Evolutionary Optimization of File Assignment for a Large-Scale Video-on-Demand System

Jun Guo, *Member*, *IEEE*, Yi Wang, *Student Member*, *IEEE*, Kit-Sang Tang, *Member*, *IEEE*, Sammy Chan, *Member*, *IEEE*, Eric W.M. Wong, *Senior Member*, *IEEE*, Peter Taylor, and Moshe Zukerman, *Fellow*, *IEEE*

Abstract—We present a genetic algorithm for tackling a file assignment problem for a large-scale video-on-demand system. The file assignment problem is to find the optimal replication and allocation of movie files to disks so that the request blocking probability is minimized subject to capacity constraints. We adopt a divide-and-conquer strategy, where the entire solution space of file assignments is divided into subspaces. Each subspace is an exclusive set of solutions sharing a common file replication instance. This allows us to utilize a greedy file allocation method for finding a good-quality heuristic solution within each subspace. We further design two performance indices to measure the quality of the heuristic solution on 1) its assignment of multicopy movies and 2) its assignment of single-copy movies. We demonstrate that these techniques, together with ad hoc population handling methods, enable genetic algorithms to operate in a significantly reduced search space and achieve good-quality file assignments in a computationally efficient way.

Index Terms—File assignment, genetic algorithm, video-on-demand.

1 Introduction

With the rapid advances in multimedia, communications, and mass storage technologies, the deployment of commercial video-on-demand (VOD) services to a large population of users has become a reality [1]. For such a large-scale VOD system, it is essential to manage and store an extensive collection of movie titles in a digitized and compressed format by using a storage subsystem made of a large cluster of online disks. Due to the significant asymmetry in access demand for different movie titles, it is necessary to replicate popular movie titles over multiple disks so as to increase the stream capacity of the system in serving user requests for popular movie titles.

Because of the large disk storage space and I/O bandwidth required in storing and delivering movie contents, it is important to manage the limited disk resources in the VOD system efficiently to achieve good service quality. Due to the long-lived nature of the channel holding time, an important performance metric of the VOD system is the *request blocking probability* (RBP). The impact of file assignment optimization on the RBP of the VOD system has been well established in the literature [2], [3], [4], [5], [6], [7], [8], [9], [10], [11]. Given a large number of disks with limited capacity in both storage space and I/O bandwidth and a large library of movie titles with significant asymmetry in access demand and file size, the file assignment problem is to find how we can replicate and allocate movie files to disks so that the RBP of such a capacity-constrained system is minimized.

As we will see in Section 2, earlier proposals in the literature for the file assignment problem relied on more or less simplified and, thus, unrealistic assumptions. Although the complexity of the file assignment problem can be greatly reduced with such unrealistic assumptions, the practicability of the obtained solutions in real systems is significantly undermined. This concern has led us to approach the file assignment problem in a more realistic way, which gives rise to a challenging constrained nonlinear integer optimization problem.

Our focus in this paper is to present an evolutionary approach based on genetic algorithms (GAs) [12] for finding good-quality solutions in a computationally efficient way for this difficult file assignment problem. The proposed evolutionary approach exploits an elaborate transformation of the file assignment problem so that ad hoc methods can be designed to manipulate the stochastic search of GAs within a drastically reduced yet effective solution space. To circumvent the cumbersome RBP evaluation for each feasible solution explored in the evolution process, we design two performance indices to estimate the quality of a file assignment on 1) its assignment of multicopy movies and 2) its assignment of single-copy movies. By means of these two easy-to-compute attributes that jointly measure the quality of a file assignment, we further expedite the

- *J. Guo is with the Networks Research Group, School of Computer Science and Engineering, The University of New South Wales, NSW 2052, Australia. E-mail: jguo@cse.unsw.edu.au.*
- *Y. Wang is with the School of Computer Science and Information Technology, RMIT University, VIC 3001, Australia. E-mail: yi.wang@computer.org.*
- *K.-S. Tang, S. Chan, and E.W.M. Wong are with the Department of Electronic Engineering, City University of Hong Kong, Hong Kong SAR, China. E-mail: {eekstang, eeschan, eeewong}@cityu.edu.hk.*
- *P. Taylor is with the Department of Mathematics and Statistics, The University of Melbourne, VIC 3010, Australia. E-mail: P.Taylor@ms.unimelb.edu.au.*
- *M. Zukerman is with the Department of Electrical and Electronic Engineering, The University of Melbourne, VIC 3010, Australia. E-mail: m.zukerman@ee.unimelb.edu.au.*

Manuscript received 7 Feb. 2007; revised 13 Oct. 2007; accepted 3 Dec. 2007; published online 14 Dec. 2007.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-0056-0207.
Digital Object Identifier no. 10.1109/TKDE.2007.190742.

stochastic search of GAs and yet obtain file assignment solutions of comparable quality.

The rest of this paper is organized as follows: We discuss related work in Section 2. The VOD system model is described in Section 3. Section 4 presents the problem formulation and the transformation method. Section 5 deals with the design of the ad hoc population handling methods and the implementation of the evolutionary optimization program based on GAs. Details of the two performance indices are provided in Section 6. In Section 7, we show how we solve the file assignment problem by means of the two easy-to-compute performance indices using multiobjective optimization techniques. We demonstrate in Section 8 the superior performance of our proposed algorithms through extensive numerical experiments. Finally, we provide concluding remarks in Section 9.

2 Related Work

The impact of file assignment optimization on the RBP performance of VOD systems and the motivation of the file assignment problem considered in this paper have been well established in the literature [2], [3], [4], [5], [6], [7], [8], [9], [10], [11]. It is complicated to the point that the RBP of the VOD system is not only susceptible to how movie files are assigned to disks but also sensitive to how disks (servers) are selected to serve user requests for multicopy movies [10].

Little and Venkatesh [2] studied a simplified version of the problem, assuming that requests for multicopy movies are handled in accordance with what we call a *single random trial* (SRT) server selection scheme. In the SRT system, when a request for a multicopy movie arrives, one of the disks storing a file copy of the requested movie title is randomly selected. If the I/O bandwidth (stream capacity) of the disk is used up, the request is simply blocked, without further attempting any other disk that keeps a file copy of the requested movie title. Little and Venkatesh proved that the RBP of the SRT system is minimized if each homogeneous disk has an equal probability of being accessed. In other words, the problem is reduced to finding a movie file assignment that achieves *disk load balancing* [11].

Taking advantage of the latter, various algorithms were proposed in [3], [4], [5], and [6] to find optimal or near-optimal solutions for the simplified file assignment problem. The polynomial time greedy algorithm proposed in [4] guarantees the file assignment to attain disk load balancing. However, it relies heavily on a weighted scheduler and the assumption of unlimited disk storage space. For a practical system with disks of limited storage space, it was proved in [3] that, even if all movie titles are restricted to having only one single file copy in the system, the file assignment problem is still NP-hard [13]. An interesting method was proposed in [3] to decide the number of file copies for each movie title. It is essentially similar to the well-known apportionment method used to achieve fairness of state representation in a government congress [14]. However, such an approximation method works for the VOD system only if all movie titles are of identical file size, which is not a realistic assumption in practice.

Considering, again, in the context of the SRT system but for a more realistic assumption of heterogeneous movie file sizes and a system of heterogeneous disks, a hybrid evolutionary algorithm was presented in [5] to find near-optimal solutions for the file assignment problem. A similar version of the problem in [5] was studied in [6], which allows for the same weighted scheduler as in [4]. The methodology in [6] relies on solving a relaxed problem for finding the ideal access probability of each heterogeneous disk and a goal programming approach for performing the assignment and reassignment of movie files iteratively until a sufficiently near-optimal file assignment solution is found.

