & 0.1 & 0.2 \\ 0.2 & 0.3 & 0.2 & 0.1 & 0.1 & 0.1 \\ 0.2 & 0.2 & 0.2 & 0.1 & 0.1 & 0.2 \\ 0.2 & 0.2 & 0.1 & 0.2 & 0.2 & 0.1 \\ 0.2 & 0.2 & 0.1 & 0.2 & 0.2 & 0.1 \\ 0.4 & 0.3 & 0 & 0 & 0 & 0.3 \end{bmatrix} \end{array},$$

where N represents the not-start-browsing in the next transition.

The user behavior in Europe (P^2) is,

$$\begin{array}{c} \\ D_1 \\ D_2 \\ D_3 \\ D_4 \\ D_5 \\ N \end{array} \begin{array}{c} \begin{array}{cccccc} D_1 & D_2 & D_3 & D_4 & D_5 & N \end{array} \\ \begin{bmatrix} 0.2 & 0.1 & 0.1 & 0.2 & 0.2 & 0.2 \\ 0.2 & 0.2 & 0.1 & 0.2 & 0.2 & 0.1 \\ 0.2 & 0.1 & 0.2 & 0.2 & 0.2 & 0.1 \\ 0.1 & 0.1 & 0.1 & 0.3 & 0.3 & 0.1 \\ 0.1 & 0.1 & 0.1 & 0.3 & 0.3 & 0.1 \\ 0.3 & 0 & 0 & 0.3 & 0.3 & 0.1 \end{bmatrix} \end{array}.$$

Correspondingly, that in South America (P^3) is,

$$\begin{array}{c} \\ D_1 \\ D_2 \\ D_3 \\ D_4 \\ D_5 \\ N \end{array} \begin{array}{c} \begin{array}{cccccc} D_1 & D_2 & D_3 & D_4 & D_5 & N \end{array} \\ \begin{bmatrix} 0.2 & 0.2 & 0.2 & 0.1 & 0.1 & 0.2 \\ 0.1 & 0.3 & 0.3 & 0.1 & 0.1 & 0.1 \\ 0.1 & 0.3 & 0.3 & 0.1 & 0.1 & 0.1 \\ 0.2 & 0.2 & 0.2 & 0.2 & 0.1 & 0.1 \\ 0.2 & 0.2 & 0.2 & 0.1 & 0.2 & 0.1 \\ 0.3 & 0.3 & 0.3 & 0 & 0 & 0.1 \end{bmatrix} \end{array}.$$

In the long run, the probability matrix of the initial browsing document at each site, B, is (F is set to {1,000, 1,000, 1,000}),

$$\begin{array}{c} \\ S_1 \\ S_2 \\ S_3 \end{array} \begin{array}{c} \begin{array}{ccccc} D_1 & D_2 & D_3 & D_4 & D_5 \end{array} \\ \begin{bmatrix} 317 & 293 & 146 & 122 & 122 \\ 182 & 114 & 114 & 295 & 295 \\ 182 & 295 & 295 & 114 & 114 \end{bmatrix} \end{array}.$$

Denote the first row of this row as x. We can get x from P^1 by solving the linear equations, (similarly, use P^2 for row 2 and P^3 for row 3)

$$x \cdot (P^1)^{\mathrm{T}} = x$$

where A^{T} is the transpose of matrix A.

Discard the long run probability of not browsing and normailize the remaining probabilities, we will get the probability of using each document as entry point for every browsing session. Multiply these values with expected number of browsing sessions initiated in unit time, we will have the access frequencies of each document as initial browsing document in unit time.

By multiplying B matrix by the matrix of expected navigation path (R), we get the expected access frequency matrix for retrieving each document from each site in a period of time, A,

$$\begin{array}{c} \\ S_1 \\ S_2 \\ S_3 \end{array} \begin{array}{c} \begin{array}{ccccc} D_1 & D_2 & D_3 & D_4 & D_5 \end{array} \\ \begin{bmatrix} 454 & 454 & 380 & 290 & 290 \\ 345 & 266 & 266 & 502 & 502 \\ 345 & 502 & 502 & 266 & 266 \end{bmatrix} \end{array}.$$

From this matrix and from the *OCPN* specifications of each document, we can estimate the expected number of times each *MDO* is retrieved from each site, $A \times U$ (see Table 4).

The delay of the multimedia presentation of the document D_j in site S_i will be,

$$d_{ij} = max\left(\frac{size_k}{c_{site(k)\cdot i}} - dur_{jk} - start_{jk}\right), \quad for\ \forall k, u_{jk} = 1.$$

Suppose we allocate the *MDO*s randomly to the sites, for example, allocate O_i at site (i **mod** 3). The presentation delay of each document in each site is (in seconds),

	Site 1	Site 2	Site 3
Document 1	20.1	60.1	70.1
Document 2	36.7	90.1	60.1
Document 3	20.1	30.1	60.1
Document 4	50.1	60.1	90.1
Document 5	36.7	90.1	60.1

Using the expected number of retrievals for each document from each site, we can calculate the expected total delay in each site as,

	Site 1	Site 2	Site 3
Delay	58,597	128,108	124,548

The total expected delay of the whole system is 311,313 seconds.

