<table>
<thead>
<tr>
<th><strong>Title</strong></th>
<th>Empirical evaluation of data allocation algorithms for distributed multimedia database systems</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Author(s)</strong></td>
<td>Karlapalem, Kamalakar; Ahmad, Ishfaq; So, SiuKai; Kwok, YuKwong</td>
</tr>
<tr>
<td><strong>Citation</strong></td>
<td>Proceedings - Ieee Computer Society's International Computer Software And Applications Conference, 1997, p. 296-301</td>
</tr>
<tr>
<td><strong>Issued Date</strong></td>
<td>1997</td>
</tr>
<tr>
<td><strong>URL</strong></td>
<td><a href="http://hdl.handle.net/10722/46070">http://hdl.handle.net/10722/46070</a></td>
</tr>
<tr>
<td><strong>Rights</strong></td>
<td>©1997 IEEE. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from the IEEE.; This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License.</td>
</tr>
</tbody>
</table>
Empirical Evaluation of Data Allocation Algorithms for Distributed Multimedia Database Systems

Kamalakar Karlapalem, Ishfaq Ahmad, Siu-Kai So and Yu-Kwong Kwok
Department of Computer Science
The Hong Kong University of Science and Technology, Hong Kong
Email:{kamal, iahmad, kai, csricky}@cs.ust.hk

Abstract
Given a distributed multimedia database system and a set of queries as well as their frequencies from each site, the objective of a data allocation algorithm is to locate the multimedia data objects (MDOs) at different sites so as to minimize the total data transfer cost incurred in executing the queries. The data allocation problem, however, is NP-complete, and thus requires fast heuristics to generate efficient solutions. In this paper we propose three data allocation algorithms which are based on a genetic technique, an evolutionary process, and neural networks. We have implemented and evaluated these algorithms on our distributed multimedia database system test-bed. A comparison of the algorithms reveals trade-offs between their solution quality and time-complexity.

1 Introduction
A distributed multimedia database system [2], [7], [13], [14], [15] is a database system loosely coupled with a multimedia data provider introduced in [10]. In this architecture, Multimedia Data Provider (MDP) enables users to retrieve multimedia data objects (MDOs) from different sites. A Common Multimedia User Interface (CMUI) identifies the access queries and facilitates presenting the result to the user. The synchronization for the presentation of the multimedia data is handled by the CMUI. Whereas, the Multimedia Data Provider (MDP) identifies the relevant multimedia data for an user query and facilitates shipping of the multimedia data to the CMUI. The CMUI is a client process and the Distributed Database Management System (DDBMS) and the MDP are server processes.

A major component of multimedia query execution cost is the data transfer cost [1], [5], [6]. The MDOs are made of two kinds of data. The first is the single-media data that is managed by the DDBMS servers, such as relations (fragments), records, etc. The second is the multimedia data, such as audio, video, and image, managed by the MDP servers. These two types of data are managed by different specialized storage managers, and need to be retrieved for the user queries. Optimal allocation of MDOs is a complex problem because of mutual interdependency between allocation scheme (which gives the location of each of the MDOs at various sites of a distributed database system) and query optimization strategy (which decides how a query can be optimally executed, given an allocation scheme) [11], [16], [17]. The processing strategy of distributed multimedia objects retrieval involves shipping of all the multimedia objects to the user's query site because this strategy supports efficient access for synchronization during the presentation of the result by the CMUI. We introduced the data allocation problem and performed simulated studies regarding effectiveness of different data allocation algorithms in [10].

The rest of the paper is organized as follows: Section 2 further elaborates the data allocation problem. Section 3 includes the algorithms proposed in this paper. The experimental environment for empirical evaluations for these algorithms is described in Section 4, the results are presented in Section 5, and Section 6 concludes this paper.

2 The Data Allocation Problem
In this section, we describe concisely the inputs to the data allocation problem addressed in this paper. These inputs characterize the underlying distributed multimedia database system and help in formulating the problem. We also introduce a number of notations throughout the paper which are summarized in Table 1.