The SRT system is inherently inefficient in disk resource utilization, given the existence of multicopy movies. Moreover, the quality of a file assignment with respect to the RBP established in the SRT system does not hold true in situations where more efficient server selection schemes are used [10].

Without much concern for the cost and complexity of real-time stream scheduling in large-scale VOD systems, a stream repacking scheme was considered in [7], which utilizes disk resources more efficiently than SRT. The heuristic algorithm proposed in [7] for the file assignment problem contains two parts. First, it again relies on the unrealistic assumption of a homogeneous movie file size so that the number of file copies replicated for each movie title could be decided approximately by the apportionment method [14]. Second, taking advantage of multicopy movies, it provides a greedy file allocation method that seeks to *connect* more pairs of disks by allocating some common movie files on both disks in each pair. It was observed that high disk connectivity produced this way increases the potential of finding a feasible schedule for facilitating stream repacking. Subsequently, two other groups of researchers [8], [9] followed the strategy in [7] but offered variations of the stream repacking scheme and of the heuristic algorithm to achieve high disk connectivity.

Clearly, these earlier proposals for the file assignment problem rely on more or less simplified and, thus, unrealistic assumptions so that the complexity of the problem can be greatly reduced. In this paper, we approach the file assignment problem in a more realistic way. We assume disks of limited storage space and movies of heterogeneous file sizes. We consider a more realistic server selection scheme called *least busy fit* (LBF) [10] in handling requests for multicopy movies. LBF can achieve efficient utilization of disk resources at reasonable implementation cost and complexity. Making use of the available system state information, the LBF system always directs a request for a multicopy movie to the least busy disk where a file copy of the requested movie title is placed. We will see that the file assignment problem in the context of the LBF system is challenging. Our goal in this paper is to develop robust algorithms for tackling this difficult file assignment problem in a computationally efficient way.

3 System Model

For the reader's convenience, we provide in Table 1 a list of major symbols that we shall define and use in this paper.

TABLE 1
Summary of Major Symbols

Symbol	Definition
$\mathcal{D}$	Set of disks in the system
J	Number of disks in the system
$\mathcal{F}$	Set of movie titles in the system
M	Number of movie titles in the system
C	Disk storage space
N	Disk stream capacity
L_m	File-size of movie m
n_m	Number of file-copies of movie m
ϖ	Upper bound on n_m, $\forall m \in \mathcal{F}$
Ω_m	Set of disks storing a file-copy of movie m
Φ_j	Set of movie titles placed on disk j
X	File assignment matrix, $X = [x_{m,j}]_{M \times J}$
$\mathbf{n}$	File replication instance, $\mathbf{n} = (n_1, n_2, \ldots, n_M)$
$\mathbf{\Omega}$	File allocation instance, $\mathbf{\Omega} = (\Omega_1, \Omega_2, \ldots, \Omega_M)$
$\mathbf{\Phi}$	File allocation instance, $\mathbf{\Phi} = (\Phi_1, \Phi_2, \ldots, \Phi_J)$
λ	Total request arrival rate
p_m	Popularity of movie m
$1/\mu_m$	Mean channel holding time of movie m
A_m	Traffic load of movie m
$\hat{A}_c$	Traffic load of type-c movies
A	Total traffic load
ζ	Movie popularity distribution skewness parameter
σ	Crossover rate of GAs
δ	Mutation rate of GAs
K	Population size of GAs
G	Predefined number of generations per run of GAs

Consider a VOD system with a set $\mathcal{D}$ of J online disks labeled $1, 2, \ldots, J$ and a set $\mathcal{F}$ of M movie titles marked $1, 2, \ldots, M$. Note that each online disk in this context can be either a conventional hard disk drive or a disk striping group [15]. Without loss of generality, we describe any substantive secondary storage device in the storage subsystem of the VOD system as a "disk" throughout this paper. In situations where the system consists of heterogeneous disks, we assume the use of the disk merging technique [16] so that a logical collection of J homogeneous disks can be constructed from the array of heterogeneous disks. It was shown in [16] that the disk merging technique yields secondary storage with high availability and maximum flexibility. In our context, it allows the use of the *combination load balancing* (CLB) technique [11] to utilize disk resources more efficiently.

We assume that each disk has a limited storage space of C units. (For example, one unit of storage space could be 1 Gbyte.) Each variable-bit-rate (VBR) compressed video stream is delivered over a constant-bit-rate (CBR) channel by using temporal smoothing algorithms [17]. Bit rates of independent video streams are considered to be statistically equivalent [18]. Thus, each disk can support up to N concurrent video streams. The file size of movie m is L_m units. Therefore, it requires $L = \sum_{m \in \mathcal{F}} L_m$ units of disk storage space to allocate one file copy for each movie title in $\mathcal{F}$. We assume that $L < JC$ so that the system has spare disk storage space to place multiple file copies for certain movie titles in $\mathcal{F}$. We also assume that $L > C$ so that we cannot replicate each movie title in $\mathcal{F}$ to each disk in $\mathcal{D}$.

Let $X = [x_{m,j}]_{M \times J}$ denote a file assignment matrix, where $x_{m,j} = 1$ if a file copy of movie m, $m \in \mathcal{F}$, is placed on disk j, $j \in \mathcal{D}$; otherwise, $x_{m,j} = 0$. For X to be a *feasible* file assignment solution, it must allocate at least one file copy for each movie m in $\mathcal{F}$. It must also satisfy the storage space constraint on each disk in $\mathcal{D}$, that is, $\sum_{m \in \mathcal{F}} x_{m,j} L_m \leq C$, $\forall j \in \mathcal{D}$. To extract from X the information of how each movie title is replicated and where the movie and its replicas (if it is replicated) are allocated, we define two concepts.

We define a *file replication instance* as the vector $\mathbf{n} = (n_1, n_2, \ldots, n_M)$, where $n_m = \sum_{j \in \mathcal{D}} x_{m,j}$ indicates the integer number of file copies replicated for movie m. We call a movie title that has c file copies a type-c movie. Considering the fact that no performance gain can be obtained by storing multiple file copies of any movie title on one single disk [2], the n_m file copies of movie m must be allocated to n_m different disks. The overall disk storage space required by a file replication instance $\mathbf{n}$ is given by $\sum_{m \in \mathcal{F}} n_m L_m$. We further define a *file allocation instance* as a disk location arrangement $\mathbf{\Omega} = (\Omega_1, \Omega_2, \ldots, \Omega_M)$ or $\mathbf{\Phi} = (\Phi_1, \Phi_2, \ldots, \Phi_J)$ for the set of movie files specified in the file replication instance $\mathbf{n}$. The mth element in $\mathbf{\Omega}$ describes the set of n_m different disks, where the n_m file copies of movie m are stored. It corresponds to all nonzero items in the mth row of X and is given by $\Omega_m = \{j : j \in \mathcal{D}, x_{m,j} = 1\}$. Similarly, the jth element in $\mathbf{\Phi}$ describes the set of movie titles of which a file copy is placed on disk j. It corresponds to all nonzero items in the jth column of X and is given by $\Phi_j = \{m : m \in \mathcal{F}, x_{m,j} = 1\}$. Thus, X can be equivalently represented by the tuple $\langle \mathbf{n}, \mathbf{\Omega}, \mathbf{\Phi} \rangle$.

