Available online at www.sciencedirect.com
SCIENCE DIRECT®
The Journal of Systems and Software 73 (2004) 467–480
The Journal of Systems and Software
www.elsevier.com/locate/jss

A simulated annealing approach for multimedia data placement

Evimaria Terzi, Athena Vakali *, Lefteris Angelis

Department of Informatics, Aristotle University, Box 114, 54124 Thessaloniki, Greece

Received 20 May 2002; received in revised form 11 April 2003; accepted 10 September 2003
Available online 8 January 2004

Abstract

Multimedia applications are characterized by their strong timing requirements and constraints and thus multimedia data storage is a critical issue in the overall system's performance and functionality. This paper describes multimedia data representation models that effectively guide data placement towards the improvement of the Quality of Presentation for the considered multimedia applications. The performance of both *constructive* placement and *iterative improvement* placement algorithms is evaluated and discussed. Emphasis is given on placement schemes which are based on the *simulated annealing* optimization algorithm. A placement policy, based on a self-improving version of the simulated annealing (SISA) algorithm is applied and evaluated. Performance of the placement policies is experimentally evaluated on a simulated tertiary storage subsystem. As proven by the experimentation, the proposed approach shows considerable gain in terms of seek and service times. The improvements of the proposed SISA approach are in the range of 40% when compared to random placement and at the range of 15–35% when compared to the typical simulated annealing algorithm, depending a lot on the initial configuration and the neighborhood search.

Keywords: Multimedia data storage; Tertiary storage subsystems; Simulated annealing; Self-improving processes

1. Introduction

Increasing popularity of multimedia applications has been followed by corresponding increases in users requirements. Multimedia data differ from conventional text data since they are characterized by: (a) large size and (b) timing requirements and constraints. Representation models for multimedia data with respect to their spatio-temporal requirements have been proposed and classified in Bertino and Ferrari (1998), Kwon et al. (1999), Chung (1979), Escobar-Molano et al. (1996), Ghandeharizadeh (1996) and Hirzalla et al. (1995). More specifically, in Bertino and Ferrari (1998) and Kwon et al. (1999) a classification of the representation models, based on the notion of time is presented and two categories have been identified: the *interval-based* and the *constraint-based* models. A number of different representation approaches have been introduced in Bertino and Ferrari (1998), Kwon et al. (1999) and Chung (1979), and in this case the models are classified into three different categories: *Graph Models*, *Petri-Net Models* and *Object-Oriented Models*. This classification mainly focuses on the conceptual data representation and the model to be chosen depends on the application requirements, the developer's conceptual view and the existing system features. Moreover, in Escobar-Molano et al. (1996) and Ghandeharizadeh (1996) video objects representation models are categorized into *Stream-Based Models* and *Structured Models* with respect to their physical requirements and from the perspective of the DataBase Management System. Furthermore, a new timeline model, which captures user's interactivity on a set of multimedia documents is proposed in Hirzalla et al. (1995).

Current software development trends favor the involvement of hypermedia documents in most of recent large scale applications, i.e. a navigational type of access and searching is introduced. The allocation of such hypermedia documents and their corresponding multimedia objects (from the end user response time perspective) over a communication network has been discussed in Ahmad et al. (1999). In that context, the proposed model is

* Corresponding author. Tel.: +30-2310-998415; fax: +30-2310-998419.
E-mail addresses: evimaria@csd.auth.gr (E. Terzi), avakali@csd.auth.gr (A. Vakali), lef@csd.auth.gr (L. Angelis).

0164-1212/$ - see front matter © 2003 Published by Elsevier Inc.
doi:10.1016/j.jss.2003.09.020

related to multimedia objects allocation towards effective browsing and navigation in distributed environments. Indexing and declustering schemes for object-oriented or interactive navigational queries are proposed in Chen and Sinha (2000) and Han et al. (1999). Analysis and comparisons of the proposed declustering schemes and performance studies have indicated that navigational based indexing and declustering can be beneficial for the responsiveness and interactivity of the multimedia applications.

Due to the large storage requirements of multimedia data, tertiary storage subsystems are a quite appropriate proposal for multimedia objects storage. Tertiary storage level media are rather inexpensive and despite their slow data transfer rates, they are used for large scale storage mainly due to their huge space capacities. Recent research efforts in tertiary storage has focused on improving their performance towards high Quality of Service (QoS) for multimedia applications. In Prabhakar et al. (1996) the current state of the art in tertiary storage systems is discussed, tertiary system types are classified and extensive performance results are provided. Research work in Chervenak (1994) and Christodoulakis et al. (1997) evaluates storage hierarchies and the usefulness of current tertiary storage systems. In Hillyer and Silberschatz (1996) a serpentine tape drive is studied under model-driven simulation, while in Johnson and Miller (1998) detailed measurements of several tape drives and robotic storage libraries are presented, in order to provide better understanding on the issues related to integrating tertiary storage into a complete computer system. Issues related to data placement on the tertiary storage subsystem have also been studied. More specifically, in Christodoulakis et al. (1997) different data placement policies on various tape technologies and tape libraries have been implemented, while in Sesardi et al. (1994) optimal arrangements of cartridges and file-partitioning schemes are examined under a carousel type mass storage system. Furthermore, in Vakali and Manolopoulos (1998) data placement schemes are considered under three different models corresponding to three mid-range magnetic tape systems, while in Vakali and Terzi (2002) and Vakali et al. (2001) constructive and iterative improvement placement algorithms have been implemented for the placement of multimedia data on a tape-based storage subsystem.

The data placement problem is an optimization problem characterized by an objective cost function that has to be optimized (i.e. minimized or maximized depending on the application). Since optimal data placement is a particularly complicated problem which cannot be solved analytically, an algorithmic approach is necessary in search of the optimal solution. A wide class of heuristic algorithms for solving similar optimization problems is the class of *random search* algorithms. These are stochastic processes, which perform biased random walks in the space of the candidate solutions of a certain problem. They use random choice as a tool but in such a way that the search is guided towards a global optimal solution. They are highly efficient methods, easily applied on any optimization problem and restriction-free on the objective function (continuity, differentiation, etc.). Such methods for combinatorial optimization are described in Nahar et al. (1986). The most simple, yet powerful, random search technique is the simulated annealing (SA) algorithm. This method was initially presented in Metropolis et al. (1953) and received its name from the physical process called *annealing* which brings a solid to a state of minimum energy. Moreover, SA was proposed by Kirkpatrick et al. (1983) as a general-purpose optimization technique, suitable for solving many complex combinatorial problems. Since then, there was a vast amount of research work for applications of SA in various optimization problems (for example Hua et al., 1994), as well as for theoretical approaches to probabilistic mechanisms (see Aarts and van Laarhoven, 1989; Azencott, 1992).

This paper's SA-based algorithms are customized for our data placement problem, by adapting the approach presented in Angelis et al. (2001) for the solution of an optimization problem in statistical planning. The present work is an extension of author's previous research efforts as presented in Vakali and Terzi (2002), Vakali et al. (2001) and Angelis et al. (2001), and the paper's key contribution is summarized in the following points:

- The proposed multimedia data representation model captures both the users navigation (within an application) and the timing constraints (within each multimedia object) towards efficient multimedia data placement under a tertiary storage topology.
- The problem of data placement is dealt as an optimization problem, where an extension of the SA is employed. In addition to iterative improvement placement policies constructive placement algorithms are also implemented for comparison purposes.
- A self-improving process of the simulated annealing is introduced for determining multimedia data placement within the tertiary storage subsystem.
- The impact of the initial placement and of the perturbation function (i.e. the rules governing the search) in the overall performance of the SA-based algorithms are studied and the corresponding experimental results are discussed.