Table 1: Definitions of Symbols.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q</td>
<td>The set of queries</td>
</tr>
<tr>
<td>Qij</td>
<td>The data transfer size of the xth query of MDO j to site i</td>
</tr>
<tr>
<td>W</td>
<td>The site data transfer size matrix</td>
</tr>
<tr>
<td>Qij</td>
<td>The query data transfer size of the xth query of MDO j to site i</td>
</tr>
<tr>
<td>M</td>
<td>The MDO dependency matrix</td>
</tr>
<tr>
<td>dij</td>
<td>The size of the data from MDO j to the site where MDO j is located</td>
</tr>
<tr>
<td>cij</td>
<td>The site data transfer size of MDO j to site i</td>
</tr>
<tr>
<td>n</td>
<td>The number of sites in the network</td>
</tr>
<tr>
<td>k</td>
<td>The number of MDOs in the distributed database system</td>
</tr>
<tr>
<td>m</td>
<td>The number of queries</td>
</tr>
<tr>
<td>A</td>
<td>The access frequencies matrix</td>
</tr>
<tr>
<td>ai</td>
<td>The access frequency of the xth query at site i</td>
</tr>
<tr>
<td>C</td>
<td>The unit transportation cost matrix of the network</td>
</tr>
<tr>
<td>ci</td>
<td>The unit transportation cost from site i to site j</td>
</tr>
<tr>
<td>l</td>
<td>The allocation limit vector of the sites</td>
</tr>
<tr>
<td>li</td>
<td>The allocation limit of site i</td>
</tr>
</tbody>
</table>

Consider a distributed multimedia database system with m sites, with each site having its own processing power, memory and a database system. Let Si be the name of site i where 0 ≤ i ≤ m – 1. The m sites of the distributed multimedia database system are connected by a communication network. A link between two sites Si and Sj (if it exists) has a positive integer cij associated with it giving the cost for a unit data transferred from site Si to site Sj. If two sites are not directly connected by a communication link then the cost for unit data transferred is given by the sum of the cost of links of a chosen path from site Si to site Sj. Let Q = {q1, q2, ..., qm} be the most important queries accounting for say more than 80% of the processing in the distributed multimedia database system. Each query qi can be executed from any site with a certain frequency. Let ai be the
frequency with which query \( q_i \) is executed from site \( S_i \). Let there be \( k \) MDOs (or database objects, or relations), named \( \{O_0, O_1, \ldots, O_{k-1}\} \).

Any query accessing both the single-media database fragments and multimedia objects can be split into two queries, one which accesses only single-media fragments and one which accesses only multimedia data (MDO). A multimedia query execution strategy can be:

1) **Move-Small**: If a binary operation involves two data MDOs located at two different sites then ship the smaller data MDO to the site of the larger data MDO.

2) **Query-Site**: Ship all the data MDOs to the site of query origin and execute the query.

### 3 The Data Transfer Cost Model

There are two aspects of the data transfer cost incurred to process a query that need to be modeled. The first aspect is the unit data transfer cost from one site to another. This is modeled as minimum cost path and the corresponding path from one site to another. Let \( c_{ij} \) be cost of transporting a single unit of data from site \( S_i \) to site \( S_j \). In order to find the best allocation of a set of MDOs, it is enough to know the size of data from every MDO that is required from every site. Let \( r_{ij} \) be defined as the size of data of MDO \( O_i \) that needs to be transported to the site where \( q_i \) is initiated. Let there be a query \( q_i \) initiated from site \( S_i \), \( a_{ir} \) times in an unit time interval. And let \( q_i \) request MDO \( O_j \) and each request require \( r_{ij} \) amount of data transfer from the site where \( O_j \) is located.

Let \( u_{ij} \) give the amount of data needed to be transferred from the site where MDO \( O_j \) is allocated to the site \( S_i \) where the queries are initiated. That is,

\[
    u_{ij} = \sum_{x=0}^{n-1} a_{ix} \cdot r_{sj}
\]

The second type of data transfer cost corresponds to the deeper levels of the MDOs dependency graph. The data is transported from the site where one MDO is located to the site where the other MDO is located in order to perform binary operation involving two (or more) different MDOs. In this case, the amount of data of a MDO required by a site varies with the allocation of other MDOs. Let \( d_{ij} \) define the size of data from MDO \( O_i \) that needs to be transported to the site where \( O_j \) is located so as to execute some binary operation. Let the corresponding matrix, \( k \times k \), be \( D \). But this is dependent on the query that is to be processed.