It was observed in [19] that interarrival times of user requests in streaming multimedia systems are exponentially distributed. We therefore assume that the aggregate arrivals of user requests in the VOD system follow a Poisson process with a rate of λ requests per time unit. (For example, one time unit could be 1 hour.) The request arrival processes for different movie titles are mutually independent Poisson processes. The demand rate for movie m creates its popularity profile p_m, defined as the relative probability of the movie m being requested by users, and $\sum_{m=1}^{M} p_m = 1$. The channel holding time of movie m is arbitrarily distributed with a mean of $1/\mu_m$ time units. In practice, the movie popularity profiles are updated periodically to capture the variability of user demand. During the time interval between such updates, the request arrival rate of movie m is given by λp_m. Therefore, the traffic load A_m of movie m is given by $\lambda p_m / \mu_m$. The aggregate traffic load $\hat{A}_c$ of all type-c movies is obtained by $\sum_{m \in \mathcal{F}, n_m = c} A_m$. The aggregate traffic load A of all movie titles in $\mathcal{F}$ is computed by $\sum_{m \in \mathcal{F}} A_m$.

For all numerical experiments that we have conducted in this paper, we assume that the movie popularity profiles in the VOD system are distributed so that

$$p_m = \frac{m^{-\zeta}}{\sum_{k=1}^{M} k^{-\zeta}}, \ \forall m \in \mathcal{F}. \tag{1}$$

The parameter ζ in (1) determines the skewness of the popularity distribution. This distribution function is commonly known as a Zipf-like distribution [20], since, when $\zeta = 1$, it becomes a Zipf distribution [21]. Moreover, we assume that the channel holding time of movie m, taking into consideration user interactive behavior [22], follows a lognormal distribution. Without loss of generality, we assume that the mean channel holding time of movie m is proportionally equivalent in magnitude to its file size.

4 Problem Formulation and Transformation

Given the parameters and notation defined in Section 3, the file assignment problem aiming at finding a feasible solution that minimizes RBP in the VOD system can be formulated as

$$\begin{aligned} &\text{Minimize} && \text{RBP}_X \\ &\text{subject to} && \sum_{j\in\mathcal{D}} x_{m,j} \geq 1,\ \forall m \in \mathcal{F}, \end{aligned} \tag{2}$$

$$\sum_{m\in\mathcal{F}} x_{m,j} L_m \leq C,\ \forall j \in \mathcal{D}, \tag{3}$$

$$x_{m,j} \in \{0,1\},\ \forall m \in \mathcal{F},\ \forall j \in \mathcal{D}. \tag{4}$$

The inequality in (2) requires each movie in $\mathcal{F}$ to have at least one file copy stored in the system, which we call the *movie availability constraint*. The inequality in (3) describes the *storage space constraint* for each disk in $\mathcal{D}$. For each X, we derive its corresponding tuple $\langle \mathbf{n}, \mathbf{\Omega}, \mathbf{\Phi} \rangle$ so that its RBP can be evaluated by a fixed-point approximation model of the LBF system provided in the Appendix.

The fixed-point approximation model can be used to evaluate the RBP of the LBF system with sufficient accuracy [10]. However, it is complex in nature. Among other things, the evaluation of the fixed-point approximation model is, by itself, an iterative process. Thus, the objective function RBP in this context has to be treated like a *black box*. Moreover, this black box must take in the complete binary representation of a feasible file assignment as the input so that it can return a valid RBP result. This makes it infeasible to drop the integrality requirements on the binary variables in (4) and to compute for gradients. We are not aware of any existing mathematical approach that can be applied to solve such a constrained nonlinear integer optimization problem. We have therefore resorted to GAs [12].

An important point to realize in a practical VOD system is that the spare disk storage space available for accommodating multicopy movies is usually not very large. Consequently, the region of infeasible solutions to the file assignment problem can be rather large. Although constraint handling methods [23] can be applied to guide the search direction of GAs toward the feasible region, we will see in Section 8.1 that it can take a considerably long time before GAs can locate the feasible region. For the sake of a more efficient stochastic search using GAs, we have designed a *divide-and-conquer* strategy that allows us to transform the original file assignment problem in such a way that we can operate GAs within a drastically reduced yet effective search space.

To that end, we have divided the entire solution space of the original problem into subspaces. Each subspace is an exclusive set of file assignments sharing a common file replication instance and is conquered by finding a good-quality heuristic solution through the greedy file allocation method presented in [11]. In that paper, the authors established that such a heuristic solution, if any, aims for uniform *disk resource sharing* of multicopy movie traffic and disk load balancing on single-copy movie traffic. It yields a closer-to-optimal RBP in the VOD system than those methods proposed in [7], [8], and [9], which typically aim for high disk connectivity. This way, the original problem is reduced to finding an optimal file replication instance, of which the heuristic solution achieves a globally minimal RBP. Let $\mathbf{\Omega}^* = (\Omega_1^*, \Omega_2^*, \ldots, \Omega_M^*)$ or $\mathbf{\Phi}^* = (\Phi_1^*, \Phi_2^*, \ldots, \Phi_J^*)$ denote the heuristic file allocation instance, if any, of a file replication instance $\mathbf{n}$. The file assignment problem can now be reformulated as

$$\begin{aligned} &\text{Minimize} && \text{RBP}_{\langle \mathbf{n}, \mathbf{\Omega}^*, \mathbf{\Phi}^* \rangle} \\ &\text{subject to} && n_m \in \{1, 2, \ldots, \varpi\},\ \forall m \in \mathcal{F} \\ & && \sum_{m\in\Phi_j^*} L_m \leq C,\ \forall j \in \mathcal{D}, \end{aligned}$$

where $\varpi = J$ is defined as the upper bound on $n_m, \forall m \in \mathcal{F}$. We will see in Section 8.2 that in practice, an upper bound smaller than J may be selected. Since the value of each element in the file replication instance can be easily controlled within a positive integer range $[1, \varpi]$, the movie availability constraint in (2) of the original problem formulation is readily satisfied in the transformed problem formulation. For a system of J disks and M movie titles, the search space is reduced from 2^{JM} file assignment solutions to merely ϖ^M file replication instances.

Recall that the overall disk storage space required by a file replication instance $\mathbf{n}$ is given by $\sum_{m\in\mathcal{F}} n_m L_m$. Clearly, if $\sum_{m\in\mathcal{F}} n_m L_m > JC$, no feasible file assignment solutions can be found for $\mathbf{n}$. Such a file replication instance is therefore *strictly nonallocatable*. On the other hand, we will see that, even if

$$\sum_{m\in\mathcal{F}} n_m L_m \leq JC, \tag{5}$$

there is no certainty that any feasible file assignment solution can be found for $\mathbf{n}$. If this occurs, the file replication instance is not *strictly allocatable*. This is due to heterogeneous movie file sizes but is also attributed to the stringent requirement that multiple file copies of any movie title in $\mathcal{F}$ must be placed on different disks. A file replication instance satisfying (5) is hence considered as *likely allocatable*. For the transformed problem, a likely allocatable file replication instance is allocatable only if a heuristic solution can be found by the greedy file allocation method.