TABLE 4
EXPECTED RETRIEVAL FREQUENCIES OF *MDO*S FROM EACH SITE

Media Item	Object	Site 1	Site 2	Site 3
O_1	V_1	1868.29	1881.82	1881.82
O_2	V_2	1868.29	1881.82	1881.82
O_3	V_3	1868.29	1881.82	1881.82
O_4	S_1	453.66	345.45	345.45
O_5	S_2	453.66	345.45	345.45
O_6	S_3	453.66	345.45	345.45
O_7	Br	907.32	611.36	847.73
O_8	Ho	834.15	611.36	847.73
O_9	Fr	743.90	847.73	611.36
O_{10}	Cr	743.90	847.73	611.36
O_{11}	T	453.66	345.45	345.45
O_{12}	B_1	453.66	265.91	502.27
O_{13}	B_2	453.66	265.91	502.27
O_{14}	B_3	453.66	265.91	502.27
O_{15}	TB	453.66	265.91	502.27
O_{16}	H_1	380.49	265.91	502.27
O_{17}	H_2	380.49	265.91	502.27
O_{18}	H_3	380.49	265.91	502.27
O_{19}	TH	380.49	265.91	502.27
O_{20}	F_1	290.24	502.27	265.91
O_{21}	F_2	290.24	502.27	265.91
O_{22}	F_3	290.24	502.27	265.91
O_{23}	TF	290.24	502.27	265.91
O_{24}	C_1	290.24	502.27	265.91
O_{25}	C_2	290.24	502.27	265.91
O_{26}	C_3	290.24	502.27	265.91
O_{27}	TC	290.24	502.27	265.91

Now, suppose we allocate the data differently, place *MDO*s O_1 to O_{11} in Asia, O_{12} to O_{19} in South America and others in Europe. The presentation delay will become,

	Site 1	Site 2	Site 3
Document 1	0	60.1	43.4
Document 2	36.7	90.1	0.1
Document 3	36.7	90.1	0.1
Document 4	50.1	0.1	90.1
Document 5	50.1	0.1	90.1

And the expected total time delay in each site will be,

	Site 1	Site 2	Site 3
Delay	59,666	68,828	63,050

The total expected delay of the whole system in this case is 191,544 seconds, with 38 percent reduction in the response time. In a system with three sites and five hypermedia documents, such performance improvement shows that good data allocation schemes are critically needed.

4 PROPOSED DATA ALLOCATION ALGORITHMS

The data allocation problem in its simple form has been shown to be NP-complete [4] and the problem discussed here is more complex than the simple case; there are k^m different allocation schemes for a system with m sites and k *MDO*s, implying that an exhaustive search would require $O(k^m)$ in the worst case to find the optimal solution. Therefore, we must use heuristic algorithms to solve the problem.

4.1 The Hill-Climbing Approach

We have developed an algorithm based on the Hill-Climbing technique to find a near optimal solution. The data allocation problem solution consists of the following two steps:

1) Find an initial solution.
2) Iteratively improve the initial data allocation by using the hill climbing heuristic until no further reduction in total response time can be achieved. This is done by applying some operations on the initial allocation scheme. Since there are finite number of allocation schemes, the heuristic algorithm will complete its execution.

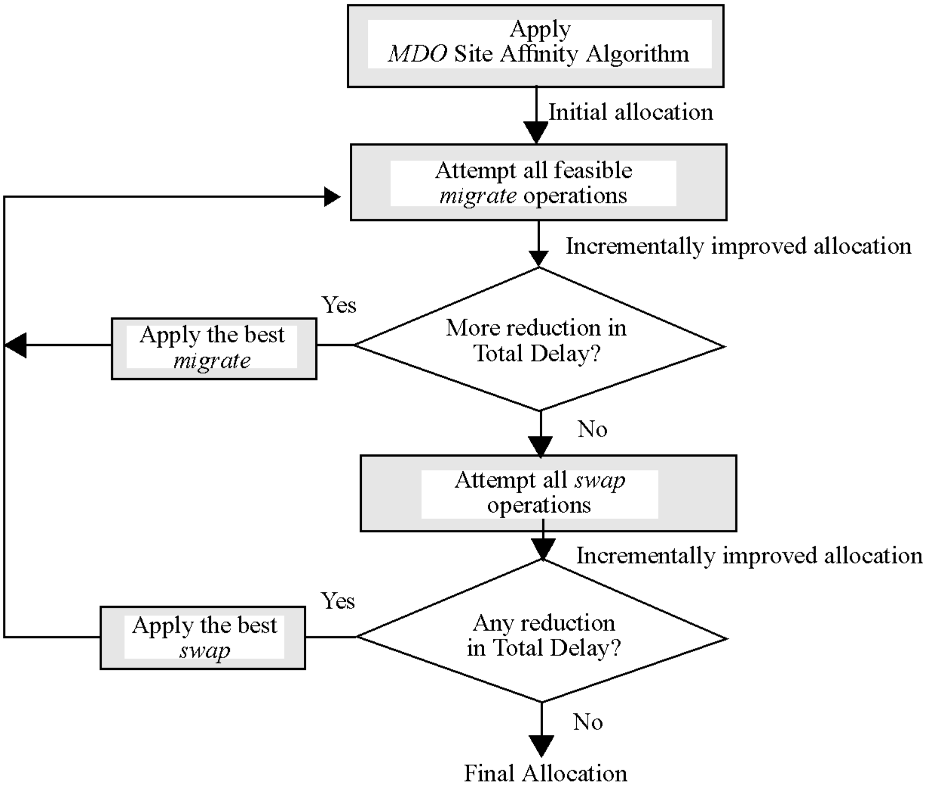


Fig. 13. Steps in the Hill-Climbing algorithm for data allocation problem.

For step one, one possibility is to obtain the initial solution by allocating the *MDO*s to the sites randomly. However, a better initial solution can be generated by allocating an *MDO* to the site which retrieves it most frequently (this information can be obtained from the matrix $A \times U$). If that site is already saturated, we allocate the *MDO* to the next site that needs it the most. We call this method the *MDO site affinity algorithm*.

In the second step, we apply some operations on the initial solution to reduce the total response time. Two types of operations are defined, namely

- *migrate* (move *MDO*s from its currently allocated site to another site), and
- *swap* (swap the locations of one set of *MDO*s with the locations of another set of *MDO*s).