Let \( \delta_{ij} \) be the data size of \( O_j \) needed to be transported to the site where \( O_i \) is located to process \( q_i \). Then the amount of data that needs to be transported from the site where \( O_j \) is located to the site where \( O_j \) is given by:

\[
    d_{ij} = \sum_{x=0}^{n-1} \sum_{y=0}^{m-1} \delta_{ij}
\]

Let \( site(O_i) \) denote the site where MDO \( O_i \) is located.

Then the total transportation cost, \( T \), is given by:

\[
    T = \sum_{j=0}^{k-1} \sum_{i=0}^{k-1} c_{i,site(O_i), j, site(O_j)} \cdot d_{ij} + \sum_{i=0}^{k-1} \sum_{j=0}^{k-1} c_{i, site(O_i), j, site(O_j)} \cdot u_{ij}
\]

where the first term gives the data transfer cost incurred to process the binary operations between the MDOs located at different sites, and the second term gives the data transfer cost incurred to transfer the results of the binary operations of MDOs to the site where the query is initiated. The objective in data allocation problem is to minimize \( T \) by altering the function \( site(O_i) \) (which maps a MDO to a site).

### 4 Proposed Data Allocation Algorithms

Developing an efficient solution to the data allocation problem highly depends on the query execution strategy employed by the distributed database system. This is because different query execution strategies have different MDO migration patterns. We develop solutions for the data allocation problem when **query-site** and **move-small** query execution strategies are respectively used by the distributed database management system and the multimedia data provider. The proposed algorithms are described as follows:

#### 4.1 The Genetic Algorithm

Genetic algorithms manipulate a population of potential solutions to an optimization problem [8], [9], [12], [18]. They operate on encoded representations of the solutions, equivalent to the genetic material of individuals in nature, and not directly on the solutions themselves. As in nature, the selection mechanism provides the necessary driving force for better solutions to survive. Each solution is associated with a fitness value that reflects how good it is, compared with other solutions in the population. The higher the fitness value of an individual, the higher the chance of survival in the subsequent generation. Recombination of genetic material in genetic algorithms is simulated through a crossover mechanism that exchanges portions between strings. Another operation, called mutation, causes sporadic and random alternation of the bits of strings.

In the proposed genetic algorithm for the data allocation problem, we encode the assignment of each MDO in a binary representation. For example, if an MDO is assigned to site 3, then its assignment value is 11. The assignment value of all the MDOs are concatenated to form a binary string. Each binary string then represents a potential solution to the data allocation problem. The fitness of the string is simply the cost of the allocation. The selection mechanism is implemented as a simple proportionate selection scheme: a string with fitness \( f \) is allocated \( f / (f) \) offspring, where \( f \) is the average fitness value of the population. A string with a fitness value higher than the average is allocated more than one offspring, while a string with a fitness value lower than the average is allocated less than one offspring.

Pairs of strings are picked at random from the population to be subjected to crossover. We use the single point crossover. Assuming that \( l \) is the string length, the algorithm randomly chooses a crossover point that can assume values in the range 1 to \( l - 1 \). The portions of the two strings beyond this crossover point are exchanged to form two new strings. The crossover point may assume any of the \( l - 1 \) possible values with equal probability. Note that crossover is performed only when a randomly generated number in the range is greater than a pre-specified crossover rate \( p_c \) (also called the probability of crossover); otherwise, the strings remain unaltered. The value of
$p_c$ lies in the range from 0 to 1. In a large population, $p_c$ gives the fraction of strings actually crossed.

Mutation of a bit is to flip a bit. Just as $p_c$, controls the probability of a crossover, another parameter, $p_m$ (the mutation rate), gives the probability that a bit will be flipped. The bits of a string are independently mutated.

The genetic algorithm for the data allocation problem is briefly described below.

**Genetic Data Allocation Algorithm:**
(1) Initialize population. Each individual of the population is a concatenation of the binary representations of the initial random allocation of each MDO.
(2) Evaluate population.
(3) no_of_generation = 0
(4) WHILE no_of_generation < MAX_GENERATION DO
(5) Select individuals for next population.
(6) Perform crossover and mutation for the selected individuals.
(7) Evaluate population.
(8) no_of_generation += 1;
(9) ENDWHILE
(10) Determine final allocation by selecting the fittest individual.