Based on this reasoning, we are able to further partition the solution space of the transformed problem into two separate regions: 1) strictly nonallocatable file replication instances and 2) likely allocatable file replication instances. Now, the transformed problem boils down to the design of ad hoc methods so that we can operate GAs solely within

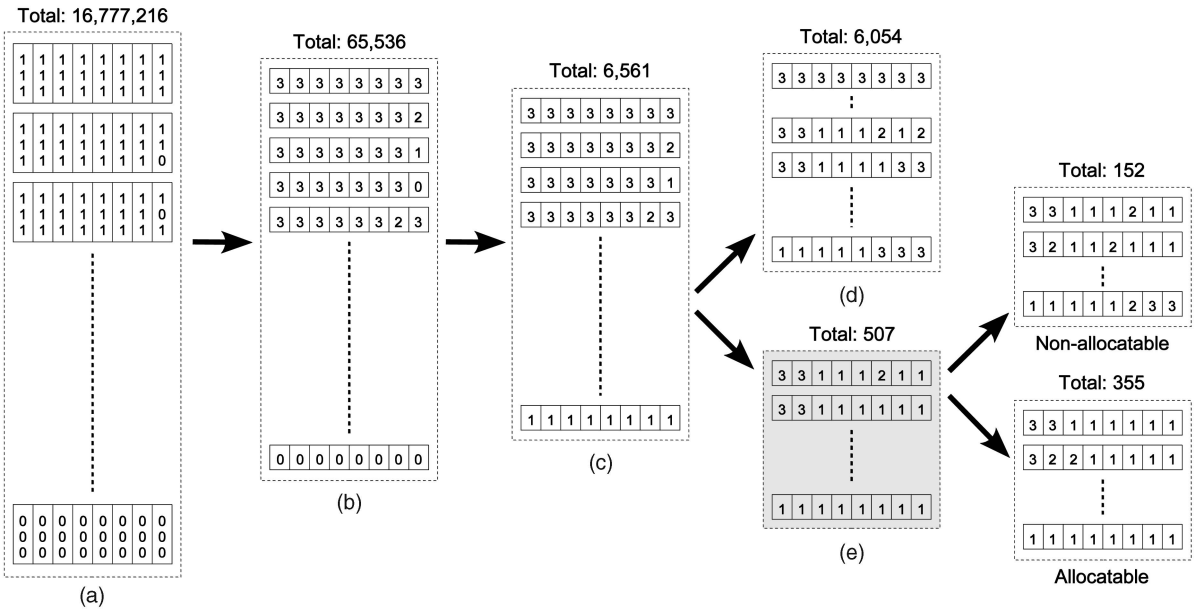

Fig. 1. Reduce the search space by problem transformation in a system of three disks and eight movie titles. (a) File assignment solutions to the original problem. (b) File replication instances to the original problem. (c) File replication instances to the transformed problem. (d) Strictly nonallocatable file replication instances. (e) Likely allocatable file replication instances.

the search space of likely allocatable file replication instances and thus further improve the stochastic search efficiency.

Fig. 1 illustrates an example of how the proposed transformation method drastically reduces the search space of the file assignment problem. Here, we consider a system of three disks and eight movie titles. Each disk has a storage space of four units. Table 2 provides the corresponding movie file size. As a result, 36.5 percent of the disk storage space can be utilized to replicate certain movie titles. The entry (m, j) of each file assignment matrix depicted in Fig. 1 is 1 if a file copy of movie m is placed on disk j. The element n_m of each file replication instance describes the number of file copies of movie m. We see in Fig. 1 that, even for such a small system, the search space of the original problem is reduced by more than four orders of magnitude. Only 355 out of the 507 likely allocatable file replication instances are strictly allocatable, but we are not able to further confine the search space to this narrower region.

5 Single-Objective Evolutionary Optimization

GAs are population-based generic search methods inspired by the mechanism of natural selection obeying the rule of "survival of the fittest" [24]. In a typical implementation of GAs, a population of chromosomes is processed. Each chromosome represents a candidate solution to the problem. Starting from an initial population of randomly created chromosomes, GAs perform multidirectional stochastic search through a genetic evolution process, without the need for any problem information, except for the objective function values. It is hoped that after a certain number of generations, the best chromosome represents a good-quality solution that is reasonably close to the optimal solution. Considering that GAs have been extensively and successfully used for solving various real-world complex optimization problems due to their broad applicability, ease of use, and global perspective [12], we shall base our implementation of the single-objective evolutionary optimization (SOEO) program in this paper on GAs.

5.1 Chromosome Representation

Fig. 2 depicts the chromosome structure of a file assignment matrix in the original problem and that of a file replication instance in the transformed problem. The gene at the kth locus, $k = J(m-1)+j$, of a chromosome X in the original problem controls a binary number indicating if a file copy of movie m is placed on disk j in the corresponding file assignment solution. Thus, the allele space of each gene in the original problem is $\{0, 1\}$. Similarly, the gene at the mth locus of a chromosome $\mathbf{n}$ in the transformed problem controls a positive integer representing the number of file copies replicated for movie m specified in the corresponding file replication instance. Thus, the allele space of each gene in the transformed problem is $\{1, 2, \ldots, \varpi\}$.

TABLE 2
Movie File-Size Distribution in the Three-Disk Example

Movie ID	1	2	3	4	5	6	7	8
File-size (Unit)	1.17	0.67	1.08	1.07	1.25	0.61	0.77	1.00

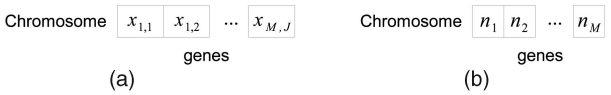

Fig. 2. Chromosome structure. (a) File assignment matrix in the original problem. (b) File replication instance in the transformed problem.

TABLE 3
Impact of ϖ on the Population Size K

ϖ	2	3	4	5	6	7	8	9	10
K	16	26	38	50	62	74	86	98	110
$\alpha(\%)$	99.7	99.2	99.3	99.3	99.3	99.2	99.2	99.1	99.1

5.2 Choice of Population Size

The performance of GAs is susceptible to population size. In general, a small population size increases the likelihood of premature convergence due to the loss of niche, while a large population size leads to a high computation cost. In this paper, the principle of *allele coverage* established in [25] has been used as a guideline to decide an appropriate value for population size.

To prevent the premature convergence of GAs due to poor population diversity, Reeves [25] suggested a preferable property of an initial population such that "every possible point in the search space should be reachable from the initial population by crossover only." This property may only be achieved if the entire allele space is covered at each locus in the whole population of chromosomes. Given the population size K, the chromosome length M, and the cardinality ϖ of the gene at each locus of the chromosome, the probability α of allele coverage is computed by

$$\alpha = \left[\frac{\varpi! S(K, \varpi)}{\varpi^K}\right]^M \tag{6}$$

in the transformed problem, where $S(K, \varpi)$ is the Stirling number of the second kind [26]. Equation (6) serves as a guideline for us to choose an appropriate K such that it is large enough to ensure a sufficiently high probability of allele coverage in the initial population. Note that for the original problem, we can compute α from (6) simply by replacing ϖ with 2 and using MJ as the chromosome length. Table 3 enumerates the smallest K required to meet $\alpha \geq 99$ percent for each ϖ in the range $[2, J]$, given $J = 10$ and $M = 100$.

5.3 Initial Population

A naive population handling approach for population initialization is to generate the initial population of chromosomes in a purely random way within the entire solution space. For the original problem, this is done by randomly selecting a binary number for each gene of the chromosome. For the transformed problem, this is done by marking the corresponding gene with a positive integer randomly selected from the allele space. Due to the large infeasible region inherent in the file assignment problem, the initial population created in such a purely random way likely contains many chromosomes far from the feasible region. Our proposed transformation method allows us to exploit the problem-specific knowledge of the file assignment problem so that we can initialize the population in an ad hoc way and obtain a diverse population sufficiently close to the feasible region.

To that end, we create chromosomes for the initial population by replicating movie titles in a greedy biased yet random manner. Since our design goal with the proposed transformation method is to operate GAs solely within the search space of likely allocatable file replication instances, we ensure that chromosomes in the initial population are randomly generated within the domain of likely allocatable file replication instances. Additionally, given the spare disk storage space, it is preferable that multiple file copies are allocated for movie titles with high traffic load. On the other hand, we may replicate movie titles with a small file size (though less popular) to fully utilize the disk storage space.