These operations are iteratively applied until no more reduction is observed in the total response time. Fig. 13 shows the major steps in the Hill-Climbing heuristic algorithm.

The set of *migrate* and *swap* operations are as follows.

- *migrate*(O_j, S_i): Move *MDO* O_j to S_i. This operation can be applied to each *MDO*, and an *MDO* can be moved to any one of the $m - 1$ sites at which it is not located. Therefore, there can be a maximum of $k(m - 1)$ migrate operations that can be applied during each iteration.
- *swap*$(O_x, O_{x'})$: Swap the location of *MDO* O_x with the location of *MDO* $O_{x'}$. This operation can be applied to each distinct pair of *MDO*s. Therefore, there can be a maximum of $k(k - 1)/2$ swap operations that can be applied during each iteration. Although this operation is equivalent to two *migrate* operations, it is necessary as some of the sites may be already saturated such that we cannot *migrate MDO* to it any more.

Here, we apply the Hill-Climbing algorithm to the example presented in Section 3.1. First, we calculate the matrix $A \times U$ for finding the initial allocation solution:

$$A \times U = \begin{bmatrix} 585 & 1{,}171 & 1{,}266 \\ 700 & 972 & 1{,}152 \\ 715 & 1{,}493 & 1{,}178 \end{bmatrix}.$$

It is easy to see that the initial allocation is $\{S_3, S_3, S_1\}$. Then, the total response time given by this initial allocation is 8,860 + 43,280+ + 23,910 = 76,050.

Table 5 shows the migrate and swap operations applied to improve the initial solution provided by the *MDO* site affinity algorithm. For example, after applying the *migrate*(O_1, S_1), the total response time delay reduces from 76,050 to 66,170. The solution to the data allocation problem generated by the Hill-Climbing algorithm is $\{S_1, S_3, S_1\}$ (i.e., *MDO A* is allocated at site S_1, *MDO B* at site S_3, and *MDO C* at site S_1) with the total response time incurred to execute the query being equal to 66,170. Table 6 shows all of the feasible allocation schemes and the total response time incurred for each of them.

Comparing the Hill-Climbing algorithm with the exhaustive solution, we observe that the Hill-Climbing algorithm can generate the optimal solution (which is allocation number 7 in Table 6). However, the Hill-Climbing algorithm does not guarantee an optimal solution.

TABLE 5
OUTPUT OF THE HILL-CLIMBING ALGORITHM FOR DATA ALLOCATION

Current Allocation	Operation Type	Operation Applied	New Allocation	Total Response Time Delay
$\{S_3, S_3, S_1\}$	—	Initial solution	—	76050
$\{S_3, S_3, S_1\}$	migrate	$migrate(O_1, S_1)$	$\{S_1, S_3, S_1\}$	66170
$\{S_1, S_3, S_1\}$	migrate	None applied	—	—
$\{S_1, S_3, S_1\}$	swap	None applied	—	—

TABLE 6
ENUMERATION OF ALL ALLOCATION SCHEMES AND THEIR RESPECTIVE RESPONSE TIME DELAYS

Allocation Number	Allocation Scheme	Total Delay in each Site S_1	S_2	S_3	Total Delay
1	$\{S_1, S_1, S_1\}$	0	41940	24230	66170
2	$\{S_1, S_1, S_2\}$	37870	14500	32990	85360
3	$\{S_1, S_1, S_3\}$	32510	48540	5450	86500
4	$\{S_1, S_2, S_1\}$	10	41940	32990	74940
5	$\{S_1, S_2, S_2\}$	37870	14500	32360	84730
6	$\{S_1, S_2, S_3\}$	32510	48540	5450	86500
7	$\{S_1, S_3, S_1\}$	10	41940	24220	66170
8	$\{S_1, S_3, S_2\}$	37870	14500	32990	85360
9	$\{S_1, S_3, S_3\}$	32520	48540	5440	86500
10	$\{S_2, S_1, S_1\}$	11438	40650	29130	81210
11	$\{S_2, S_1, S_2\}$	39090	10	37900	77000
12	$\{S_2, S_1, S_3\}$	33730	47250	12260	93240
13	$\{S_2, S_2, S_1\}$	11430	40640	29140	81210
14	$\{S_2, S_2, S_2\}$	39090	0	37900	76990
15	$\{S_2, S_2, S_3\}$	33730	47240	12260	93230
16	$\{S_2, S_3, S_1\}$	11430	40650	29130	81210
17	$\{S_2, S_3, S_2\}$	39090	10	37900	77000
18	$\{S_2, S_3, S_3\}$	33730	47250	5230	86210
19	$\{S_3, S_1, S_1\}$	160	43280	23920	67360
20	$\{S_3, S_1, S_2\}$	38020	18100	32690	88100
21	$\{S_3, S_1, S_3\}$	32660	49880	20	82560
22	$\{S_3, S_2, S_1\}$	8860	43280	23930	76070
23	$\{S_3, S_2, S_2\}$	38020	18100	32690	88810
24	$\{S_3, S_2, S_3\}$	32660	49890	30	82580
25	$\{S_3, S_3, S_1\}$	8860	43280	23910	76050
26	$\{S_3, S_3, S_2\}$	38020	18100	50770	106890
27	$\{S_3, S_3, S_3\}$	32660	49880	0	82540

4.2 The Neighborhood Search Approach

One drawback of the Hill-Climbing approach is its high complexity. Another problem is that the algorithm can be trapped in some local minimum. This is because the exchange or migration of MDO is done only if the movement will give a better solution. To increase the chances of finding the global optimal solution, we must introduce some probabilistic jumps [11]. The probabilistic jumps must be large enough by involving MDOs that can have a great effect on the solution quality. Otherwise, if the jump is small, the algorithm may remain trapped in the same local minimum. Thus, before the executing the algorithm, we must determine which subset of the MDO set is important.