For $a$, each gene is a single bit. A value of 1 indicates that the corresponding allocation space is allowed to be used for this chromosome. Otherwise, if the bit is 0, the space cannot be used. This reduces the effective allocation limit for each sites. For $b$, each gene is an integer which represents the priority of the MDO to be considered; a large value means high priority and a small value means low priority.

The first chromosome in the initial population is constructed from the information of the table of $w_j$ which represents the cost of allocating MDO $j$ to site $i$. For each MDO $j$, we calculate

$$\chi_j = \sum_{i=1}^{n} w_{ij}$$

The objective is to minimize the allocation cost by giving a higher priority to MDO with larger $\chi_j$ (a large value of $\chi_j$ means that this MDO will use more transmission time (cost)). Thus, we simply assign $\chi_j$ as the genes for each MDO position in part $a$ of the first chromosome in the initial population. All of the genes in part $a$ of the chromosomes are set to be 1 for the first chromosome in the initial population since this is the allocation limit of the original problem. For the remaining chromosomes in the initial population, the genes in $a$ are chosen randomly as 0 or 1. The genes in $b$ are perturbations of the first chromosome's corresponding genes in $b$. Notice that for genes in $a$, we must check whether the new effective allocation limit is enough for all MDO to be allocated. This can be done simply by counting the number of 1's in $a$ and by checking that this sum is greater than or equal to the total number of MDOs $n$.

For each chromosome, we find a solution by allocating MDO $j$ with the highest priority to the site $i$ such that $w_{ij}$ is smallest for all $w_{ij}, 1 < k < m$. If the effective allocation limit embedded in the genes in part $a$ of the chromosome for that site is exceeded (the site is already saturated), we allocate this MDO to the site with the next smallest value of $w_{ij}$ for all $w_{ij}, 1 < l < m, l \neq k$. We continue the process for the next MDO with the highest priority among MDOs not yet allocated.

For each chromosome, the cost function is the total transmission cost after allocating all the MDOs to some site using the mapping heuristic. The fitness value for the chromosome is calculated as

$$f(i) = \frac{(MaxCost - Cost(i))^{\tau}}{\sum_{i=1}^{N_p} (MaxCost - Cost(i))^{\tau}}$$

where $N_p$ is population size, $MaxCost$ is the maximum cost among the chromosomes in the population, $\tau$ is the convergence
factor used for controlling the rate of convergence.

The genetic operators selection, crossover, and mutation are applied. Selection means proportionately selecting the chromosomes in the population according to their fitness values. Crossover is cutting two chromosomes at the same position and exchanging the genes after the point of cutting. Mutation is choosing a gene in a chromosome and reset its value. For genes in a, simply set the value to either 1 or 0. For genes in b, perturbing the value in the gene by adding a randomly chosen value from -\eta to \eta (\eta and \mu are set as the maximal value of b-genes divided by 4 in this chromosome). The time complexity of the problem-based simulated evolution algorithm, like GA, is \(O(GP(k^2 + km))\), where \(G\) is the number of generations and \(P\) is the population size.

4.3 The Mean Field Annealing Algorithm

The mean field annealing (MFA) technique [3], [4], combines the collective computation property of the famous Hopfield Neural Network (HNN) with the annealing notion of simulated annealing (SA) [20]. The MFA algorithm is derived from an analogy to the Ising spin model which is used to estimate the state of a system of particles or spins in thermal equilibrium. In the Ising spin model, the energy of a system with \(S\) spins has the following form:

\[
H(s) = \frac{1}{2} \sum_{k=1}^{k} \sum_{l=1}^{l} \beta_{kl}s_{k}s_{l} + \sum_{k=1}^{k} h_{k}s_{k}
\]

where \(\beta_{kl}\) represents the level of interaction between spins \(k\) and \(l\), and \(s_{k} \in \{1, 0\}\) is the value of spin \(k\). It is assumed that \(\beta_{kl} = \beta_{lk}\) and \(\beta_{k} = 0\) for \(1 \leq k, l \leq S\). At the thermal equilibrium, spin average \(\langle s_k \rangle\) of spin \(k\) can be calculated using Boltzmann distribution as follows:

\[
\langle s_k \rangle = \frac{1}{1 + e^{\frac{-H(s)}{T}}}
\]

where \(\Phi(s) = \frac{H(s)}{T}\) represents the mean field acting on spin \(k\), where the energy average \(\langle H(s) \rangle\) of the system is:

\[
\langle H(s) \rangle = \sum_{k=1}^{k} \sum_{l=1}^{l} \beta_{kl}\langle s_{k} \rangle\langle s_{l} \rangle + \sum_{k=1}^{k} h_{k}\langle s_{k} \rangle
\]

The complexity of computing \(\Phi_k\) using the above equation is exponential. However, for large number of spins, the mean field approximation can be used to compute the energy average:

\[
\langle H(s) \rangle = \frac{1}{2} \sum_{k=1}^{k} \sum_{l=1}^{l} \beta_{kl}\langle s_{k} \rangle + \sum_{k=1}^{k} h_{k}\langle s_{k} \rangle
\]

Hence \(\langle H(s) \rangle\) is linear in \(\langle s_k \rangle\), the mean field \(\Phi_k\) can be computed using the equation:

\[
\Phi_k = \langle H(s) \rangle |_{k=0} - \langle H(s) \rangle |_{k=1} = \frac{\partial \langle H(s) \rangle}{\partial s_k} = \left\{ \sum_{l \neq k} \beta_{kl}\langle s_l \rangle + h_k \right\}
\]

Thus, the complexity of computing \(\Phi_k\) reduces to \(O(S)\).

At each temperature, starting with initial spin averages, the mean field \(\Phi_k\) acting on a randomly selected spin is computed. Then the spin average is updated. This process is repeated for a random sequence of spins until the system is stabilized for the current temperature.

We formulate the data allocation problem as MFA in the following manner. A spin matrix \(s_{ij}\) is used to encode the allocation of the data MDOs to sites. The matrix consists of \(k\) rows and \(m\) columns, representing \(k\) MDOs and \(m\) sites, respectively. A value of 1 in each entry indicates the MDO is allocated to the corresponding site. For example, if \(s_{ij} = 1\), then MDO \(i\) is allocated to site \(j\). A valid allocation is one in which each row of the spin matrix has exactly a single 1. Each spin variable is a continuous variable in the range \([0, 1]\). Spin values converge to either 1 or 0 at the fixed point. Given this formulation, the energy function (i.e., the data transfer cost function) for the data allocation problem can be formalized below.

\[
E(s) = \sum_{i=0}^{k-1} \sum_{j=0}^{m-1} \sum_{i' \neq i} \sum_{j' = 0}^{m-1} c_{ij} d_{ij} s_{ij} + \sum_{j=0}^{m-1} t_{ij} s_{ij}
\]

Using the mean field approximation, the expression for the mean field \(\Phi_{ij}\) experienced by spin \(s_{ij}\) is:

\[
\Phi_{ij} = \frac{\partial E(s)}{\partial s_{ij}} = -\sum_{i' \neq i} \sum_{j' = 0}^{m-1} c_{ij} d_{ij} s_{ij} - t_{ij}
\]

In a feasible allocation, each MDO should be allocated to exclusively one site. Thus, the sum of the spins across each row of the matrix should equal unity. This constraint can be explicitly handled while updating by normalizing each spin \(s_{ij}\) as:

\[
s_{ij} = \frac{e^{	heta T}}{\sum_{i'=0}^{k-1} e^{	heta T}}
\]

Given the above formulation, the MFA algorithm to solving the data allocation problem can be briefly formalized below.