The structure of the remainder of the paper is organized as follows: in Section 2 the proposed representation model is introduced. The considered storage system is presented in Section 3 and the criteria for multimedia objects allocation are emphasized. Extensive discussions on the placement policies are given in Section 4, with particular emphasis on the simulated-annealing-based

techniques. The simulation model and its basic components are analyzed in Section 5, while in Section 6 experimentation is described and results are presented. Finally, future work topics are discussed in Section 7. Notice that this work does not include an extensive description of the data model and the intuition of the algorithmic techniques behind it, since these are defined and discussed in authors' earlier work (Vakali and Terzi, 2002; Vakali et al., 2001; Angelis et al., 2001).

2. Multimedia data representation

This part is a summary over the *Graph-Tree* multimedia data representation structure that have been proposed and used by the authors in their earlier works (Vakali and Terzi, 2002, 2001a; Vakali et al., 2001) dealing with the problem of effective multimedia data placement on storage devices based on users' behavior when they navigate a multimedia application. There are two main goals of the proposed representation model: (1) to capture the users access patterns and (2) to adequately convey information about the timing constraints within the multimedia data that will prove to be important for their hiccup-free presentation to the end user. In order these two requirements to be fulfilled, a two-level model is adopted. The *external level* describes the users' interaction with the multimedia application, while the *internal level* is a tree-based timeline model that describes the multimedia data itself. The most important parameters for multimedia data representation are summarized in Table 1.

- *External level*: This level captures the user's interactions as employed in a navigational environment. The main data structure to support this level is a browsing graph for describing the multimedia objects as visited by the user. The *Browsing Graph* is a directed graph $G = (V, E)$ where $V = \{1, 2, \ldots, M\}$ is a set of M nodes (corresponding to M multimedia objects) and E is a set of directed edges, corresponding to physical connection from one multimedia object (*MObj*) to another (Fig. 1). [1]
- *Internal level*: Each of the external level's graph nodes is further analyzed to its objects used to build the multimedia objects. A tree-based structure, similar to an index tree used for conventional database indexing is used as the basis for our representation model. The finest units that the nodes of this tree can be broken into are the physical objects that correspond to a specific format data type entity that also corresponds to a physical storage unit. [2] An indicative example of this structure, called *timeline tree*, is given in Fig. 2. As depicted in this figure, the proposed tree structure involves two types of nodes:
  - *Multi-nodes*: these are the internal nodes (represented by the rectangular shape) and they can be further analyzed since they consist of more than one physical objects need to be synchronized.
  - *Single-nodes*: these are the leaf nodes (represented by the circular shape) and they correspond to physical objects that need to be played for a specific continuous time unit.

[1] From now on we will use the terms multimedia object, node and the notation *MObj* interchangeably.

[2] The term physical object and the notation *PObj* will be used interchangeably in the text.

Table 1
Most important parameters for multimedia data representation

Parameter	Description
f	Vector of access frequencies
f_i	Access frequency of node i
G	Browsing graph
M	Number of multimedia objects
$MObj_i$	Multimedia object i
O	Number of physical objects
nr_{ix}	Number of object x playouts in node i
P	Transition matrix of the p_{ij} probabilities of accessing node j from node i at a single step
$PObj_i$	Physical object i
pop[x]	Popularity of physical object x

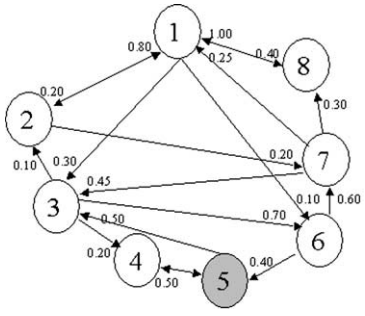

Fig. 1. The external browsing graph for a navigational multimedia application.

The proposed tree has the attributes of an index tree structure and the key attribute that is used for insertion, deletion and search operations is the starting play time of the multimedia or the physical object that corresponds to a multi or a single tree node. This time-based approach makes the tree a timeline multimedia representation model.

3. The storage approach

3.1. Criteria for multimedia objects allocation

Based on the adopted multimedia data representation, it is important to notice that at the external level,

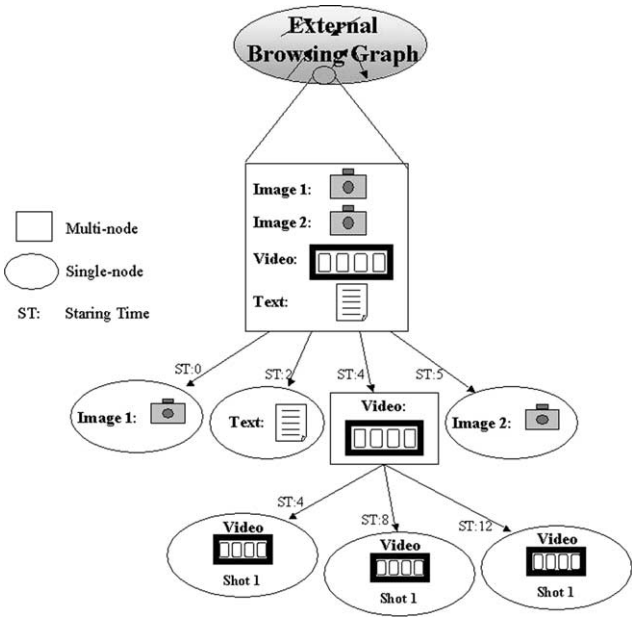

Fig. 2. The timeline tree structure.

we consider the so called multimedia nodes which are the actual multimedia objects. Then, the proposed browsing graph can be represented by a *transition matrix P* associated with the graph G, a $(M \times M)$ matrix of access or transition probabilities, where by $P(i,j) = p_{ij}$, $(i,j \in \{1,2,\ldots,M\})$ we denote the probability of accessing node j from node i in a single step. In order to capture the frequency of access of each node we define the access frequency:

Definition 1. The frequency of access of the ith $MObj$ can be found in the ith element of the access frequency vector f. The vector of access frequencies $f = (f_1,\ldots,f_M)$ for the M $MObj$s involved in a multimedia application can be evaluated by

$$f = \lim_{k\to\infty} P^k$$

where P is the graph's transition matrix.

Due to the multimedia objects high-capacity requirements, these objects are further analyzed to their (internal level) physical nodes which include the actual physical objects (to be stored on a storage medium). Therefore, an appropriate mapping between the (external level) multimedia objects and the actual physical locations, should be adopted. Each multimedia object corresponds to a number of physical objects and thus physical locations of the storage medium. These locations correspond to a number of KBs to MBs for each physical object. This mapping (of multimedia to physical objects) will be employed here based on a popularity criterion.

It should be noted that although Definition 1 provides the evaluation of the frequency of access (and thus the popularity of a multimedia object), it gives no explicit way of calculating the popularity of the physical objects that actually constitute our storage entities. This is done by the following definition, also stated in Vakali and Terzi (2002, 2001a) and Vakali et al. (2001).

Definition 2. The *popularity of a physical object* x, which belongs to k nodes in the Graph-Tree representation model, is defined by

$$\mathrm{pop}[x] = \sum_{i=1}^{k} f_i \times nr_{ix}$$

where f_i is the frequency of access (Definition 1) of the nodes containing x and nr_{ix} is the number of object x playouts in node i.