One possible set of important MDOs are the MDOs that are presented at the beginning of some hypermedia document. The reason being that when we browse a hypermedia document, we must retrieve and use the starting MDO immediately, so their transmission delay will have a great effect on the overall document presentation delay. Thus, we have two sets of MDOs: critical MDOs and noncritical MDOs.

Based on the above discussion, the neighborhood search algorithm is designed as follows,

(1) Get the initial allocation scheme use *MDO* site affinity algorithm (see Section 4.1);
(2) Construct the two list of critical *MDO* (*CMDO*) and other *MDO* (*OMDO*);
(3) BestRes = infinity; searchcount = 0;
(4) **repeat**
(5) searchstep = 0; counter = 0;
(6) **do** { /* neighborhood search */
(7) Choose an *OMDO* randomly and migrate it to a random site;
(8) Choose two *OMDO*s and swap them;
(9) Compare the two resultant response times and select the better one;

```
                If total response time is smaller then
                BestRes, do the movement and set
                counter to 0. Otherwise, increase
                counter by 1;
            } while (searchstep++ < MAXSTEP and
            counter < MARGIN);
            if BestRes > ResTime(NewScheme) then
                BestScheme = NewScheme;
                BestRes = ResTime(NewScheme);
            endif
            Choose a CMDO randomly and migrate it
            to a random site or choose two CMDOs
            and swap them; /* probabilistic jump */
        until (searchcount++ > MAXCOUNT);
        searchstep = 0; counter = 0;
        do { /* neighborhood search */
            Choose a CMDO randomly and migrate it
            to a random site;
            Choose two CMDOs and swap them;
            Compare the two resultant response
            times and select the better one;
            If total response time is smaller then
            BestRes, do the movement and set counter
            to 0. Otherwise, increase counter by 1;
    } while (searchstep++ < MAXSTEP and
      counter < MARGIN);
    if BestRes > ResTime(NewScheme) then
            BestScheme = NewScheme;
            BestRes = ResTime(NewScheme);
      endif
```

The random algorithm starts with an initial solution using the site affinity algorithm and then constructs two lists of *MDO*s. It then tries to merge OMDOs to some random sites by using either the migrate operation or the swap operation whichever gives more improvement in the solution quality. It continues to do so for *MAXSTEP* times but will stop if there is no improvement in *MARGIN* number of trials. It then chooses one of the CMDOs and migrates it to a random site or swap it with another randomly selected CMDO. This continues for *MAXCOUNT* times. The algorithm preserves the best solution found so far and then performs a neighborhood search on CMDOs again for further improvement.

The worst-case running time of the algorithm is

$$O(MAXSTEP \times MAXCOUNT).$$

It is reasonable to set *MAXSTEP* as a multiple of the number of *OMDO*s and the number of sites. Similarly, *MAXCOUNT* is set to be a multiple of the number of *CMDO*s and the number of sites.

$$MAXSTEP = a \cdot \langle m \cdot \ |OMDO\text{s}| \rangle$$
$$MAXCOUNT = b \cdot \langle m \cdot \ |CMDO\text{s}| \rangle$$

With these assumptions, we will have an $O(m^2k^2)$ algorithm.

5 Results

In this section, we present the experimental results for the data allocation algorithms described in the previous sections. Comparisons among these algorithms will be made by considering the quality of solutions and the algorithm running times.

5.1 Workload

The example considered in the previous section was used for illustrating how the Hill-Climbing algorithm works. But it had only four documents, three *MDO*s and three sites and thus only 27 different allocation schemes. Since the solutions were to be compared with the optimal solutions generated by an exhaustive search which takes a large amount of time to experiment for a distributed database system even with moderate number of sites and *MDO*s (for k *MDO*s and m sites, there are k^m allocation schemes, and for each allocation scheme the total response time needs to be calculated), the problem sizes of the experiments we conducted were limited.

We conducted 25 experiments with the number of *MDO*s ranging from four to eight, and the number of sites ranging from four to eight. Each experiment consisted of 100 allocation problems with the number of sites and the number of *MDO*s fixed. Each allocation problem had between four and 16 documents, and each document used a subset of the *MDO*s with its own temporal constraints on them. The communication network, the *MDO* sizes, the link costs, and the temporal constraints between *MDO*s in each document were randomly generated from a uniform distribution. The two data allocation algorithms described above were tested for every case and statistics were collected.

5.2 Comparison of Allocation Costs

In Table 7 and Table 8, for each of the experiments conducted in a columnwise fashion, we list the following:

1) the number of *MDO*s,
2) the number of sites,
3) the number of problems,
4) the number of problems for which the algorithm generated the optimal solution,
5) the average percentage deviation from the optimal solution for those allocations for which the algorithm did not generate optimal solution,
6) the number of near optimal solutions with deviation of less than 5 percent,
7) the number of near optimal solutions with deviation of 5 percent or more but less than 10 percent,
8) the number of near optimal solutions with deviation of 10 percent or more but less than 20 percent, and
9) the number of near optimal solutions with deviation of 20 percent or more.

From Table 7, we note that the Hill-Climbing algorithm generated optimal solutions for a large number of problems: 2,173 cases out of a total of 2,500 cases, corresponding to about 87 percent of the test cases. Most of the nonoptimal solutions are in the range of 0-5 percent deviation from the optimal solution while a few solutions are in the range of equal to or more than 20 percent. The average percentage (only for nonoptimal cases) is about 9.1557 across all cases. These results indicate that the Hill-Climbing algorithm is able to generate high quality solutions.