**MFA Data Allocation Algorithm:**

1. Get the initial temperature \(T_0\), set \(T = T_0\).
2. Initialize the spin averages \(s = [s_{00}, s_{01}, \ldots, s_{k, m-1}]\), each \(s_{ij}\) is initialized as a random number between 0 and 1.
3. WHILE temperature \(T\) is in the cooling range DO
4. WHILE \(E\) is decreasing DO
5. Select a MDO \(i\) at random.
6. Compute the mean field of the spins at the \(i\)-th row, i.e., \(\Phi_{ij}, \forall j\).
7. Compute the summation \(\sum_{i'=0}^{k-1} e^{	heta T}\).
8. Compute the new spin values at the \(i\)-th row.
9. Compute the energy change due to these updates.
10. ENDWHILE
11. Update the temperature \(T\) according to the cooling schedule.
12. ENDWHILE
13. Determine the final allocation by allocating each MDO to the site with the largest spin value. If the final allocation is not feasible, then consider each over-allocated site to migrate the MDOs to other sites so that the increase in cost is the minimum.

Note that the last step of the MFA algorithm is necessary because we do not explicitly check for feasibility in the search process, which can then explore a broader regions in the search space. However, we found that this adjustment of the final allocation were seldom invoked as the allocation limits were usually very loose. The time complexity of the MFA algorithm is \(O(BK(k^2 + km))\).

5 Experimental Setup

In this section, we present the experimental setup for the
emirical evaluation of 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 Environment

Empirical evaluation of the algorithms was done by implementing a prototype loosely-coupled distributed multimedia database system on a cluster of SUN workstations with single media relational data stored in the distributed SYBASE database system. Each of the workstations has an application system consisting of a client CMUI process, and a multimedia database server. The multimedia server extracts the data from the multimedia files and the SYBASE database system, whereas the CMUI process accepts the users queries and presents the results. The MDOs in the distributed multimedia database system are allocated based on the allocation schemes generated by the algorithms presented in the previous section.

5.2 Relations

The distributed multimedia database system consists of the following relations, wherein, the ImageID, VideoID and AudioID attributes of Product_Media relation point to the files containing the corresponding image file, video file and audio file, respectively.

O1: Product_Category(CategoryID, Description)
O2: Product_Type(TypeID, CategoryID, Description)
O3: Media_Info(MediaID, MediaType, IP_addr, Filename, Path, Size)
O4: Company(CompanyID, Name, Address, Phone)
O5: Product_general(ProductID, CompanyID, TypeID, Name, Brandname, Price, Description)
O6: Product_media(ProductID, ImageID, VideoID, AudioID)

5.3 Queries

We illustrate only one of the ten queries considered for the experimentation due to lack of space. SQL is used as the query language for expressing the queries. Each of the queries access some information from the distributed SYBASE database and some information from the MDOs. The query 1 accesses all the office products and presents the images of the office products, a video clip about the product, and an audio clip describing the product.

Query 1 - product_category: = 'Office Products'

```sql
SELECT ProductID, CompanyID, TypeID, Name, Brandname, Price, Description
FROM Product_general
WHERE Description = 'Office Products'
```

The network cost between two sites is calculated as the average of two costs from A to B and from B to A and is given in Table 3. Network cost is in terms of time in ms required for transferring one Kbyte.

6 Experimental Results

In these experiments we allocate all the MDOs to either one of the two SYBASE servers, namely, server 1 and server 0.

6.1 Evaluation of Data Allocation Algorithms

Each query was executed a number of times with the set frequencies and the total time to execute the queries was calculated. Table 2 shows the result of data allocation generated by the mean field annealing algorithm. Each query is executed five time from each site and average execution time is calculated. Table 3 shows the average query execution time times the frequency with which the query executed at each site. The total.

Table 2: Allocation scheme generated by the MFA algorithm.

<table>
<thead>
<tr>
<th>MDO</th>
<th>O1</th>
<th>O2</th>
<th>O3</th>
<th>O4</th>
<th>O5</th>
<th>O6</th>
</tr>
</thead>
<tbody>
<tr>
<td>O7</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 3: Average execution time x query access frequency (MFA).