The popularity criterion is adopted to guide physical allocation in a way that the multimedia synchronization requirements are also met. More specifically, since the popularity metric involves the frequency of access value (which refers to the popularity of the multimedia object as a whole), it is expected that physical objects (included in a multimedia object) will have popularity values in a close range. Therefore they will be stored in close or parallel locations of the underlying storage topology.

Synchronization requirements will be met since if two (or more) physical objects are synchronized, they will have their f_i and nr_{ix} values (of the popularity formula) in close range. This is the case since they will be synchronized when they participate in common multimedia objects (f_i values) and their real-time play back will involve concurrent playouts (nr_{ix} values) per multimedia object.

3.2. The storage topology

The proposed data placement algorithms will be employed under a hierarchical (tertiary) storage subsystem, which will be the storage medium for the artificial multimedia data workloads used in the experimentation part.

Tertiary storage topologies are considered here since they are appropriate in cases where large volumes of data need to be stored (multimedia applications are definitely such as case, due to their high data capacities and requirements). As emphasized in Scheier (2003) current technology advancements continue to extend the usefulness of tape as the cost-effective primary mid-term machine-readable archive medium. Moreover the recent storage trends highlighted in Scheier (2003) (according

to customers, analysts and vendors) point that the demand for tape will increase, not diminish, because it's still the only high-capacity media portable and inexpensive enough for long-term, off-site archiving.

Additionally, tertiary storage medium has been proposed for multimedia data storage in many earlier research efforts such as Gemmel et al. (1995) and Triantafillou and Papadakis (1997, 2001) mainly due to its high-capacity/cost ratio. Indicative are the following examples. The cost of tertiary storage memory that is based on tapes is about \$0.004 per MB on average (according to Triantafillou and Papadakis, 2001). Of course the cost varies based on the performance that is required within a specific application. Higher performance, guaranteed for example by the Ampex DST812 Automated Tape Library is more expensive than lower performance gained with Ampex DST421 Automated Tape Library that is a more affordable solution towards mass storage. There is a variety of commercial tertiary storage tape libraries such as the systems' specifications found in the sites of Ampex, Quantum and IBM.

Thus tertiary storage subsystems are rather appropriate for multimedia applications with high storage requirements. On the other hand one may argue that such a storage solution involves delays due to the data elevation among the tertiary and the main memory levels. In order to overcome this impediment we adopt the approach used in Triantafillou and Papadakis (1997) where the multimedia data playback is employed by considering the upper levels as cache to the tertiary storage. Thus, the popular physical objects are considered to reside in main memory for as long as needed and the service starts by the main memory, whereas the rest of the data stored in the tertiary storage will be elevated next. Such an approach will most appropriately match the multimedia application requirements with the storage system performance. Under this approach the latency time of the tertiary storage subsystems does not cause any problem in the overall evaluation. Additionally it should be noted that the methods proposed here are rather generic and can be appropriately applied to any kind of placement problem. The key point here is to see the optimization aspect of the problem rather than to strictly consider its specific application for tape libraries.

A detailed presentation of the tertiary storage subsystems has already been provided by the authors in Vakali and Terzi (2001b). Basically tertiary storage includes magnetic tapes, optical disk devices and some more recent technologies like optical tapes and holographic storage and recent technological advances have increased the interest of system designers in adopting the usage of such systems. The *Robotic Tape Libraries* are widely adopted tertiary storage subsystems that come in various types and configurations. They hold large number of cartridges that can be loaded by robot arms into a collection of magnetic tape drives and they can be classified into *Large Libraries*, *Carousel Devices* and *Stacker Devices* depending on the number of tapes they can hold and the mechanism used to load a tape into a drive.

The tertiary storage tape library that has been considered for our simulation studies has one robot arm, which is capable of moving between any tape stored in the library and the tapes are assigned at drives.

Mainly there are two types of magnetic tapes that can be included in magnetic libraries:

- *Tapes that rewind to the PBOT*: These are tapes that always rewind to their beginning before being ejected.
- *Tapes that rewind to the nearest zone*: These tapes enable tape rewinding to the nearest zone as opposed to those require to be rewound to their physical beginning before being ejected.

Tapes can be thought of as linear areas and each such linear area is divided into a certain number of fixed-size *segments*, which are the smallest accessible parts of the tape. *Sections* consist of a number of consequent segments, while *tracks* consist of a number of consequent sections. The number of segments that will be allocated to a data object obviously depends on the object's and the segment's size.

The two most common performance metrics, for our simulation study are: the *mean service time* and the *mean seek time*, required in order to evaluate the requests servicing. These metrics are widely adopted within simulated tertiary storage subsystems and have been estimated by using the following formulae:

$$Robot\_Arm\_Service\_Time = Pick\_Time + Move\_Time + Put\_Time$$

The time required by the robot during the service period of a single request is determined by the time it needs to pick the appropriate tape, to move towards the tape drive and finally put the tape into the drive. The drives, which are also assumed to be identical, perform the following operations: seek, rewind, read, write, load and eject. The seek and rewind operations for tapes are modeled as constant startup times followed by a constant transfer rate. Therefore, the tape access time is defined by the times involved in the servicing main actions:

$$Drive\_service\_time = Seek\_T + Rewind\_T + Transfer\_T + Load\_T + Eject\_T$$

Therefore, the total access time for a tape operation which includes a tape switch operation is defined as follows:

$$Total\_Service\_Time = Robot\_Arm\_Service\_Time + Drive\_service\_time$$

4. Multimedia data placement algorithms

The problem of proposing effective placement of multimedia data on a tertiary storage subsystem is "translated" to the problem of proposing a placement of N physical objects onto T tapes (each tape has Z zones) so that the imposed requests to be serviced in a way that minimizes the data seek and transfer time. Therefore, the data placement problem is an optimization problem. Various approaches have been proposed towards this research direction. In general, the data placement algorithms fall in two main categories: *Constructive placement* and *Iterative Improvement placement*.

For the constructive placement algorithms only a brief description will be given here since a more detailed one is presented in Vakali and Terzi (2002). The main characteristic of the constructive placement algorithms is that the data are placed on a location of the storage media based on a certain criterion (popularity in our case) without considering any other alternative placements, possibly slight modifications of the current one that may end up in better performance rates. The main constructive placement algorithms are the *Organ-pipe* and *Camel* placement algorithms (briefly described in Table 2). Since our storage system does not consist of a single tape but of a series of tapes the constructive placement algorithms are slightly modified for the considered tape library configuration (as depicted in Table 3).

It should be noted that the physical objects placement is guided by the popularity metric introduced in Section 2. The objects are placed in decreasing order of popularity, in order to preserve the synchronization requirements as explained in Section 3.1.

Table 2
Constructive placement algorithms

Organ-pipe placement
1. Place the most popular *PObj* on the middle zone ($Z/2$) of the tape
2. Allocate the next two popular *PObj*s on either side of the middle zone
3. Repeat step (2) until all objects are placed

Camel placement
1. Divide the tape into two consecutive tapes consisting of ($Z/2$) zones
2. Implement organ-pipe placement alternatively on the two consecutive tapes
3. Repeat step (2) until all objects are placed.