Table 8 summarizes the results of the random search algorithm. Compared to the Hill-Climbing algorithm, the

TABLE 7
EXPERIMENTAL RESULTS OF THE HILL-CLIMBING ALGORITHM

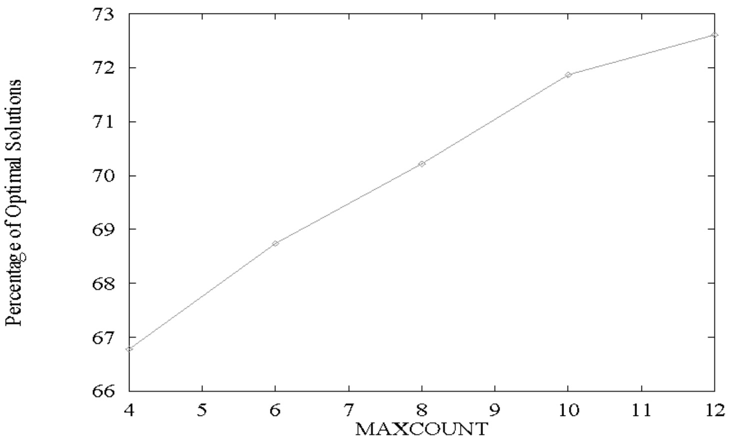
No. of Sites	No. of *MDOs*	No.of Problems	No. of Opt. Sol.	Aver. % Deviation	Number of Sol. with deviation in range			
					(0, 5%)	[5, 10%)	[10, 20%)	[20%, -)
4	4	100	93	15.42	3	2	0	2
4	5	100	91	25.08	2	1	1	5
4	6	100	91	8.64	5	0	3	1
4	7	100	90	9.23	7	1	0	2
4	8	100	81	5.37	10	7	1	1
5	4	100	93	15.42	3	2	0	2
5	5	100	91	25.08	2	1	1	5
5	6	100	91	8.64	5	0	3	1
5	7	100	90	9.23	7	1	0	2
5	8	100	81	5.37	10	7	1	1
6	4	100	92	19.36	3	1	0	4
6	5	100	89	6.30	7	1	2	1
6	6	100	84	12.59	6	3	4	3
6	7	100	83	4.40	11	3	2	1
6	8	100	83	11.54	13	1	1	2
7	4	100	97	5.23	1	2	0	0
7	5	100	90	7.31	7	1	1	1
7	6	100	79	8.74	15	2	2	2
7	7	100	85	4.41	10	4	1	0
7	8	100	82	11.66	9	3	3	3
8	4	100	88	3.06	9	3	0	0
8	5	100	87	4.81	12	0	0	1
8	6	100	84	11.20	10	2	3	1
8	7	100	81	8.79	9	5	2	3
8	8	100	77	4.00	18	2	2	1

TABLE 8
EXPERIMENTAL RESULTS OF THE RANDOM SEARCH ALGORITHM

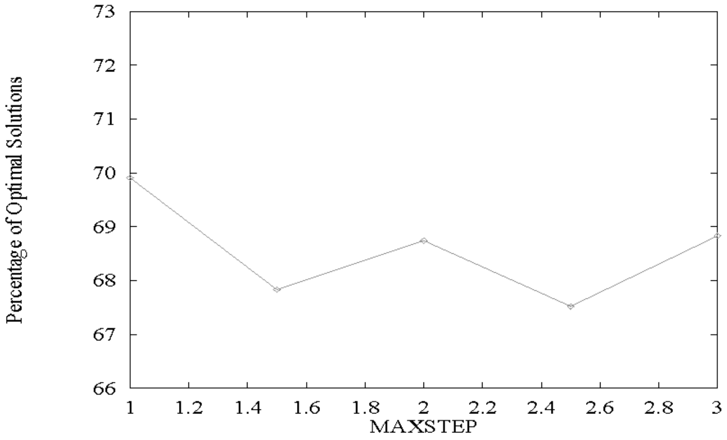
No. of Sites	No. of *MDOs*	No.of Problems	No. of Opt. Sol.	Aver. % Deviation	Number of Sol. with deviation in range			
					(0, 5%)	[5, 10%)	[10, 20%)	[20%, -)
4	4	100	98	11.01	1	0	0	1
4	5	100	90	12.72	5	1	1	3
4	6	100	80	7.00	12	3	3	2
4	7	100	73	4.90	19	4	3	1
4	8	100	70	5.10	19	4	6	1
5	4	100	97	1.10	3	0	0	0
5	5	100	88	4.97	9	1	0	2
5	6	100	77	4.24	17	3	2	1
5	7	100	75	5.13	17	5	0	3
5	8	100	62	5.63	26	7	3	2
6	4	100	92	4.89	5	3	0	0
6	5	100	88	5.01	8	3	1	0
6	6	100	74	3.05	21	3	2	0
6	7	100	68	4.08	22	7	3	0
6	8	100	56	2.87	37	5	2	0
7	4	100	86	1.86	13	0	1	0
7	5	100	84	2.63	13	2	1	0
7	6	100	68	2.91	27	3	2	0
7	7	100	63	3.55	27	6	3	1
7	8	100	62	4.49	27	6	4	1
8	4	100	90	1.51	10	0	0	0
8	5	100	81	2.44	16	2	1	0
8	6	100	68	2.34	28	3	1	0
8	7	100	66	2.63	27	7	0	0
8	8	100	50	3.65	40	6	4	0

number of optimal solutions is less but the average deviation is similar. The Hill-Climbing algorithm is a high-complexity algorithm and is expected to yield better results. On the other hand, the complexity of the Random Search algorithm is low and its performance is satisfactory.