Procedure: Create Chromosome
Let $O' = 0$;
Let $L' = L$;
Do for each movie m in the decreasing order according to A_m
  Decrease L' by L_m;
  $x = \min(\varpi, \lfloor \frac{JC - O' - L'}{L_m} \rfloor)$;
  Randomly select an integer from $[1, x]$ for n_m;
  Mark the gene for movie m with n_m;
  Increase O' by $n_m L_m$;
End

Fig. 3. Ad hoc method for creating a chromosome for the initial population.

Bearing these concerns in mind, we have specifically arranged all movie titles in decreasing order according to A_m. During the procedure of generating a chromosome, for each movie m in that order, we mark its corresponding gene with a positive integer n_m randomly selected from the range $[1, x]$. Let O' count the required disk storage space for all marked genes. Let L' count the overall file size of all unmarked genes (one file copy for each such movie title), except that of movie m. We set x as ϖ if $\varpi L_m \leq JC - O' - L'$. Otherwise, it is given by the largest positive integer satisfying $xL_m < JC - O' - L'$. A procedure that implements this ad hoc method is shown in Fig. 3.

5.4 Genetic Operators

In each cycle of the evolution process, a mating pool is formed from the current population to enable reproduction in the subsequent generation. For each pair of coupled chromosomes in the mating pool, a uniform crossover operator is used to promote information exchange between the two chromosomes with an operational rate σ. To introduce greater variability into the offspring, for each gene of the offspring, a uniform mutation operator with a small probability δ is applied. For the original problem, if the gene $x_{m,j}$ is chosen for mutation, it is simply changed from 1 to 0, and vice versa. For the transformed problem, if the gene n_m is chosen for mutation, it is altered with a positive integer randomly selected from the allele space.

5.5 Repair Mechanism

It is possible that an offspring generated through the genetic operators lies in the region of strictly nonallocatable file replication instances. A purely random population handling approach would leave the offspring as is. In contrast, to meet our design goal with the proposed transformation method, we handle it in an ad hoc way. We specifically apply a repair mechanism to correct the offspring into the

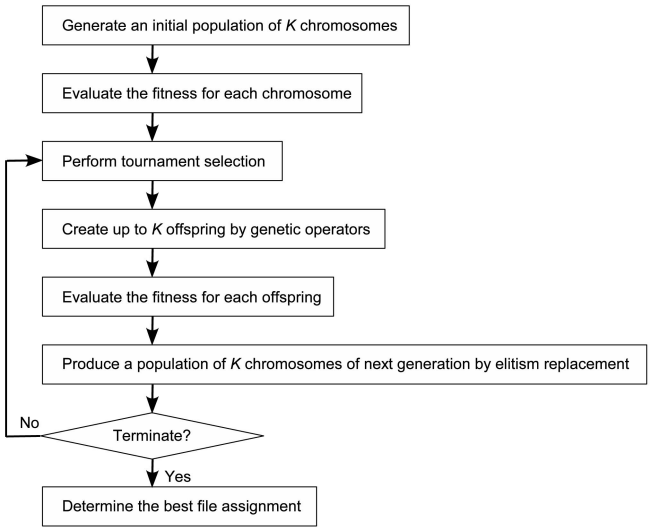

Fig. 4. Flowchart of the SOEO implementation.

region of likely allocatable file replication instances. An operation of the repair mechanism randomly selects a multicopy movie m and decreases n_m by one. This operation is repeated until the overall disk storage space required by the chromosome drops below JC.

5.6 Implementation

Fig. 4 shows the general procedures of the SOEO implementation. Note that such an implementation is applicable to both the original problem and the transformed problem. While we will describe in Section 8.1 the detailed constraint handling method required for dealing with the original problem and the transformed problem with purely random population handling, here, we provide the details of the SOEO implementation for the transformed problem with ad hoc population handling.

We start by creating an initial population of size K by using the ad hoc method presented in Fig. 3. We identify the heuristic solution for each chromosome through the greedy file allocation method. If the chromosome is allocatable, we compute its RBP. If no heuristic solution is found, we impose a death penalty by setting the RBP of the chromosome to 1. The fitness $f(\mathbf{n})$ of each chromosome $\mathbf{n}$ is obtained by $1 - \text{RBP}$.

In each generation of SOEO, we use the tournament selection operator to select K solutions from the current population, some of which may be duplicate, to form the mating pool. Tournament selection is known to have better or equivalent convergence and computational time complexity properties when compared to any other selection operator that exists in the literature [27]. For two chromosomes chosen for competition in a tournament, the winner is the one with higher fitness. The rule is set in such a way that any chromosome is made to participate in exactly two tournaments. As a result, any chromosome in the current population will have at most two copies in the mating pool. Tournament selection essentially limits the number of copies of each chromosome to be placed in the mating pool, which is useful to prevent predominance of elitist solutions and thus prevent premature convergence of GAs.

The K solutions in the mating pool are then genetically varied through crossover and mutation operations, as explained in Section 5.4. If the resulting chromosome is strictly nonallocatable, we apply the repair mechanism to correct the solution into the region of likely allocatable file replication instances.

The mating pool and the offspring are combined into one transitional population. In the same way as for the initial population, we decide the RBP of each offspring and obtain its fitness value. The best K chromosomes are selected and placed in the new population for the next generation.

The evolution process is terminated after a predefined number of G generations have been completed. Alternatively, we presume convergence if the best solutions are not improved over a certain succession of generations. In either case, the file assignment with the highest fitness in the last population is selected as the optimal solution. Ties are broken by choosing the solution that requires the smallest overall disk storage space.

6 Performance Indices

It was demonstrated in [11] that, for an LBF system with a specified file replication instance, a file allocation instance that ideally attains CLB always yields a lower bound on the RBP. CLB is defined in such a way that for each c, $c \geq 1$, the traffic wishing to access $\text{type-}c$ movies is uniformly distributed among all $\binom{J}{c}$ groups of c disks chosen from the set of J disks in the system. It was observed in [11] that the working principle of CLB is intuitively due to the fact that it maximizes the disk resource sharing of multicopy movie traffic, in addition to a straightforward disk load balancing on single-copy movie traffic. Based on the definition of disk resource sharing, the amount of movie traffic that comes to disk i generated by multicopy movies that also reside in disk j is $\sum_{m \in \Phi_i \cap \Phi_j} A_m/n_m$. It was shown in [11] that if a file allocation instance ideally attains CLB, the amount of $\text{type-}c$ movie traffic received by disk i from $\text{type-}c$ movies that also reside in disk j is given by

$$\frac{(c-1)\hat{A}_c}{J(J-1)}. \tag{7}$$

If each disk in the VOD system can accommodate at least one file copy of each movie title, a request for any movie title is blocked only if the stream capacity of all disks is used up upon the arrival of the request. Recall that no further performance gain can be obtained, even if multiple file copies of the same movie title are stored on a single disk. We would therefore be able to achieve the ideally minimal RBP of the system adequately by placing exactly one file copy of each movie title on each disk. Such a *full-replication* solution is extremely unlikely and unnecessary to be employed in practice. Nevertheless, the concept of full replication allows us to utilize the idea of disk resource sharing to design efficient performance indices that can be used to estimate the quality of a feasible file assignment.

6.1 Multicopy Traffic Index

The full-replication solution indicates an ideal file replication instance $\mathbf{n}$, where $n_m = J$, $\forall m \in \mathcal{F}$. Such an ideal file

replication instance has one unique file allocation instance that clearly attains CLB. Letting $c = J$ and $\hat{A}_c = A$ in (7), we readily find in this case that disk i receives a total of A/J movie traffic from all movies that are also stored on disk j.