Table 3
The basic function of constructive placement algorithms

Constructive placement
pool[1 . . . O]: Pool of O Physical Objects
pop[1 . . . O]: Popularity of O Physical Objects
notstored[ ]: array-index to non-stored Physical Objects
PO[1 . . . O] ← Sorted pool[ ] array in decreasing pop[ ] values
$i \leftarrow 1$
$j \leftarrow 1$
while (there are still Unallocated Physical Objects and Free Tape Space)
STORE the ith *PObj* in (i mod T)th tape at the first available segment estimated by the *Organ-Pipe or (camel)* placement
if (the Physical Object is stored)
$i \leftarrow i + 1$
else if (space is not enough)
STORE the ith Physical Object on the *next* tape with adequate free space
if (the Physical Object is stored)
$i \leftarrow i + 1$
else //there is no tape with enough space available
notstored[j] ← i
$j \leftarrow j + 1$
$\imath \leftarrow i + 1$
endif
endif
endwhile

4.1. The simulated annealing approach

In the sequel the basics (and the considered alternatives) of the simulated annealing algorithm are described briefly. Notice that SA is a well-known iterative optimization technique and when considering it for the data placement problem, it is applied during the system design process since such iterative improvement techniques cannot be applied at real-time (due to their high time overhead). Thus SA's applicability is mostly employed in simulation studies aiming in deciding about optimal data placement (on an application-oriented basis). Once the placement has been determined by the annealing method, the data are placed on the storage media once and according to the placement indicated by the annealing technique.

4.1.1. Preliminaries

Here, we describe the principles governing the typical SA process, as suited for any optimization problem, and next we discuss the adaptation of the method to the specific data placement problem. More specifically, the SA involves the following actions:

- *Initialization*: Consider an arbitrary finite set Ω, called the solution space and an arbitrary function $C : \Omega \rightarrow R$, called the cost function. Our aim is to find a global minimum of C in Ω. Note that as the objective function represents cost, it is realistic to assume that $C(S) \geqslant 0$ for every $S \in \Omega$ and that the notion of optimization is equivalent to the minimization of the function. The SA algorithm generates a random sequence $S_n \in \Omega$ of feasible solutions which tends to converge as $n \rightarrow \infty$.

- *Searching*: The search begins from an arbitrarily chosen initial solution $S_{\text{ini}} \in \Omega$ and proceeds by generating and testing a sequence of solutions, each obtained as a random perturbation of the preceding one. The term "*perturbation*" means the move of the algorithm from any current solution S to a neighbor S_{try} through predefined legal operations. Obviously, the determination of perturbation rules is strongly dependent on the definition of "*neighborhood*" of any solution.

As search proceeds, any time a neighbor of the current solution S that leads to an improvement of the cost function (i.e. if $C(S_{\text{try}}) < C(S)$) is accepted as the basis for the next iteration. On the other hand, if the neighbor leads to a worsening (i.e. if $C(S_{\text{try}}) > C(S)$), then it is accepted with probability

$$p = \exp\left(-\frac{\Delta C}{Tmpr}\right)$$

where $\Delta C = C(S_{\text{try}}) - C(S)$ and *Tmpr* is an algorithm parameter called "*temperature*". Since the acceptance probability is decreasing as ΔC increases, perturbations leading to slightly worse solutions in a standard temperature are more likely to be accepted than perturbations which worsen significantly the current solution.

- *Towards the "best solution"*: The temperature is gradually reduced during the process, following a cooling schedule such that $Tmpr_n \to 0$. Thus, a certain worsening of the objective function is more likely to be accepted in early stages of the search than later when the temperature is "cooled". In this manner, the algorithm manages to escape from solutions, which are only locally optimal and tend to trap the search around them. The algorithm continuously keeps record of the best solution the random walk passed from and this is denoted by S_{best}. After a predefined number of perturbations in a standard temperature without improvement of S_{best}, the temperature is lowered according to a cooling schedule and the search continues. The most common cooling schedule, the so-called exponential, is defined by $Tmpr_{n+1} = Tmpr_n \cdot R$, where $0 < R < 1$ is a reduction factor, usually close to 1. The algorithm needs also a stopping criterion, so we usually agree to terminate the search when after a predefined large number I of iterations at a standard temperature no improvement of the current solution occurs. On termination of the process, the solution S_{best} is reported as optimal.

4.1.2. The typical SA and multimedia data placement

Here, we use the SA idea to a tape topology described in the previous section. Following the terminology of Section 3, the solution space of our data placement problem is the set of all possible permutations of the O_t physical objects allocated in each of the T tapes $(1 \geqslant t \geqslant T)$, $\{POBj_1, POBj_2, \ldots, POBj_{O_t}\}$. Notice that in our implementation the assignment of physical objects to the tapes is not part of the simulated annealing process. The SA algorithm is implemented within each tape of the tape library separately.

Therefore, for the problem of multimedia data placement the simulated annealing placement commences with an initial placement determined by a constructive placement procedure and is repeatedly modified in search for cost reduction. The main steps of the algorithm (as it is implemented for the specific data placement problem) are presented in Table 4 while Table 5 contains the main notation of the algorithm. In fact, the algorithm consists of the following main components:

- *The initial placement*: The initial placement of the physical objects in the storage system can follow either one of the constructive placement methods (camel, organ-pipe), or a random policy.
- *Neighborhood search*: For finding a neighbor of the current solution the physical objects within each tape

Table 4
General concept of the SA algorithm

Simulated annealing
$Tmpr_0$: Starting condition value
R: Reduction Value $(0 < R < 1)$
I: Number of Perturbations (if no improvement of S_{best} occurs)
$S_{\text{ini}} \leftarrow$ initial placement (*randomly selected*);
$Tmpr = Tmpr_0$; $S = S_{\text{ini}}$; $S_{\text{best}} = S_{\text{ini}}$;
while *STOP = FALSE* (*Conditional Loop*)
STOP = TRUE; *POINTER* = 1;
while *POINTER* $\leqslant I$ (*Perturbation Loop*)
$S_{\text{try}} = S$;
$cost = cost(S)$;
$S_{\text{try}} = perturb(S)$;
$\Delta cost = cost(S_{\text{try}}) - cost(S)$;
if $(\Delta cost < 0)$ **then**
$S = S_{\text{try}}$; (*accept the improvement*)
STOP = False;
else
$p = \mathrm{e}^{-\frac{\Delta cost}{Tmpr}}$;
$u \leftarrow$ *random number in* $U(0,1)$;
if $(u < p)$ **then**
$S = S_{\text{try}}$; (*accept the worsening*)
STOP = False;
endif
endif
if $(cost(S_{\text{try}}) < cost(S_{\text{best}}))$ **then**
POINTER = 1;
$S_{\text{best}} = S_{\text{try}}$;
else
POINTER + +;
endif
endwhile
if (*STOP* == *FALSE*)
$Tmpr = Tmpr \cdot R$;
endif
endwhile

Table 5
SA notation table

SA	Simulated annealing
S_{ini}	Initial placement
S	Current placement
S_{try}	Alternative to current placement obtained by a legal operation
S_{best}	The best finally accepted placement
$C()$	Cost function
$Tmpr_0$	Initial temperature value
$Tmpr$	Current value of the temperature
R	Reduction factor

Table 6
Most important parameters for the storage subsystem

Parameter	Description
s_j	Number of bytes to search from current head location towards requested location (j)
s_{rate}	Search rate
t_j	Number of bytes to transfer
t_{rate}	Transfer rate
T	Number of tapes of the tape library
Z	Number of zones of the tape

are rearranged Three alternatives have been implemented in our model (Fig. 3).