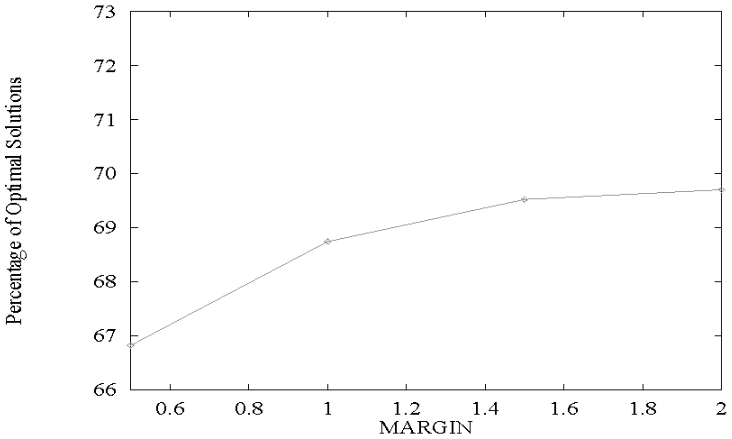
The Random Search algorithm may depend on the values selected for its parameters such as *MAXCOUNT*, *MAXSTEP*, and *MARGIN*. As mentioned above, these parameters are multiples of the number of MDOs and the number of sites. To study the sensitivity of the algorithm to these parameters and seek the possibility of further improvement in the solution quality, we varied these parameters. The results are plotted in Fig. 14 which indicates the number of optimal solutions



(a)



(b)



(c)

Fig. 14. Sensitivity of parameters in the random search algorithm: (a) percentage of optimal solutions obtained against MAXCOUNT; (b) percentage of optimal solutions obtained against MAXSTEP; (c) percentage of optimal solutions obtained against MARGIN.

against these numbers. In Fig. 14a, MAXCOUNT is varied from four to 12 times the number of sites multiplied by the number of CMDOs. This figure indicates that MAXCOUNT does have an impact on the performance and with a higher value can yield more optimal solutions. However, the performance saturates beyond a certain range. The parameter *MAXSTEP* (see Fig. 14b) does not seem to impact the performance and is in fact included in the algorithm to control its searching steps. The *MARGIN* algorithm (Fig. 14c) does have an impact but, similar to *MAXCOUNT*, has a certain range of values that are more effective (such as 1.0 to 1.5).

TABLE 9
AVERAGE RUNNING TIMES OF ALL ALGORITHMS (IN MSECS)

No. of Sites	No. of *MDO*s	Exhaustive Search	Hill Climbing	Random Search
4	4	7.63	38.95	27.44
4	5	62.99	90.42	51.37
4	6	276.57	179.85	81.06
4	7	1088.42	355.07	121.39
4	8	6457.69	1014.08	183.12
5	4	18.34	68.36	45.57
5	5	134.40	153.56	81.77
5	6	1085.80	313.18	134.38
5	7	8330.39	821.18	202.27
5	8	50224.66	2011.78	314.17
6	4	51.88	85.14	67.28
6	5	529.97	236.69	120.29
6	6	5159.62	630.65	192.95
6	7	38925.23	1385.54	289.33
6	8	273494.96	2885.52	446.66
7	4	176.10	166.09	97.67
7	5	1610.73	401.65	165.79
7	6	22203.47	992.97	277.62
7	7	515439.65	2300.59	427.97
7	8	1587830.28	4721.17	620.85
8	4	333.23	169.28	134.06
8	5	3560.16	567.55	217.09
8	6	40439.50	1520.83	355.32
8	7	576860.17	4369.09	697.61
8	8	5755754.63	12053.53	875.84

5.3 Comparison of Running Times

Table 9 contains the average running times of the two algorithms for each experiment. For comparison, the time taken to generate the optimal solutions by using an exhaustive search are also listed. The algorithms were implemented on a SPARC IPX workstation and the timing data was measured in milliseconds. As can be seen from the table, although the Random Search algorithm took much shorter time compared with the exhaustive search and about 1 order of magnitude less time than the Hill-Climbing Approach. Such margins become highly significant when the problem size gets large. Therefore, while the Hill-Climbing algorithm may be preferred for small problem sizes, the Random algorithm would be a better choice for large problem sizes.

From the experimental results presented in the previous section, we observe that there is a trade-off between the execution time and solution quality. The random search algorithm is very cost-effective if fast execution is desired. If the solution quality is the more prominent factor, the Hill-Climbing approach is a viable choice for an off-line allocation.

6 RELATED WORK

There is little doubt that the next generation of information processing systems are multimedia in nature and are built on top of a communication network. These resultant systems will be called as distributed multimedia systems (*DMS*). Multimedia documents are different from the traditional single-media documents in that they have synchronization requirements between different media. One of the problems encountered in *DMS*s is the lack of specification models for capturing temporal constraints among various objects. Both *HyTime* [6], [16] and *OCPN* [13] are developed to solve this problem. Numerous variations in *OCPN* have been proposed (see [25] for a survey of *OCPN* and its variants). As the original *OCPN* model does not incorporate user interactions, an extension called *AOCPN* is developed in [17] for modeling user interactions such as stop and resume, reverse and terminate.

One limitation of the OCPN model is that there is no explicit way for modeling the navigation path from one document to another. This is because of the restriction of *OCPN* that each place can only have one outgoing link, but one can browse to many documents from the current document.

In [20], a model called *TPN* is proposed and can be used to model user interactions that are timed (i.e., can occur only in a predefined period). This model can specify the browsing semantics (when and how). The main problem with this model is that it cannot capture the situation where the user continues to with a document after its first presentation. The time constraint of the model is too strict that when the presentation of a document is ends, that is, one of the transition must be fired (either go to another document or terminate).