<table>
<thead>
<tr>
<th>Query Site</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>17233.69</td>
<td>40200.11</td>
<td>29892.40</td>
<td>19651.50</td>
<td>41178.08</td>
</tr>
<tr>
<td>2</td>
<td>39343.76</td>
<td>9956.60</td>
<td>21352.00</td>
<td>62863.92</td>
<td>19834.92</td>
</tr>
<tr>
<td>3</td>
<td>28076.64</td>
<td>18171.44</td>
<td>14508.29</td>
<td>92879.00</td>
<td>9518.84</td>
</tr>
<tr>
<td>4</td>
<td>19191.66</td>
<td>12047.52</td>
<td>27267.99</td>
<td>9314.75</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>60039.09</td>
<td>61240.44</td>
<td>0</td>
<td>69871.97</td>
<td>5112.15</td>
</tr>
<tr>
<td>6</td>
<td>20939.26</td>
<td>8320.89</td>
<td>9798.00</td>
<td>3058.38</td>
<td>9046.53</td>
</tr>
<tr>
<td>7</td>
<td>21717.27</td>
<td>2645.47</td>
<td>12720.25</td>
<td>0</td>
<td>4661.70</td>
</tr>
<tr>
<td>8</td>
<td>18611.81</td>
<td>8465.94</td>
<td>0</td>
<td>10710.28</td>
<td>5287.46</td>
</tr>
<tr>
<td>9</td>
<td>3317.73</td>
<td>0</td>
<td>7591.36</td>
<td>21326.86</td>
<td>3281.39</td>
</tr>
<tr>
<td>10</td>
<td>2571.32</td>
<td>20106.96</td>
<td>9774.56</td>
<td>14608.65</td>
<td>7088.93</td>
</tr>
</tbody>
</table>

Total: 241145.14 181188.37 132200.65 200995.31 180909.00

Table 4: Allocation scheme generated by the GA,.

<table>
<thead>
<tr>
<th>MDO</th>
<th>O1</th>
<th>O2</th>
<th>O3</th>
<th>O4</th>
<th>O5</th>
<th>O6</th>
</tr>
</thead>
<tbody>
<tr>
<td>O7</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 5: Average query execution time x query access frequency (GA).

<table>
<thead>
<tr>
<th>Query Site</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>18313.27</td>
<td>40646.64</td>
<td>29608.80</td>
<td>17855.49</td>
<td>40758.54</td>
</tr>
<tr>
<td>2</td>
<td>39436.24</td>
<td>9915.68</td>
<td>21431.96</td>
<td>40626.16</td>
<td>20515.88</td>
</tr>
<tr>
<td>3</td>
<td>27536.46</td>
<td>17512.32</td>
<td>13799.28</td>
<td>9536.46</td>
<td>9046.06</td>
</tr>
<tr>
<td>4</td>
<td>19675.56</td>
<td>11632.84</td>
<td>28429.85</td>
<td>8756.85</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>42783.24</td>
<td>60640.50</td>
<td>0</td>
<td>70980.42</td>
<td>50517.35</td>
</tr>
<tr>
<td>6</td>
<td>20608.77</td>
<td>9506.17</td>
<td>8750.82</td>
<td>2778.88</td>
<td>8750.19</td>
</tr>
<tr>
<td>7</td>
<td>24372.81</td>
<td>2341.66</td>
<td>12209.70</td>
<td>0</td>
<td>4059.00</td>
</tr>
<tr>
<td>8</td>
<td>21181.68</td>
<td>8588.71</td>
<td>0</td>
<td>10612.20</td>
<td>3584.10</td>
</tr>
<tr>
<td>9</td>
<td>33430.41</td>
<td>0</td>
<td>7726.28</td>
<td>22626.56</td>
<td>3400.33</td>
</tr>
<tr>
<td>10</td>
<td>2521.07</td>
<td>18625.56</td>
<td>9566.82</td>
<td>12144.45</td>
<td>8843.98</td>
</tr>
</tbody>
</table>

Total: 249256.41 199340.04 313655.01 193741.57 180727.23

Table 6: Allocation scheme generated by the SE algorithm.

<table>
<thead>
<tr>
<th>MDO</th>
<th>O1</th>
<th>O2</th>
<th>O3</th>
<th>O4</th>
<th>O5</th>
<th>O6</th>
</tr>
</thead>
<tbody>
<tr>
<td>O7</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 7: Average execution time x query access frequency (SE).