1. *Circular shift—CS*: Assume than O_t physical objects $(PObj_1, PObj_2, \ldots PObj_{O_t})$ are allocated to a specific tape. The neighborhood explorations moves $PObj_{O_t}$ to the position of the tape currently occupied by $PObj_1$ while for every $i \neq O_t$ $PObj_i$ is moved to the position of $PObj_{i+1}$.
2. *Single random selection—SRS*: A physical object is selected randomly and moved to a random location on the tape. The rest of the objects are moved and placed after the chosen object. Assume a tape on which O_t physical objects have been allocated. If the current placement of the objects is

$$(PObj_1, \ldots, PObj_{k-1}, PObj_k, PObj_{k+1}, \ldots, PObj_{O_t})$$

and we select object $PObj_k$ and move it to position 1, we get the permutation

$$(PObj_k, PObj_1, \ldots, PObj_{k-1}, PObj_{k+1}, \ldots, PO_{O_t})$$

3. $CS + SRS$ which is a combination of Circular shift and Single random selection perturbation procedures: at each iteration either Circular Shift or Random Selection perturbation is implemented. The two perturbation procedures have equal probability to be adopted at a specific iteration of the inner **repeat** loop.

- *The cost function*: In the data placement problem we considered, as the cost of each permutation S, the expected service time (EST) for retrieving all the O_t physical Objects stored on tape t. This cost is defined by the formula:

$$\mathrm{EST}(S) = \sum_{i=1}^{O_t} \sum_{j=1}^{O_t} \mathrm{pop}[i]\mathrm{pop}[j](s_j s_{rate} + t_j t_{rate})$$

where i, j refer to the current head location (i) towards the requested location (j). We also denote by s_j and t_j the number of bytes to search and transfer (respectively), while s_{rate} and t_{rate} are the search and transfer rates (respectively). These parameters have been identified in Table 6.

Again, it should be noted that the cost of the employed permutations involves the popularity measure which in turn is related to the physical objects real-time playback requirements (as explained in Section 3.1).

Notice that in the calculation of the expected service time for a specific placement instance S, the above formula exhaustively sums all the possible combinations of requests and therefore, the more requests are serviced the more representative the above cost function becomes. Thus for a given placement S as the number of requests $Req_{Num} \to \infty$ the EST(S) is equal to the real cost of servicing Req_{Num} requests.

4.2. A self-improving version of the simulated annealing algorithm

In our previous work Vakali et al. (2001) we have proposed an improved version of the simulated annealing algorithm (called ISA) for the multimedia data placement problem. The key idea behind this improvement was the repetitive execution of the simulated annealing procedure (described in the previous subsection) for a prefixed number of times N_{total}. The total number of trials is distributed to blocks of equal

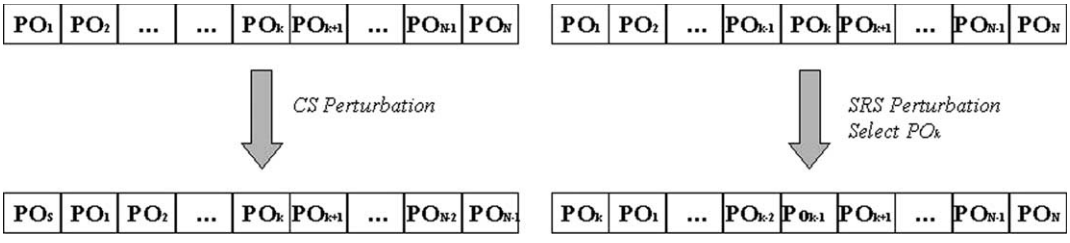

Fig. 3. Neighborhood search alternatives.

length N_{block}. When a block of trials is executed, the process checks which is the minimum service cost up to that point and calculates its relative frequency f_{best} in the specific block. Then, f_{best} is compared with a predefined (small) probability level $0 < f_{low} < 1$. If $f_{low} < f_{best}$, the search proceeds to the next block with the same parameters. Every B blocks, i.e. every BN_{block} repetitions of the SA algorithm, the process calculates the relative frequency f^*_{best} of the best design up to that point and makes a comparison with a (large) probability level $0 < f_{high} < 1$. In case $f^*_{best} > f_{high}$, the procedure stops and the best placement found is reported. Therefore, the repetition was the key factor of the improved version of the simulated annealing algorithm that we had previously proposed. Therefore, the ISA version of the simulated annealing algorithm implemented on the tertiary storage placement problem was based on repetitive execution of the algorithm. This procedure was expected to un-trap the algorithm from any local minima that he might be trapped at, by giving it the chance to reconsider its previous decisions.

The idea of Vakali et al. (2001) has proven to improve the system's performance under the implementation of the ISA algorithm. In this paper, based on the work of Angelis et al. (2001), we propose a self-improving process of the algorithm. The most critical part of the SA algorithm is the report of the optimal or near-optimal placement. In order to "secure" that the proposed (by the algorithm) placement is the best one, certain criteria need to be adopted. Of course, when the optimal is unknown, it is impossible to know securely the efficiency of the result. In this regard, it seems reasonable not to rely on a single execution of the SA algorithm, but to conduct a series of repeated executions in the same way as previously, but additionally modify the initial values of the parameters I and R. This gives to the process a self-improving feature that aims to locate the unknown optimal design with high probability.

Similarly to the previous case, the SA algorithm is repeatedly executed for a prefixed number of times N_{total}. The total number of trials is distributed to blocks of equal length N_{block}. When a block of trials is executed, the process checks which is the minimum service cost up to that point and calculates its relative frequency f_{best} in the specific block. Then, f_{best} is compared with a predefined (small) probability level $0 < f_{low} < 1$. If $f_{best} \leqslant f_{low}$, the values of the parameters I and R are

Table 7
The self-improving process of the simulated annealing algorithm

SISA: self-improving approach based on SA 1
Determine the initial SA parameters I and R
Determine the parameters of the self-improving process:
N_{total}: Total number of repetitions
N_{block}: The size of blocks (N_{total}/N_{block} is the number of all blocks and it must be integer)
f_{low} and f_{high}: The probability levels
I_{upper} and R_{upper}: The upper bounds for I and R respectively
B: Every B blocks the process checks whether the optimal placement appears with relative frequency greater that f_{high} ($B < N_{total}/N_{block}$)
$EST_{best} = 10^9$
repeat for $m = 1$ to N_{total}
Set $Tmpr_0$
Execute the SA algorithm with $Tmpr_0, I, R$ and return $EST(S_{best})$
if $(EST(S_{best}) < EST_{best})$ **then**
$Vect(1\ to\ m-1) = 0$;
$Vect(m) = 1$;
$EST_{best} = EST(S_{best})$;
else if $(EST(S_{best}) = EST_{best})$
$Vect(m) = 1$;
endif
if $(MOD(m, N_{block}) = 0)$ **then**
$f_{best} = SUM(Vect(m - N_{block} + 1\ to\ m))/N_{block}$
if $(f_{best} \leqslant f_{low})$ and $(I < I_{upper})$ **then**
Increase I and R
endif
endif
if $(MOD(m, B \cdot N_{block}) = 0)$ **then**
$f_{best} = SUM(Vect)/m$
if $(f_{best} \geqslant f_{high})$ **then**
stop
endif
endif
endrepeat

increased and the search continues with the next block. If $f_{low} < f_{best}$, the search proceeds to the next block with the same parameters. Every B blocks, i.e. every BN_{block} repetitions of the SA algorithm, the process calculates the relative frequency f^*_{best} of the best statement up to that point and makes a comparison with a (large) probability level $0 < f_{high} < 1$. In case $f^*_{best} \geqslant f_{high}$, the procedure stops and the best placement found is reported. Since the parameters I and R are continuously increased, the process becomes gradually slower. It is therefore necessary to fix reasonable upper bounds I_{upper} and R_{upper} which restrict the duration of the process. In our implementation we set the initial values relatively small and they are gradually increased by the formulae: $I_{NEW} = 1.1 \cdot I$ and $R_{NEW} = R + \frac{1-R}{8}$. The complete self-improving process is presented in pseudocode in Table 7.