The objective of the above two extensions of *OCPN* is to develop models suitable for interactive multimedia document specifications. However, these models only specify the possible path of document presentation, they do not provide the information about the expected number of times each state (representing *MDO* or hypermedia document) is needed in a unit time interval. Without this information, the total response time of the *DMS* cannot be estimated. Another extension of *OCPN* called *XOCPN* has been proposed which models the object sizes and synchronization at a finer level [26].

Our main contribution is the development of the probabilistic models for navigation and hypermedia presentation. By analyzing these models, we can estimate the total response time of the *DMS*. We can use the cost function to compare different *MDO* allocation schemes generated by different allocation algorithms. Our main concern is in finding optimal data allocation scheme for *MDO*s in different sites of a distributed hypermedia database system such that the total response time of the database system is minimal.

The enforcement of the synchronized multimedia presentation is another important aspect of a *DMS*. In [14], the general composition (including the spatio and temporal constraints) problem of distributed multimedia objects is discussed and a scheme for mapping the whole composition process to network resources is proposed.

By using the buffers in each site efficiently, the delays of subsequent querying will be reduced significantly if the data requested is still in a buffer. A buffer management scheme for continuous media sharing can be found in [9]. Another component that will have a great effect on the presentation is the network [22]. By partitioning the network bandwidth to channels, we can use the network as efficiently as possible [2]. The original problem considered in [2] is for I/O buffering but the principle can be easily adopted to network channel utilization. If a fault has occurred in the network during a multimedia presentation, the presentation may stop and need to wait for data arrival, which is unacceptable. In [24], a scheme of adaptive presentation management is proposed to be used with slow data arrival. This scheme lowers the presentation quality but ensures that the presentation is carried out smoothly.

7 Conclusions

In this paper, we address the problem of response time driven allocation of MDOs for browsing hypermedia documents in distributed environments. This problem addresses both the response time optimization, and adherence to synchronization constraints in the context of data allocation. We develop a probabilistic navigational model for modeling the user behavior while browsing hypermedia documents. This model is used to calculate the expected number of accesses to each hypermedia document from each site. The synchronization constraints for presenting the MDOs of hypermedia documents are modeled by using the OCPN specification. A cost model is developed to calculate the average response time observed by the end users while browsing a set of hypermedia documents for a given allocation of MDOs. This cost model is generalized to take into consideration end-user interaction while accessing MDOs, and limited buffer space constraints at the end-user site. A real-life example is presented to illustrate the utility of the cost model and motivate the need for a good data allocation of MDOs. After that, two MDO data allocation algorithms, one based on Hill-climbing heuristic, and other based on Neighborhood search are proposed.

The two algorithms use extreme approaches:

1) a high complexity extensive incremental strategy, and
2) fast random search.

Results indicate that there is a trade-off between the execution time and solution quality. The neighborhood search algorithm is cost-effective if fast execution is desired. If the solution quality is the more prominent factor, the Hill-Climbing approach is a viable choice for small problem sizes.

There are some other related aspects we have not explored in this paper. Dynamic data allocation is one such aspect. In environment where user behavior change frequently, the allocation scheme must be adapted to maintain the system performance. Another important aspect is data replication. For MDOs that are read only, duplicating them to all the sites needing them can enhance the overall system performance. However, such strategy may not be feasible due to limited storage space. For static allocation scheme, we can employ some variant of the greedy algorithm to give a near optimal data replication scheme. The real challenging situation is to develop dynamic data replication algorithms. There are several methodologies in the literature [27] to attack this problem for reducing data transfer cost, we are currently adapting these approaches to this problem.

References

[1] P.B. Berra, C.Y.R. Chen, A. Ghafoor, C.C. Lin, T.D.C. Little, and D. Shin, "Architecture for Distributed Multimedia Systems," *Computer Comm.*, vol. 13, no. 4, pp. 217-31, May 1990.

[2] S. Chaudhuri, S. Ghandeharizadeh, and C. Shahabi, "Avoiding Retrieval Contention for Composite Multimedia Objects," *Proc. VLDB, 21st Int'l Conf. Very Large Data Bases*, pp. 287-298, 1995.

[3] R. Erfle, "HyTime as the Multimedia Document Model of Choice," *Proc. Int'l Conf. Multimedia Computing and Systems*, pp. 445-454, 1994.

[4] K.P. Eswaran, "Placement of Records in a File and File Allocation in a Computer Network," *Information Processing*, pp. 304-307, 1974.

[5] A. Ghafoor, "Multimedia Database Management Systems," *ACM Computing Surveys*, vol. 27, no. 4, pp. 593-598, Dec. 1995.

[6] C.F. Goldfarb, "Standards-HyTime: A Standard for Structured Hypermedia Interchange," *Computer*, vol. 24, no. 8, pp. 81-84, Aug. 1991.

[7] E. James, "Media and Hypermedia," *Proc. IEE Colloquium on Large Databases in Press and Publishing: The Present and the Future*, Digest no. 101, pp. 1-2, 1990.

[8] D.S. Johnson, C.H. Papadimitriou, and M. Yannakakis, "How Easy is Local Search," *J. Computer and System Sciences*, vol. 37, no. 1, pp. 79-100, Aug. 1988.

[9] M. Kamath, K. Ramamritham, and D. Towsley, "Continuous Media Sharing in Multimedia Database Systems," *Proc. Fourth Int'l Conf. Database Systems for Advanced Applications*, pp. 79-86, 1995.

[10] F. Kappe, G. Pani, and F. Schnabel, "The Architecture of a Massively Distributed Hypermedia System," *Internet Research*, vol. 3, no. 1, pp. 10-24, Spring 1993.