Accordingly, we define a *multicopy traffic index* (MTI) given by

$$\mathrm{MTI}_{\langle \mathbf{n},\mathbf{\Omega},\mathbf{\Phi}\rangle} = \sqrt{\frac{2}{J(J-1)}\sum_{\substack{i,j\in\mathcal{D}\\ i<j}}\left(\sum_{m\in\Phi_i\cap\Phi_j}\frac{A_m}{n_m}-\frac{A}{J}\right)^2} \quad (8)$$

to estimate the quality of a file assignment solution $\langle \mathbf{n}, \mathbf{\Omega}, \mathbf{\Phi}\rangle$ on its assignment of multicopy movies as compared with the ideal full-replication solution. More explicitly, we measure for its file replication instance what proportion of the movie traffic is for multicopy movies and for its file allocation instance how evenly the multicopy movie traffic is shared within each pair of disks. A smaller value of MTI indicates a better file assignment solution regarding multicopy movies.

6.2 Single-Copy Traffic Index

The ideal full replication solution also indicates that there is no single-copy movie on any disk in the system. For a real file assignment solution $\langle \mathbf{n}, \mathbf{\Omega}, \mathbf{\Phi}\rangle$ that has single-copy movies, the traffic load of single-copy movies on disk j is given by $\sum_{m\in\Phi_j, n_m=1} A_m$.

We therefore define a *single-copy traffic index* (STI) given by

$$\mathrm{STI}_{\langle \mathbf{n},\mathbf{\Omega},\mathbf{\Phi}\rangle} = \sqrt{\frac{1}{J}\sum_{j\in\mathcal{D}}\left(\sum_{m\in\Phi_j,\ n_m=1}A_m\right)^2} \quad (9)$$

to estimate the quality of a file assignment solution $\langle \mathbf{n}, \mathbf{\Omega}, \mathbf{\Phi}\rangle$ on its assignment of single-copy movies as compared with the ideal full-replication solution that has $\mathbf{0}$ single-copy movie traffic on each disk. More explicitly, we measure for its file replication instance what proportion of the movie traffic is for single-copy movies and for its file allocation instance how evenly the single-copy movie traffic is distributed among the disks. A smaller value of STI indicates a better file assignment solution with respect to single-copy movies.

6.3 Conflicting Relationship between MTI and STI

At first glance, each of these two performance indices appears to indicate an attribute of an exclusive class of movie titles for a given file assignment solution. While MTI estimates the quality of a file assignment solution on its assignment of multicopy movies, STI evaluates the quality of the file assignment solution on its assignment of single-copy movies only. We therefore might hope that MTI and STI do not conflict with each other so that an optimal file assignment solution would always be associated with the best values in both MTI and STI. If that is the case, we would be able to form a composite index simply by adding MTI and STI. Such an easy-to-compute composite index would promise a substantial speedup of the SOEO presented in Section 5 by simply replacing the cumbersome RBP evaluation.

However, it needs to be noted that the movie access demand in the VOD system is highly skewed. Moreover, the movie file size is also significantly asymmetric. To achieve a good-quality allocation of multicopy movies, we may have to suffer a poor-quality allocation of single-copy movies to guarantee a feasible file assignment solution satisfying the disk storage space constraint. Consequently, there may exist a conflicting relationship between MTI and STI, as we will see in Section 8.3.

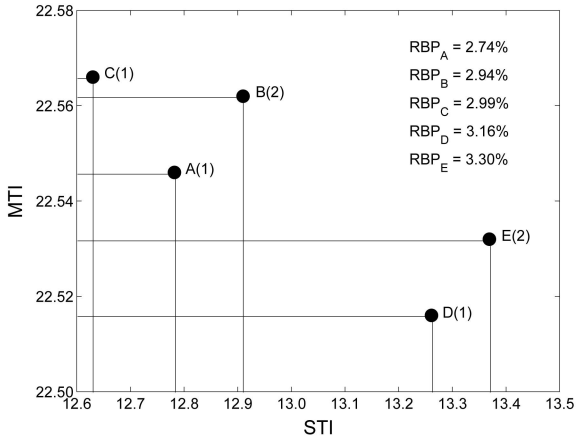

Fig. 5. Domination in MTI and STI among five file assignment solutions. A, C, and D are rank-1 solutions. B and E are rank-2 solutions.

6.4 Domination-Based Ranking

The concept of *domination* [12] has been conventionally used in the classification of solutions for a multiobjective optimization problem. Using this concept, solution x is said to dominate solution y if 1) x is not worse than y in all objectives and 2) x is strictly better than y in at least one objective. On the other hand, the two solutions are said to be nondominated solutions if neither can be said to dominate the other. The nondominated set of solutions among a set of solutions $\mathcal{P}$ therefore contains those chromosomes that are not dominated by any member of $\mathcal{P}$.

For a given $\mathcal{P}$, it is useful to classify the entire set of solutions into various nondomination levels. One effective way of doing this is by the method of *nondominated sorting* [28]. Using this method, solutions in $\mathcal{P}$ are sorted according to an ascending level of nondomination. The nondominated solutions are grouped as rank-1 solutions. Once all rank-1 solutions are identified, they are removed from the set. Similarly, the nondominated solutions of the remaining set of solutions are identified as rank-2 solutions and are subsequently removed from the set in order to find rank-3 solutions. This procedure is continued until all members of $\mathcal{P}$ are classified into a nondomination level. An illustrative example of how file assignment solutions can be classified using the concept of domination with respect to MTI and STI is shown in Fig. 5.

7 Multiobjective Evolutionary Optimization

Given MTI and STI that jointly estimate the quality of a file assignment solution, we can convert the transformed problem into the following multiobjective optimization problem:

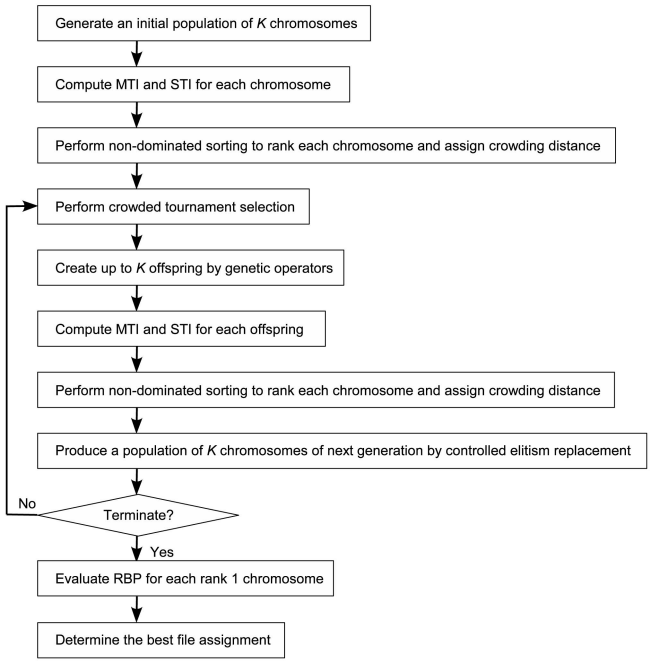

Fig. 6. Flowchart of the MOEO implementation.

$$\begin{aligned} &\text{Minimize} && \text{MTI}_{\langle \mathbf{n}, \mathbf{\Omega}^*, \mathbf{\Phi}^* \rangle} \\ &\text{Minimize} && \text{STI}_{\langle \mathbf{n}, \mathbf{\Omega}^*, \mathbf{\Phi}^* \rangle} \\ &\text{subject to} && n_m \in \{1, 2, \ldots, \varpi\},\ \forall m \in \mathcal{F} \\ & && \sum_{m \in \Phi_j^*} L_m \leq C,\ \forall j \in \mathcal{D}. \end{aligned}$$

The principle in solving a multiobjective optimization problem is to search for a set of Pareto-optimal solutions, including all the ultimate nondominated solutions [28], [29]. The curve formed by joining all Pareto-optimal solutions is known as the *Pareto-optimal front*. The goal of multiobjective optimization is thus to find a diverse set of nondominated solutions that converges as close as possible to the Pareto-optimal front.