5. The simulation model

A simulation model has been developed such that the proposed data placement policies can be evaluated under an appropriately supported tertiary storage topology. The request process is also simulated by supporting bursts of clients' requests arriving in the system. Our simulation model is depicted in Fig. 4 and it is mainly based on the introduction of four modules:

- *The Data Representation Module*: This module mainly represents the multimedia application environment to which the users have access. Both the external and the internal data representation models are supported. Therefore this software component is responsible for the following actions:
  - Construction of the browsing graph and the multimedia objects tree structure based on the users access patterns and the imposed timing constraints.

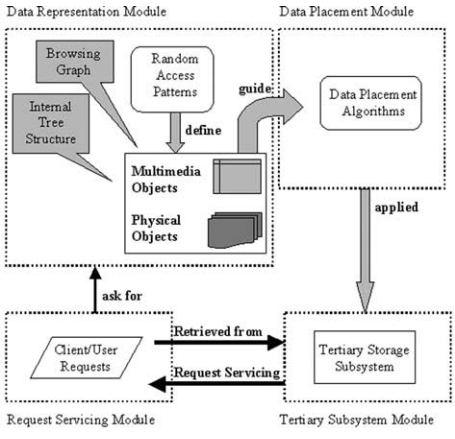

Fig. 4. The modules of the simulation model.

  - Evaluation of the access frequencies and the popularity values of the physical objects.

Notice that in the simulation studies only one copy of any physical object (participated in the application) is stored in the storage system, with no relevance to the number of distinct nodes (it participates in) and the number of its distinct presentations (within each node). These two parameters play important role in the determination of its access frequency which is vital in the presented placement approach.

- *The Tertiary Storage System Module*: The considered tape library is modeled to include a number of tapes, one robot arm and one tape drive. The storage subsystem has been modeled in order to measure the system's performance under various data placement algorithms. The estimation of the total service time is based on the formulae presented in Section 3.2.
- *The Data Placement Module*: This module has the appropriate routines for employing data physical allocation on the tertiary storage subsystem according to the proposed data placement algorithms. The final location of the physical objects within the storage subsystem depends on the placement scheme and the adopted criteria to determine the physical object's popularity and synchronization constraints, which in our case is the popularity of the physical objects.
- *The Request Servicing Module*: The request workloads refer to specific nodes of the external browsing graph while the requests arrive based on the estimated access patterns. Once a request for a specific node of the multimedia application arrives, the physical objects that should be displayed are specified. These physical objects are sorted with respect to the time of their first display within the specified node, and then they are brought to the cache memory. The performance metrics are evaluated in order to evaluate the impact of the data placement algorithms on the system's performance.

6. Experimentation—Results

Various experimentation under different workloads and data placement policies have been applied. The experimentation focuses on the performance evaluation of the proposed placement algorithms, with respect to certain performance objective functions (such as seek and service times). *Organ-pipe* has been chosen as the most indicative algorithm in the class of constructive placements algorithms since it has been proven to outperform other algorithms. All of the SA-based algorithms (i.e. SA, ISA and SISA) are also implemented and comparative results are discussed. The requests were

generated by a process which follows the access patterns of the initial browsing graph such that the cost function (which is chosen to be the expected service time) will be more effective for the implementation of the algorithm. More specifically, the artificial workload was generated by considering the following issues:

- The total number of the physical objects of the pool increases with the number of nodes of the browsing graph.
- The number of physical objects (participating in the same node) is uniformly distributed between 1 and the total number of objects in the pool.
- Each physical object's size varies from some hundreds of KB to hundreds of MB. It is obvious that the total size of multimedia objects is compliant with the size of real multimedia data.
- The workload was generated such that a large percentage of the total tape space is occupied. More specifically, when the model contained small number of tapes (4, 2 tapes) 75–90% of the total available storage capacity is occupied. This percentage inevitably decreases when the number of tapes increases, as the workload is constant. This approach allowed experimentation and performance evaluation for cases when the stored objects are either scattered among the available tapes or stored on a small number of them.

The results of the experimentation are shown in Figs. 5–7 where the graphs show the service and seek times as resulted by the implementation of the five placement policies for storage systems with a varying number of tapes (2,...,10). In Fig. 5 the initial placement, under which the SA-based algorithms were implemented, was the *organ-pipe* placement while the initial placement for the experiments mapped in Figs. 6 and 7 were *random* and *camel* respectively.

6.1. Experimentation remarks

In summary, the experimentation results are characterized by the following:

- In all cases the SA-based algorithms performs better than *random* and *organ-pipe* placement policies.
- The SISA algorithm outperforms all the other placement policies and its performance is steady since the performance metrics remain the same irrespective to the initial placement algorithm. More specifically, the service and seek time values for the implementation of the SISA algorithm under different initial placements do not differ for more than 2% or 2.5%.
- The initial placement has an impact on the performance of both SA and ISA versions of the SA algorithm. In most of the times a better initial placement entails to better performance of the respective algorithm.
- There are cases where the SISA algorithm performs almost the same as the ISA. This may happen when the ISA algorithm manages to find the minimum of the cost function which is also achieved by the SISA algorithm as explained previously.

Concluding, we must notice that there is an obvious gain in the considered performance metrics by the implementation of the proposed SA-based data placement algorithms. The improvement achieved in seek and service time by the implementation of the SISA algorithm is on average 40% and 30% respectively when compared to the *random* placement, while the

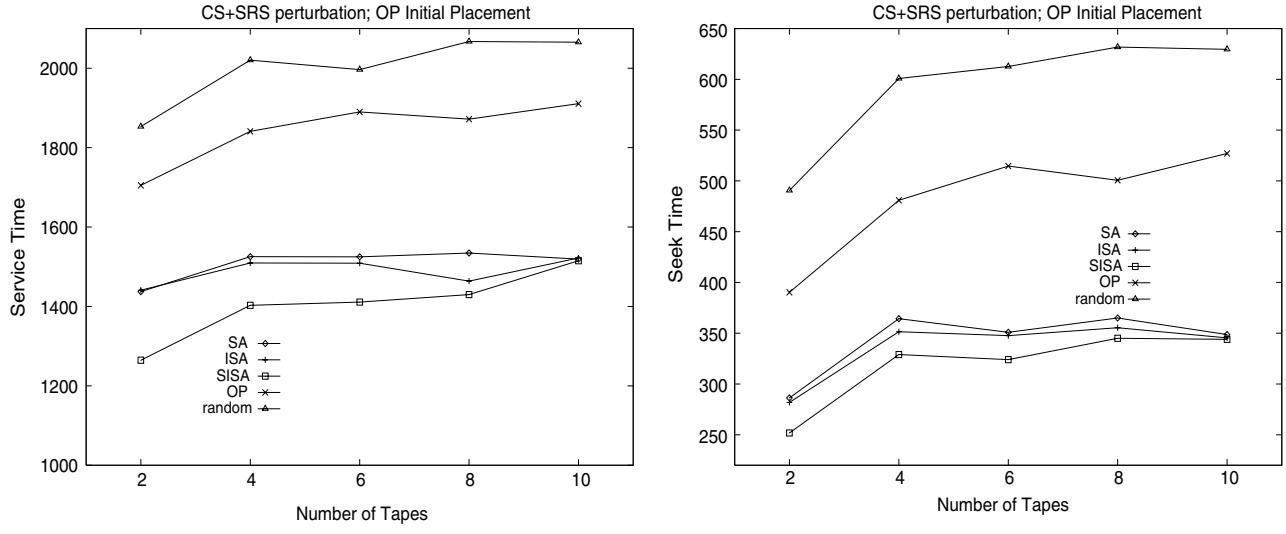

Fig. 5. Seek/service time; number of tapes.

