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 indroduced 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) enables the users to specify queries accessing the distributed multimedia database system and 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.

Symbol	Meaning
O_j	The jth MDO
S_i	The ith site
Q	The set of queries
q_x	The xth query
m	The number of sites in the network
k	The number of MDOs in the distributed database system
n	The number of queries
A	The access frequencies matrix
a_{ix}	The access frequency of the xth query at site i
C	The unit transportation cost matrix of the network
$c_{ii'}$	The unit transportation cost from site i to site i'
l	The allocation limit vector of the sites
l_i	The allocation limit of site i
R	The query data transfer size matrix
r_{xj}	The query data transfer size of the xth query of MDO j
U	The site data transfer size matrix
$u_{jj'}$	The site data transfer size of MDO j to site j'
D	The MDO dependency matrix
d_{ij}	The size of the data from MDO i to the site where MDO j is located
t	The total data transfer cost

Consider a distributed multimedia database system with m sites, with each site having its own processing power, memory and a database system. Let S_i be the name of site i where $0 \le i \le m-1$. The m sites of the distributed multimedia database system are connected by a communication network. A link between two sites S_i and $S_{i'}$ (if it exists) has a positive integer $c_{ii'}$ associated with it giving the cost for a unit data transferred from site S_i to site $S_{i'}$. 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 S_i to site $S_{i'}$. Let $Q = \{q_0, q_1, \ldots, q_{n-1}\}$ be the most important queries accounting for say more than 80% of the processing in the distributed multimedia database system. Each query q_x can be executed from any site with a certain frequency. Let a_{ix} be the

frequency with which query q_x 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} is 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_{xj} be defined as the size of data of MDO O_j that needs to be transported to the site where q_x is initiated. Let there be a query q_x initiated from site S_i, a_{ix} times in an unit time interval. And let q_x request MDO O_j and each request require r_{xj} 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,

$$u_{ij} = \sum_{x=0}^{n-1} a_{ix} \cdot r_{xj}$$

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 $d_{jj'}$ define the size of data from MDO O_j 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 $\delta^x_{jj'}$ be the data size of O_j needed to be transported to the site where $O_{j'}$ is located to process q_x. A 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:

$$d_{jj'} = \sum_{x=0}^{n-1} \left(\sum_{i=0}^{m-1} a_{ix} \right) \delta^x_{jj'}$$

Let $site(O_j)$ denote the site where MDO O_j is located. Then the total transportation cost, T, is given by:

$$T = \sum_{j=0}^{k-1} \sum_{j'=0}^{k-1} c_{site(O_j), site(O_{j'})} \cdot d_{jj'} + \sum_{i=0}^{m-1} \sum_{j=0}^{k-1} u_{ij} \cdot c_{i, site(O_j)}$$

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 $site(O_j)$ (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 $f/\bar{f}$ offspring, where $\bar{f}$ 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 ++;
(9) ENDWHILE
(10) Determine final allocation by selecting the fittest individual. 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.

The time complexity of the GA algorithm is $O(GP(k^2 + km))$, where G is the number of generations and P is the population size.

4.2 The Simulated Evolution Algorithm

This variant of the traditional genetic algorithm is called *problem-based simulated evolution* [18], [19]. Simulated evolution is an optimization method based on an analogy with the natural selection process in the biological environments. In biological processes species adapt themselves better to the environment as they evolve from one generation to the next one. In this evolution process some of the bad characteristics of the old generation are eliminated and a new generation that is more suited to the environment is created.

The principal difference between genetic algorithms and evolutionary strategies is that genetic algorithms rely on crossover, a mechanism of probabilistic and useful exchange of information among solutions, to locate better solutions, while evolutionary strategies use mutation as the primary search mechanism. Furthermore, in the proposed scheme the chromosomal representation is based on problem data, and solution is generated by applying a fast decoding heuristic (mapping heuristic) in order to map from problem domain to solution domain. The generic problem-based simulated evolution is as below.

Simulated Evolution Data Allocation Algorithm:

(1) Construct the first chromosome based on the problem data and perturb this chromosome to generate an initial population.
(2) Use the mapping heuristic to generate a solution for each chromosome.
(3) Evaluate the solutions obtained.
(4) no_of_generation = 0
(5) WHILE no_of_generation < MAX_GENERATION DO
(6) Select chromosomes for next population.
(7) Perform crossover and mutation for these set of chromosomes.
(8) Use the mapping heuristic to generate a solution for each chromosome.
(9) Evaluate the solutions obtained.
(10) no_of_generation = no_of_generation + 1.
(11) ENDWHILE
(12) Output the best solution found so far.

The chromosome structure is as follows:

a genes	b genes

where number of genes in a = total allocation limit, and number of genes in b = total number of MDOs.