Various implementations of multiobjective evolutionary algorithms exist in the literature [28], [29]. Recent studies have shown that elitism can significantly improve the performance of multiobjective evolutionary algorithms [30]. It has also been demonstrated that the newly developed NSGA-II outperforms all other popular elitist algorithms [31]. We therefore base our implementation of the multiobjective evolutionary optimization (MOEO) program in this paper on the controlled elitist NSGA-II [32]. This version of NSGA-II allows us to maintain a good population diversity through a *crowding distance* measure, which avoids the inconvenience of setting a sharing parameter commonly required in several early implementations of multiobjective evolutionary algorithms. In addition, it employs a *controlled elitism* mechanism, which largely improves the convergence capability of the algorithm toward the Pareto-optimal front.

7.1 Implementation

As the flowchart in Fig. 6 shows, we start by creating an initial population of size K using the ad hoc method. We identify the heuristic solution for each chromosome through the greedy file allocation method. If the chromosome is allocatable, we compute its MTI and STI by using (8) and (9). If no heuristic solution is found, we impose a death penalty by setting MTI of the chromosome to a sufficiently large value $\overline{\text{MTI}}$ such that no feasible solution can yield an MTI higher than $\overline{\text{MTI}}$. Similarly, we set the STI of the chromosome to a sufficiently large value $\overline{\text{STI}}$ such that no feasible solution can yield an STI higher than $\overline{\text{STI}}$. Note that in this context, it is sufficient to let $\overline{\text{MTI}} = A/J$ and $\overline{\text{STI}} = A/\sqrt{J}$. The nondominated sorting method classifies the entire population into ascending levels of nondominated solutions with respect to MTI and STI. A lower rank indicates a better front of nondominated solutions. For each solution within the same front, we further assign its crowding distance by using the procedure described in [28, p. 248]. This metric conveniently estimates the density of the neighborhood where a solution resides.

In each generation of MOEO, we use the crowded tournament selection operator [28] to select K solutions from the current population, some of which may be duplicate, to form the mating pool. For two chromosomes chosen for competition in a tournament, the winner is the one with a lower rank or the one with a larger crowding distance if both are of the same rank. These solutions are genetically varied through uniform crossover and uniform mutation operations, followed by the repair mechanism, if necessary, to correct any strictly nonallocatable chromosome into the region of likely allocatable file replication instances.

The mating pool and the offspring are combined into one transitional population. In the same way as for the initial population, we identify nondominated fronts of the transitional population and estimate the crowding distance of each solution. In order to prevent the algorithm from premature convergence to a suboptimal nondominated front, we use the controlled elitism mechanism presented in [28] to promote an adaptive selection of elitist solutions from each nondominated front in the transitional population and force these widely spread solutions to coexist in the new population for the next generation.

The evolution process is terminated after a predefined number of G generations have been completed. Alternatively, we presume convergence if the nondominated solutions are not improved over a certain succession of generations. In either case, we evaluate RBP for each rank-1 trade-off solution in the final population so that the global optimal file assignment solution can be ascertained. Ties are broken by choosing the solution that requires the smallest overall disk storage space.

8 Numerical Experiments

We have conducted extensive experiments to examine the performance of our proposed evolutionary approach. Here, we report numerical results from five test systems:

- TS-1: $J = 10$, and $M = 100$.
- TS-2: $J = 20$, and $M = 200$.
- TS-3: $J = 30$, and $M = 300$.
- TS-4: $J = 40$, and $M = 400$.
- TS-5: $J = 50$, and $M = 500$.

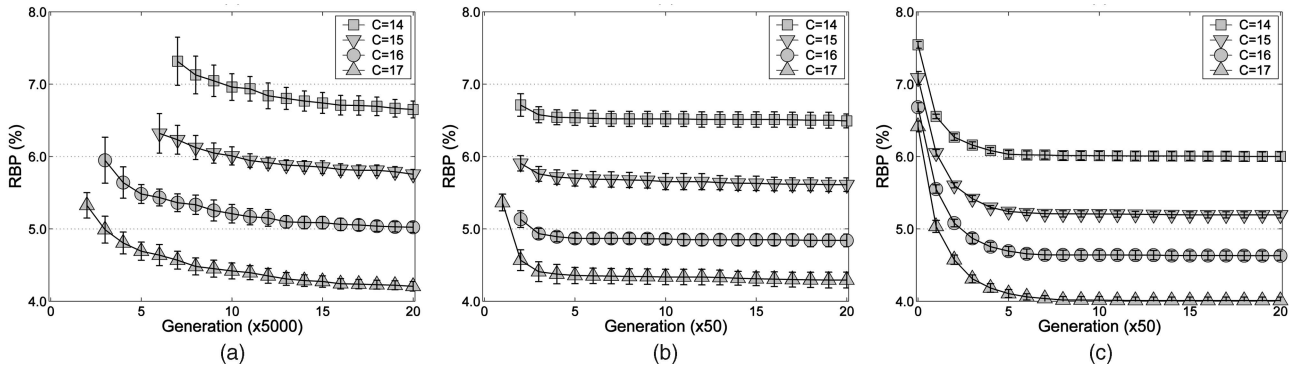

Fig. 7. Impact of problem transformation on SOEO-LBF. (a) Original problem. (b) Transformed problem with purely random population handling. (c) Transformed problem with ad hoc population handling.

Studies in [33] have shown that the performance of the evolutionary approach is not sensitive to the changes in the crossover rate σ and the mutation probability δ though with a slower convergence rate as δ increases. All experiments reported in this paper set $\sigma = 0.7$ and $\delta = 0.01$. The population size K is chosen in such a way that $\alpha \geq 99$ percent is ensured. Unless specified, all numerical results are presented in the form of an observed mean from 10 independent runs of the corresponding experiment. The radii of the 95 percent confidence intervals ([34, p. 273]) are also provided where possible.

8.1 Impact of Problem Transformation on SOEO-LBF

Our purpose with this experiment is to demonstrate the impact of problem transformation on file assignment optimization for the LBF system using SOEO (hence referred to as SOEO-LBF). As discussed in Section 4, the region of infeasible solutions to the file assignment problem can be rather large. To implement SOEO-LBF for the original problem, we need constraint handling techniques to guide the search direction of GAs from the large infeasible region toward the feasible region. The constraint handling method proposed in [23] can be used in this context. This approach efficiently handles constraints, without the need of setting any penalty parameter commonly required in conventional penalty-function-based constraint handling methods.

Let E_X define the amount of constraint violation on disk storage space by an infeasible chromosome X, which is given by

$$E_X = \sum_{j \in \mathcal{D}} \sum_{m \in \mathcal{F}} x_{m,j} L_m - JC. \tag{10}$$

Let $\bar{E}$ be a sufficiently large value such that no chromosome would require excessive disk storage space higher than $\bar{E}$. Note that it is sufficient to set $\bar{E} = J(L - C)$ for the file assignment problem.

If X is a feasible chromosome, we evaluate its fitness by

$$f(X) = \bar{E} + (1 - \mathrm{RBP}_X). \tag{11}$$

On the other hand, if X is an infeasible chromosome, we evaluate its fitness by

$$f(X) = \bar{E} - E_X. \tag{12}$$

In particular, if X violates (2), we impose a death penalty by setting $E_X = \bar{E}$. We also impose such a death penalty on X if $\sum_{m \in \mathcal{F}} x_{m,j} = 0$ for at least one j, $j \in \mathcal{D}$. Clearly, chromosomes with fitness higher than $\bar{E}$ are feasible solutions. With the fitness function defined in this form, we ensure for the original problem that 1) the fitness of a feasible chromosome is higher than that of an infeasible chromosome, 2) a comparison between two feasible chromosomes is purely based on RBP, and 3) a comparison between two infeasible chromosomes uses the information of constraint violation on disk storage space alone.