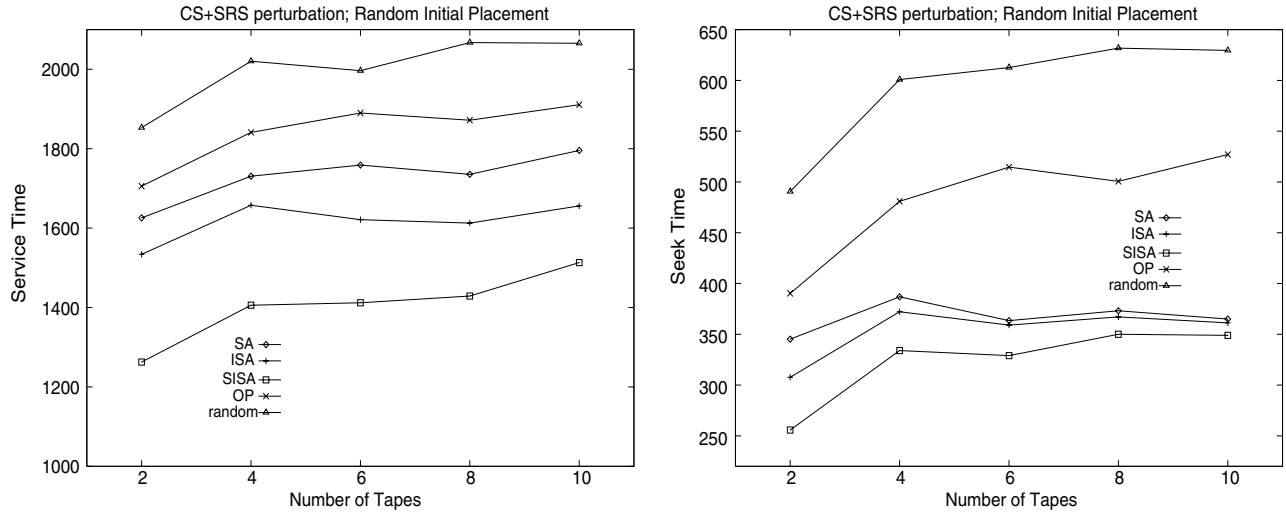

Fig. 6. Seek/service time; number of tapes.

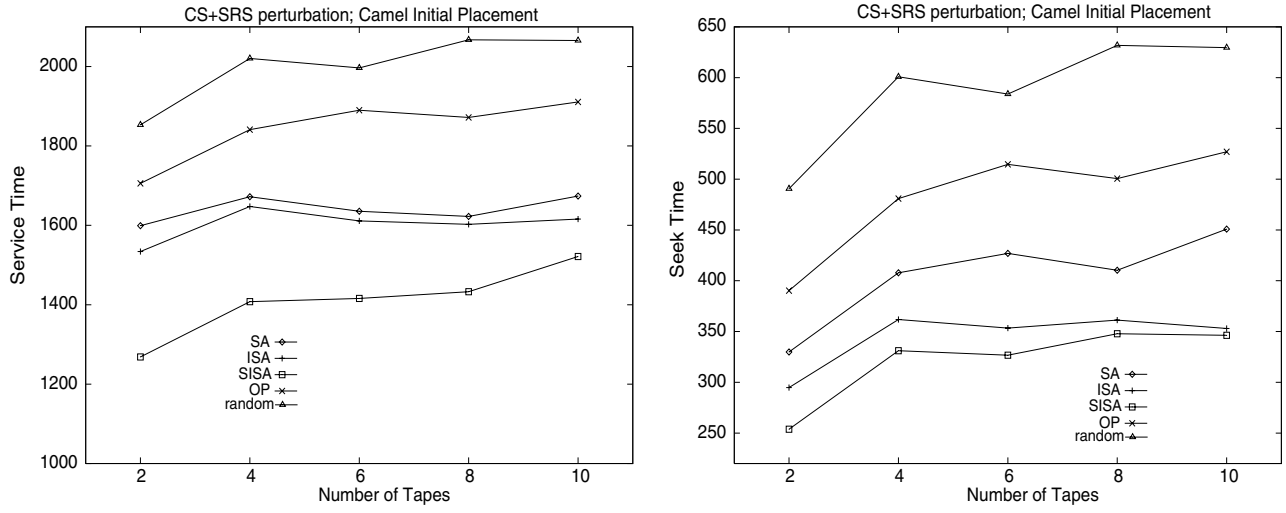

Fig. 7. Seek/service time; number of tapes.

corresponding improvement rates achieved by the simple SA algorithm respectively are 32.5% and 14.1%.

It should be noted that all versions of SA are stochastic optimization algorithms, aiming to (iteratively) improve an initial placement, in terms of an objective cost function. On the other hand, the constructive placement (CP) algorithms, although they are based on the notion of popularity, they do not take into account the cost of the entire placement. The SA, ISA and SISA algorithms are all iterative processes and thus are much less efficient than the CP policies in terms of execution times. It is obvious that (CP) placement algorithms take the decision on the objects' placements rather quickly, after the necessary sorting step, while for the iterative methods it takes a number of iterations to decide. Typically, the simple SA is slower than CP (depending on the configuration), while ISA and SISA become even more slower, due to the improving steps and the on-the run changes in the cooling schedules. However, as it has already been mentioned, the iterative methods are employed in simulation studies when decisions for the system's design need to be made. Thus, they are executed only once during the system's initialization and (given the presented experimentally verified improvements) their choice is considered as a rather successful one.

7. Future perspectives

This paper proposes different multimedia data placement algorithms based on the typical SA algorithm. The multimedia data are represented by an effective two-level model and the data placement algorithms are tailored for a tertiary storage topology. A self-improving version of

the SA algorithm is proposed in the context of the data placement problem. The placement algorithms are guided by a popularity-based criterion evaluated on multimedia application's physical objects. Experimentation was carried out under a developed simulation model. Extensive experimentation runs have shown that the SA-based placement algorithms outperform both the *random* placement approach and the constructive placement schemes in case of requests following the multimedia application's access patterns. Moreover, the proposed SA-based scheme namely the SISA algorithm has been proved to have the best performance for the considered performance metrics.

A significant advantage of the SA algorithms is their simple principles and structure, allowing either to set restrictions in the search space or to make modifications in the cost function very easily (with no significant alterations in the algorithm's structure). We can therefore use the same algorithms in order to investigate the performance under various requirements and restrictions of the placements under consideration. For example, an interesting issue for future research is the implementation of synchronization requirements as imposed to different multimedia objects by considering the various constraints which define the objects relationships. This can be done either by redefining the cost function (for example by adding a penalty for head movements between objects not related in a standard predefined way) or by considering the relationships (for example by setting the appropriate constraints in the perturbation procedure). In general, further research could extend the proposed model to implement the SA algorithms under different criteria for perturbations and/or under a different cost function $C()$ which could be based on the specific workload. Moreover, the SA approach can be further extended so that the algorithm is implemented not only at the tape, but also at a multi-tape drive library level.