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_{ij} which represents the cost of allocating MDO j to site i. For each MDO j, we calculate

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

The objective is to minimize the allocation cost by giving a higher priority to MDO with larger x_j (a large value of x_j means that this MDO will use more transmission time (cost)). Thus, we simply assign x_j as the genes for each MDO position in part b 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_{kj}, $1 < k < m$. 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_{lj}, $1 < l < m$, $l \neq k$. 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, τ 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 μ (η and μ 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 another well-known optimization algorithm known as the 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:

$$H(s) = \frac{1}{2}\sum_{k=1}^{S}\sum_{l\neq k}\beta_{kl}s_k s_l + \sum_{k=1,}^{S} h_k s_k$$

where β_{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 $\beta_{kk} = 0$ for $1 \le k, l \le S$. At the thermal equilibrium, spin average $\langle s_k \rangle$ of spin k can be calculated using Boltzmann distribution as follows:

$$\langle s_k \rangle = \frac{1}{1+e^{-\Phi_k/T}}$$

where $\Phi_k = \langle H(s)\rangle|_{s_k=0} - \langle H(s)\rangle|_{s_k=1}$ represents the *mean field* acting on spin k, where the energy average $\langle H(s)\rangle$ of the system is:

$$\langle H(s)\rangle = \sum_{k=1}^{S}\sum_{l\neq k}\beta_{kl}\langle s_k s_l\rangle + \sum_{k=1}^{S} h_k\langle s_k\rangle$$

The complexity of computing Φ_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 :

$$\langle H(s)\rangle = \frac{1}{2}\sum_{k=1}^{S}\sum_{l\neq k}\beta_{kl}\langle s_k\rangle\langle s_l\rangle + \sum_{k=1}^{S} h_k\langle s_k\rangle$$

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

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

Thus, the complexity of computing Φ_k reduces to $O(S)$

At each temperature, starting with initial spin averages, the mean field Φ_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.

$$E(s) = \sum_{i=0}^{k-1}\sum_{i'=0}^{k-1}\sum_{j=0}^{m-1}\sum_{j'=0}^{m-1} c_{jj'}d_{ii'}s_{ij}s_{i'j'} + \sum_{j=0}^{m-1}\sum_{i=0}^{k-1} u_{ji}s_{ij}$$

Using the mean field approximation, the expression for the mean field Φ_{ij} experienced by spin s_{ij} is:

$$\Phi_{ij} = -\frac{\partial E(s)}{\partial s_{ij}} = -\sum_{i'\neq i}^{k-1}\sum_{j'\neq j}^{m-1} c_{jj'}d_{ii'}s_{i'j'} - u_{ji}$$

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:

$$s_{ij} = \frac{e^{\Phi_{ij}/T}}{\sum_{j'=0}^{m-1} e^{\Phi_{ij'}/T}}$$

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 $s = [s_{00}, s_{01}, \ldots, s_{k-1,m-1}]$, 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 $\sum_{j'=0}^{m-1} e^{\Phi_{ij'}/T}$.
(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

empirical 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. A set of queries are initiated according to the frequencies from various sites and the time taken to execute the queries is measured. There are two sites SYBASE10_C1 and SYBASE_CS_SVR4 (site 0 and site 1) that store the distributed relational database, and a cluster of workstations, namely, csl3su1, csl3su10, csl3su30 and csl3su40 (i.e., site 2 to site 6, respectively) store multimedia data files.

5.2 Relations

The distributed multimedia database system consists of 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(MediaInforID, 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(Product_ID, ImageID, VideoID, AudiID)

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 :   $ID = ImageID, VideoID, AudioID
        select company_name, company_address,
         company_phone from Media_Info
        where MediaInfo_ID in (
                select $ID from Product
                where TypeID in (
                        select Type_ID from Product_Type
                        where CategoryID in (
                                select Category_ID from
                                Product_Category
                                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 on eof 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.

MDO	O1	O2	O3	O4	O5	O6
Allocated server	1	0	1	1	0	1

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

Query \ Site	1	2	3	4	5
1	17283.69	40290.11	28932.40	18961.50	41758.08
2	39343.76	9996.60	21352.00	42863.92	19834.92
3	28076.64	18717.44	14398.29	9679.00	9518.84
4	19191.66	12074.52	27704.79	9114.75	0
5	40039.00	61240.44	0	69871.97	51121.15
6	20938.26	8320.89	9789.00	3058.38	9046.53
7	21771.27	2645.47	12720.25	0	4461.70
8	18611.81	8465.94	0	10710.28	5287.46
9	33317.73	0	7529.36	21226.86	32811.39
10	2571.32	20106.96	9774.56	14608.65	7068.93
Total	241145.14	181858.37	132200.65	200095.31	180909.00

Table 4: Allocation scheme generated by the GA.

MDO	O1	O2	O3	O4	O5	O6
Allocated Server	1	1	1	0	0	1

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

Query \ Site	1	2	3	4	5
1	18213.27	40644.66	29693.80	17855.49	40758.34
2	39346.24	9915.68	21431.96	40626.16	20151.88
3	27356.46	17921.32	13799.28	9360.46	9040.06
4	19675.56	11832.84	28429.65	8756.85	0
5	42783.24	60463.50	0	70980.42	50617.35
6	20608.77	9367.11	8795.82	2778.88	8750.19
7	24372.81	2343.66	12269.70	0	4609.00
8	21118.58	8158.71	0	10612.20	5584.10
9	33430.41	0	7276.28	22626.66	34401.33
10	2321.07	18692.56	9968.52	12144.45	6814.98
Total	249226.41	179340.04	131665.01	195741.57	180727.23

Table 6: Allocation scheme generated by the SE algorithm.