<table>
<thead>
<tr>
<th>Query Site</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>17561.84</td>
<td>40925.01</td>
<td>29599.05</td>
<td>13799.14</td>
<td>41185.80</td>
</tr>
<tr>
<td>2</td>
<td>44144.48</td>
<td>9595.64</td>
<td>20152.64</td>
<td>39344.56</td>
<td>20471.28</td>
</tr>
<tr>
<td>3</td>
<td>27846.24</td>
<td>17494.49</td>
<td>14278.14</td>
<td>9039.00</td>
<td>9039.36</td>
</tr>
<tr>
<td>4</td>
<td>17266.50</td>
<td>11055.48</td>
<td>26991.90</td>
<td>8658.55</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>39680.24</td>
<td>60764.93</td>
<td>0</td>
<td>69334.58</td>
<td>9493.65</td>
</tr>
<tr>
<td>6</td>
<td>19597.97</td>
<td>8236.77</td>
<td>8625.99</td>
<td>2814.28</td>
<td>8514.87</td>
</tr>
<tr>
<td>7</td>
<td>23900.28</td>
<td>2226.06</td>
<td>12509.95</td>
<td>0</td>
<td>4708.74</td>
</tr>
<tr>
<td>8</td>
<td>18957.77</td>
<td>8311.26</td>
<td>0</td>
<td>10007.48</td>
<td>5929.54</td>
</tr>
<tr>
<td>9</td>
<td>32288.29</td>
<td>7260.86</td>
<td>22724.42</td>
<td>32721.18</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>2460.30</td>
<td>12070.28</td>
<td>9368.02</td>
<td>12425.10</td>
<td>7339.77</td>
</tr>
</tbody>
</table>

Total: 242595.01 175379.04 128297.45 192169.11 778845.19
query time cost in this case is 936209. Table 4 and Table 5 show the results for genetic algorithm. The total query time cost is 936700. Table 6 and Table 7 for simulated evolution (total query time cost is 918950). These results show the difference in the allocation schema generated by different algorithms and the total cost of processing all the queries.

6.2 Evaluation of Query Execution Time

First, we discuss the query time of different allocation algorithms at different sites as shown in Tables 2 to 9. In order to further evaluate the results more clearly, each of query execution time value is divided by the average of that of values among different allocation algorithms. Thus algorithm which consistently gives a value close to 1.0 for most of the queries is a better algorithm. Otherwise, it implies that the results generated by the algorithm provide erratic performance in executing the queries. From the experiments (not included due to lack of space), we found that the query time is quite fluctuating for some algorithms like the mean field annealing algorithm. In general, however, the simulated evolution exhibits the best performance among all the algorithms. Moreover, simulated evolution tends to give an optimal solution when compared to exhaustive search solution.

6.3 Total Running Time of Data Allocation Algorithms

First, we present the total running time for executing the four algorithms as well as an exhaustive search in Table 8. It can be seen that simulated evolution gives the least total cost but its running time is the longest among the three algorithms. Thus, there is a trade-off between the algorithm and quality of the solution. Though mean field annealing algorithm is fast, its solution quality is much worse than simulated evolution.

Table 8: The total transfer costs and running times of different allocation algorithms.

<table>
<thead>
<tr>
<th>Allocation Algorithms</th>
<th>Total Cost</th>
<th>Running Time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean Field Annealing</td>
<td>97851967</td>
<td>33.99</td>
</tr>
<tr>
<td>Genetic Algorithm</td>
<td>100350379</td>
<td>112.32</td>
</tr>
<tr>
<td>Simulated Evolution</td>
<td>86731127</td>
<td>351.01</td>
</tr>
<tr>
<td>Exhaustive Search</td>
<td>86731127</td>
<td>1034.15</td>
</tr>
</tbody>
</table>

7 Conclusions

In this paper, we proposed various algorithms for the data allocation problem in distributed multimedia database systems. We developed an experimental loosely coupled distributed multimedia database system to perform empirical evaluations of the proposed algorithms. The experimental evaluation involved studying data allocation algorithms based on mean field annealing, genetic algorithm, and simulated evolution. Further, it involved implementing these algorithms, allocating data based on the results of these algorithms, and measuring the actual time to execute queries from different sites with set frequencies. Finally, the results collected are evaluated to derive conclusions regarding the utility of these four algorithms. The best allocation algorithm was found to be simulated evolution because it gives allocation with the least total query execution time cost, and also tends to provide an optimal solution when compared to the exhaustive solution.

References