Acknowledgements

The authors thank the referees who have contributed in the improvement of the paper's quality, organization and readability. Their comments were valuable and have considerably improved the paper's overall presentation.

References

Aarts, E.H.L., van Laarhoven, P.J.M., 1989. Simulated annealing: an introduction. Statistics Neerlandica 43, 31–52.

Ahmad, I., So, S.-K., Karlapalem, K., 1999. Response time driven multimedia data objects allocation for browsing documents in distributed environments. IEEE Transactions on Knowledge and Data Engineering 11, 386–405.

Angelis, L., Bora-Senta, E., Moyssiadis, C., 2001. Optimal exact experimental designs with autocorrelated errors through a simulated annealing algorithm. Computational Statistics and Data Analysis 37, 275–296.

Azencott, R., 1992. Simulated Annealing. Parallelization Techniques. Wiley.

Bertino, E., Ferrari, E., 1998. Temporal synchronization models for multimedia data. IEEE Transactions on Knowledge and Data Engineering 10, 612–631.

Chen, C.-M., Sinha, R.K., 2000. Analysis and comparison of declustering schemes for interactive navigation queries. IEEE Transactions on Knowledge and Data Engineering 12, 763–778.

Chervenak, A.L., 1994. Tertiary Storage—An Elevation of New Applications. PhD Dissertation, University of California at Berkley.

Christodoulakis, S., Triantaflllou, P., Zioga, F., 1997. Principles of optimally placing data in tertiary storage libraries. In: 23rd International Conference of Very Large Databases (VLDB), pp. 236–245.

Chung, S.M., 1979. Computers and Intractability: A Guide to the Theory of NP-Completenes. Kluwer Academic Publishers.

Escobar-Molano, M.L., Gandeharizadeh, S., Ierardi, D., 1996. An optimal resource scheduler for continuous display of structured video objects. IEEE Transactions on Knowledge and Data Engineering 8, 508–511.

Gemmel, D.J., Vin, H., Kandlur, D.D., Rangan, P.V., Rowe, L.A., 1995. Multimedia storage servers: a tutorial. IEEE Computer, 40–49.

Ghandeharizadeh, S., 1996. Stream-based versus structured video objects: issues, solutions, and challenges. In: Multimedia Database System: Issues and Research Direction. Springer-Verlag, pp. 215–236.

Han, J., Xie, Z., Fu, Y., 1999. Join index hierarchy: an indexing structure for efficient navigation in object-oriented databases. IEEE Transactions on Knowledge and Data Engineering 11, 321–337.

Hillyer, B.K., Silberschatz, A., 1996. On the modeling and performance characteristics of a serpentine tape. In: Proceedings ACM SIGMOD'96 Conference, pp. 170–179.

Hirzalla, N., Falchuk, B., Karmouch, A., 1995. A temporal model for interactive multimedia scenarios. IEEE Multimedia 2, 24–31.

Hua, K.A., Lang, S.D., Lee, W.K., 1994. A decomposition-based simulated annealing technique for data clustering. In: Proceedings of the 13th ACM symposium on Principles of database systems.

Johnson, T., Miller, E., 1998. Performance measurements of tertiary storage devices. In: Proceedings of the 1998 Conference of Very Large Databases (VLDB 1998), pp. 50–61.

Kirkpatrick, S., Gelatt, C.D., Vechhi, M.P., 1983. Optimization by simulated annealing. Science Journal 220, 671–680.

Kwon, Y.-M., Ferrari, E., Bertino, E., 1999. Modeling spatio-temporal constraints for multimedia objects. Knowledge and Data Engineering 30, 217–238.

Metropolis, N.A., Rosenbluth, A., Rosenbluth, M., Teller, A., Teller, E., 1953. Equation of state calculations by fast computing machines. Journal of Chemical Physics 21, 1087–1092.

Nahar, S., Sahni, S., Shragowitz, E., 1986. Simulated annealing and combinatorial optimization. In: 23rd ACM/IEEE Conference on Design Automation, pp. 293–299.

Prabhakar, S., Agrawal, D., El Abbadi, A., Singh, A., 1996. Tertiary storage: current status and future trends. In: Computer Science Department, University of California, Santa Barbara (Technical Report).

Scheier, R.L., 2003. Disk and tape coexist harmoniously in the backup market. Storage Networking World.

Sesardi, S., Rotem, D., Segev, A., 1994. Optimal arrangements of cartridges in carousel type mass storage systems. The Computer Journal 37, 873–887.

Triantafillou, P., Papadakis, T., 1997. On-demand data elevation in a hierarchical multimedia storage server. In: Proceedings of the 23rd Int. Conference on Very Large Databases (VLDB '97).

Triantafillou, P., Papadakis, T., 2001. Continuous data block placement and elevation from tertiary storage in hierarchical storage servers. Cluster Computing: The Journal of Networks, Software Tools and Applications, 157–172.

Vakali, A., Manolopoulos, Y., 1998. Information placement policies In tertiary storage systems, storage models for multimedia object. In: Hellenic Conference of New Information technologies, pp. 205–214.

Vakali, A., Terzi, E., 2001a. Multimedia data elevation under a hierarchical storage model. In: 8th Panhellenic Conference on Informatics.

Vakali, A., Terzi, E., 2001b. Multimedia data storage and representation issues on tertiary subsystems: an overview. ACM—Operating Systems Review 35, 61–77.

Vakali, A., Terzi, E., 2002. Video data storage policies: an access frequency based approach. Computers and Electrical Engineering Journal 44, 49–80.

Vakali, A., Terzi, E., Angelis, L., Hacid, M.-S., 2001. Multimedia documents storage: an evolutionary based application. In: International Workshop on Multimedia Data and Document Engineering (MDDE 2001).

Evimaria D. Terzi graduated from the Department of Informatics of Aristotle University of Thessaloniki (Greece) in 2000, and after that obtained her M.Sc. from the Dept. of Computer Science of Purdue University (USA) in May 2002. Since August 2002, she is with the Dept. of Computer Science of University of Helsinki (Finland), studying towards her Ph.D. Her current research interests include algorithms and computational complexity, data mining and its applications mainly to genomic data.

Athena I. Vakali received a B.Sc. degree in Mathematics from the Aristotle University of Thessaloniki, Greece, a M.Sc. degree in Computer Science from Purdue University, USA (with a Fulbright scholarship) and a Ph.D. degree in Computer Science from the Department of Informatics at the Aristotle University of Thessaloniki. Since 1997, she is a faculty member of the Department of Informatics, Aristotle University of Thessaloniki, Greece (currently she is an Assistant Professor). Her research interests include design, performance and analysis of storage subsystems and data placement schemes for multimedia and Web based information. She is working on Web data management and she has focused on XML data storage issues. She has published several papers in international journals and conferences. Her research interests include storage subsystem's performance, XML and multimedia data, management and data placement schemes.

Lefteris Angelis received his B.Sc. degree and Ph.D. diploma in Mathematics from Aristotle University of Thessaloniki (AUTh). He works currently as a Lecturer at the Department of Informatics of AUTh. His research interests include statistical methods with applications to information systems, simulation methods and algorithms for optimization problems.