MDO	O1	O2	O3	O4	O5	O6
Allocated Server	0	0	1	1	1	1

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

Query \ Site	1	2	3	4	5
1	17361.84	40925.01	29309.05	17389.14	41316.80
2	44144.48	9596.84	20152.64	39344.56	20471.28
3	27846.24	17439.44	14278.14	9039.00	9039.36
4	17266.50	11195.48	26991.90	8636.55	0
5	39680.24	60564.90	0	69334.58	49413.65
6	19597.97	8236.77	8625.99	2814.28	8514.87
7	23390.28	2262.06	12309.95	0	4705.74
8	18914.77	8311.26	0	10907.48	5292.54
9	33288.39	0	7260.86	22278.42	32751.18
10	2408.30	17207.28	9368.92	12425.10	7339.77
Total	243899.01	175739.04	128297.45	192169.11	178845.19

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 Exeeution 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.

Allocation Algorithms	Total Cost	Running Time (*ms*)
Mean Field Annealing	97851967	33.99
Genetic Algorithm	100196279	119.32
Simulated Evolution	86731127	351.01
Exhaustive Search	86731127	1034.15

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

[1] P.M.G. Apers, "Data Allocation in Distributed Database Systems," *ACM Trans. Database Sys.*, Sep. 1988, pp. 263-304.

[2] 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 (May 1990) p217-31.

[3] D.E. Van den Bout and T.K. Miller, "Graph Partitioning using Annealed Neural Networks," *IEEE Trans. Neural Networks,* vol. 1, no. 2, 1990, pp. 192-203.

[4] T. Bultan and C. Aykanat, "A New Mapping Heuristic Based on Mean Field Annealing," *J. Parallel and Distributed Computing*, 16, 1992, pp. 292-305.

[5] D. W. Cornell and P. S. Yu, "Site Assignment for Relations and Join Operations in the Distributed Transaction Processing Environment," Proc. *Int'l Conf. Data Eng., IEEE*, Feb. 1988.

[6] B. Gavish and H. Pirkul, "Computer and Database Location in Distributed Computer Systems," *IEEE Trans. Computers*, vol. C-35, no. 7, 1986, pp. 583-590.

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

[8] D.E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison- Wesley, Mass. 1989.

[9] S. Hurley, "Taskgraph Mapping using a Genetic Algorithm: A Comparison of Fitness Functions," *Parallel Computing,* 19, 1993, pp. 1313-1317.

[10] Y-K. Kwok, K. Karlapalem, I. Ahamad and N. M. Pun, "Design and Evaluation of Data Allocation Algorithms for Distributed Multimedia Database Systems", *IEEE Journal on Selected Areas in Comm.*, Vol 14, No. 7, pp 1332-1348, Sep., 1996.

[11] X.M. Lin, M.E. Orlowska and Y.C. Zhang, "Database Placement in Communication Networks for Minimizing the overall Transmission Cost," *Mathematical and Computer Modelling*, 19(1), Jan. 1994, pp. 7-19.

[12] S.W. Mahfoud and D.E. Goldberg, "Parallel Recombinative Simulated Annealing: A Genetic Algorithm," *Parallel Computing*, 21, 1995, pp. 1-28.

[13] T. Özsu and P. Valduriez, *Principles of Database Systems*, Prentice-Hall Inc., 1991.

[14] M.T. Özsu, D. Szafron, G. El-Medani, and C. Vittal, "An Object-Oriented Multimedia Database System for a News-on-Demand Application", to appear, Multimedia Sys., 1995.

[15] T.C. Rakow, E.J. Neuhold, and M. Lohr, "Multimedia Database Systems: The Notions and the Issues," to be published in *Tagungsband GI-Fachtagung Datenbanksystems* in Buro, Technik and Wissenschaft (BTW), Dresden Marz 1995, Springer Informatik Aktuell, Berlin, 1995.

[16] S. Ram and R.E. Marsten, "A Model for Database Allocation Incorporating a Concurrency Control Mechanism," *IEEE Trans. Knowledge and Data Eng.*, 3(3), Sep. 1991, pp. 389-395.

[17] P.I. Rivera-Vega, R. Varadarjan and S.B. Navathe, "Scheduling Data Redistribution in Distributed Databases," Proc. *Int'l Conf. Data Engineering, IEEE*, Feb. 1990.

[18] M. Srinivas and L.M Patnaik, "Genetic Algorithms: A Survey," *Computer,* vol. 27, no. 6, Jun. 1994, pp. 17-26.

[19] B. Wilson and S.B. Navathe, "An Analytical Framework for the Redesign of Distributed Databases," Proc. *6th Advanced Database Symposium, Tokyo, Japan*, 1986.

[20] L. Yong, K. Lishan and D.J. Evans, "The Annealing Evolution Algorithm as Function Optimizer," *Parallel Computing*, 21, 1995, pp. 389-400.