[11] Y.-K. Kwok, I. Ahmad, and J. Gu, "FAST: A Low-Complexity Algorithm for Efficient Scheduling of DAGs on Parallel Processors," *Proc. 25th Int'l Conf. Parallel Processing*, vol. II, pp. 150-157, Aug. 1996.

[12] Y. Kwok, K. Karlapalem, I. Ahmad, and N.M. Pun, "Design and Evaluation of Data Allocation Algorithms for Distributed Multimedia Database Systems," *IEEE J. Selected Areas in Comm.*, vol. 14, no. 7, pp. 1,332-1,348, Sept. 1996.

[13] T.D.C. Little, "Synchronization and Storage Models for Multimedia Objects," *IEEE J. Selected Areas in Comm.*, vol. 8, no. 3, pp. 413-427, Apr. 1990.

[14] T.D.C. Little, "Spatio-Temporal Composition of Distributed Multimedia Objects for Value-Added Networks," *Computer*, vol. 24, no. 10, pp. 42-50, Oct. 1991.
[15] M.A. Marsan, G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis, *Modelling with Generalized Stochastic Petri Nets*, Wiley, 1995.
[16] S.R. Newcomb, "Multimedia Interchange Using SGML / HyTime," *IEEE Multimedia*, vol. 2, no. 2, pp. 86-89, Summer 1995.
[17] B. Prabhakaran and S. V. Raghavan, "Synchronization Models for Multimedia Presentation with User Participation," *Multimedia Systems*, vol. 2, pp. 53-62, 1994.
[18] R. Rada, "Hypertext, Multimedia and Hypermedia," *New Review of Hypermedia and Multimedia, Applications, and Research*, vol. 1, pp. 1-21, 1995.
[19] G.R. Rao, V. Balasubramanian, and B.A. Suresh, "Integration of Hypertext and Object-Oriented Databases for Information Retrieval," *Proc. 1993 IEEE 19th Ann. Northeast Bioeng. Conf.*, pp. 201-204, May 1993.
[20] J. Song, Y.N. Doganata, M.Y. Kim, and A.N. Tantawi, "Modeling Timed User-Interactions in Multimedia Documents," *Proc. Int'l Conf. Multimedia Computing and Systems*, pp. 407-416, 1996.
[21] P.D. Stotts and R. Furuta, "Petri-Net-Based Hypertext: Document Structure with Browsing Semantics," *ACM Trans. Information Systems*, vol. 7, no. 1, pp. 3-29, Jan. 1989.
[22] V.S. Subrahmanian, *Multimedia Database Systems: Issues and Research Directions*, Springer, 1996.
[23] H.M. Taylor and S. Karlin, *An Introduction to Stochastic Modeling*, Academic Press, 1994.
[24] H. Thimm and W. Klas, "δ-sets for Optimized Reactive Adaptive Playout Management in Distributed Multimedia Database Systems," *Proc. 12th Int'l Conf. Data Eng.*, pp. 584-592, 1996.
[25] S. Vuong, K. Cooper, and M. Ito, "Specification of Synchronization Requirements for Distributed Multimedia Systems," *Proc. Int'l Workshop Multimedia Software Development*, pp. 110-119, 1996.
[26] M. Woo, N.U. Qazi, and A. Ghafoor, "A Synchronization Framework for Communication of Pre-Orchestrated Multimedia Information," *IEEE Network*, pp. 52-61, Jan. / Feb. 1994.
[27] O. Wolfson, S. Jajodia, and Y. Huang, "An Adaptive Data Replication Algorithm," *ACM Trans. Database Systems*, vol. 22, no. 2, pp. 55-314, June 1997.

Siu-Kai So received his bachelors degree in computer science from the Hong Kong University of Science and Technology in 1996. Currently, he is now pursuing his masters degree from the same university. His research interests are in the areas of distributed systems, database systems, multimedia applications and algorithm design.

Ishfaq Ahmad received a BSc degree in electrical engineering from the University of Engineering and Technology, Lahore, Pakistan, in 1985; and his MS degree in computer engineering and PhD degree in computer science, both from Syracuse University, in 1987 and 1992, respectively. He now is an associate professor in the Department of Computer Science at the Hong Kong University of Science and Technology. His research interests are in the areas of parallel programming tools, scheduling, and mapping algorithms for scalable architectures, video technology, and interactive multimedia systems. He has published extensively in the above areas. He has received numerous research and teaching awards, including the Best Student Paper Award at Supercomputing '90 and Supercomputing '91, and the Teaching Excellence Award of the School of Engineering at the Hong Kong University of Science and Technology. He has served on the committees of various international conferences, has been a guest editor for two special issues of *Concurrency Practice and Experience* related to resource management, and is co-guest-editor of a forthcoming special issue of the *Journal of Parallel and Distributed Computing* on software support for distributed computing. He serves on the Editorial Board of *Cluster Computing*. He is a member of the IEEE and the IEEE Computer Society.

Kamalakar Karlapalem received his MStat degree from the Indian Statistical Institute, Calcutta, India, in 1985; the MTech degree in computer science from the Indian Institute of Technology, Kharagpur, in 1986; and the PhD degree from the Georgia Institute of Technology in December 1992. He has been an assistant professor in the Department of Computer Science at the Hong Kong University of Science and Technology since 1993. He worked for two years as a software engineer at Tata Consultancy Services, India, and the National Informatics Centre, India. He was a student intern at Hewlett-Packard, Cupertino, California, working on conceptualization, design, and implementation of a distributed relational database system. He is interested in the following research areas: cooperative problem solving technology, data warehousing and data mining, distributed database design/redesign, multimedia data allocation, object-oriented class partitioning and workflow systems, and electronic commerce security. He is a member of the IEEE, the IEEE Computer Society, and ACM.