For the purpose of comparison, we use the same constraint handling method to implement SOEO-LBF for the transformed problem with purely random population handling. In this approach, the search space of GAs is the entire domain of file replication instances. Consequently, we again need constraint handling techniques to guide the search direction of GAs from the large region of strictly nonallocatable file replication instances toward the region of allocatable file replication instances.

The results in Fig. 7 are obtained from the test system TS-1. The disk model uses $N = 30$. The movie file size is randomly generated. The movie popularity distribution follows (1), with $\zeta = 0.271$. We consider four different cases by varying C from 14 to 17. With a smaller C, the region of infeasible solutions to the file assignment problem is larger, and the RBP of the LBF system is, in general, higher. The results in Fig. 7 confirm that the constraint handling method is effective in directing the search of GAs toward the feasible region of the original problem. However, we observe in all cases that SOEO-LBF for the transformed problem consistently outperforms that for the original problem. Even for such a small size system and even after 100,000 generations, the latter can only obtain solutions with RBP up to 11 percent higher than those achieved by the former in only 1,000 generations. Comparing between the two population handling approaches for the transformed problem, we see in all cases that the ad hoc approach converges to better quality solutions within fewer generations. The performance gain is more evident in large-sizes systems, as shown in Table 4.

8.2 Impact of ϖ on SOEO-LBF

It needs to be noted that SOEO-LBF can require considerable CPU time due to the cumbersome evaluation of RBP

TABLE 4
Efficiency of SOEO-STI when $\zeta = 0.271$

Case	RBP (%)					CPU Time (Hour)			
	SOEO-SRT ($G = 1000$)	SOEO-LBF ($G = 1000$)	MOEO ($G = 1000$)	MOEO ($G = 2000$)	SOEO-STI ($G = 1000$)	SOEO-LBF ($G = 1000$)	MOEO ($G = 1000$)	MOEO ($G = 2000$)	SOEO-STI ($G = 1000$)
TS-1	1.94 ± 0.22	0.83 ± 0.02	0.89 ± 0.05	0.83 ± 0.02	0.85 ± 0.02	0.40	9.62E-3	1.76E-2	5.88E-3
TS-2	2.00 ± 0.24	0.95 ± 0.02	1.25 ± 0.02	1.12 ± 0.03	0.97 ± 0.02	0.79	6.99E-2	1.28E-1	4.32E-2
TS-3	1.90 ± 0.08	1.00 ± 0.03	1.38 ± 0.02	1.26 ± 0.03	0.98 ± 0.02	1.39	1.62E-1	2.97E-1	1.05E-1
TS-4	-	0.90 ± 0.02	1.48 ± 0.05	1.37 ± 0.04	0.88 ± 0.02	5.27	5.73E-1	9.03E-1	2.77E-1
TS-5	-	1.06 ± 0.03	1.56 ± 0.04	1.49 ± 0.05	1.01 ± 0.02	7.91	8.18E-1	1.43	4.52E-1

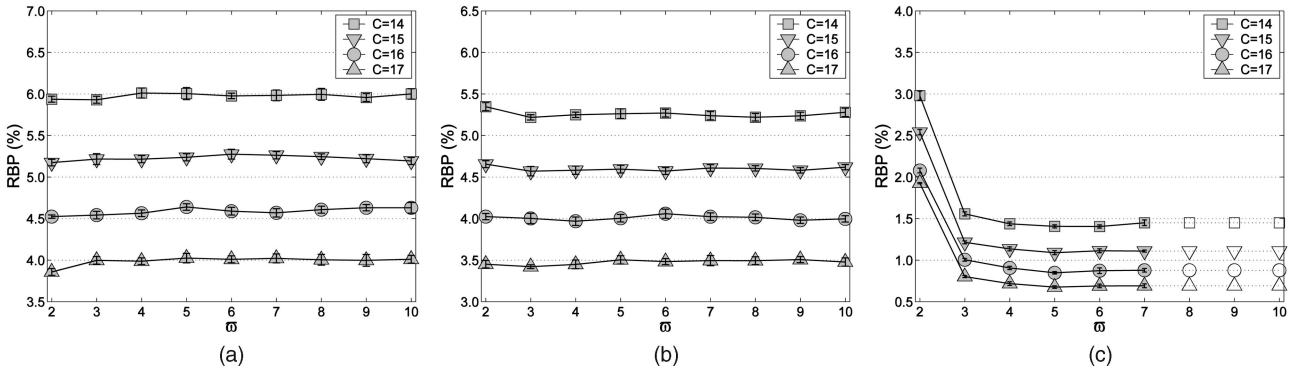

Fig. 8. Impact of ϖ on the RBP performance of SOEO-LBF. (a) $\zeta = 0.271$. (b) $\zeta = 0.4$. (c) $\zeta = 1$.

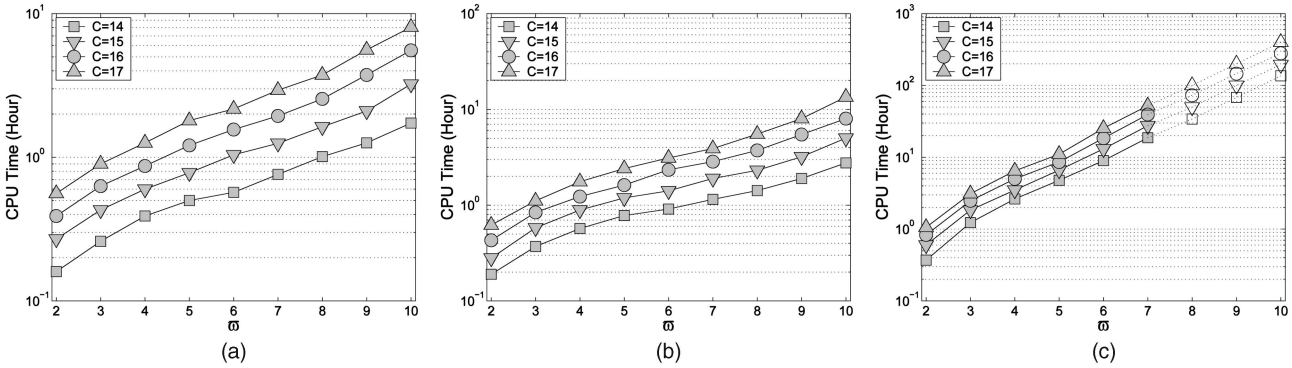

Fig. 9. Impact of ϖ on the runtime efficiency of SOEO-LBF. (a) $\zeta = 0.271$. (b) $\zeta = 0.4$. (c) $\zeta = 1$.

for each feasible solution explored in the evolution process. This is true, because the RBP calculation in (17) relies on an iterative process to obtain a fixed-point solution. It is also true due to the enumeration of the combination set $\Upsilon(\cdot)$ involved in (13) to estimate the reduced load request arrival rate for a multicopy movie m on each disk in the set Ω_m. Such an enumeration would be rather cumbersome if a movie title had a large number of file copies. On the other hand, it is important to observe in (18) the dramatic performance gain obtainable by movie m if its number of replicas increases from n_m to $n_m + 1$. Consequently, it is not necessary to place a large number of file copies for any movie title (even those popular ones) in a system of limited disk storage space.

This point is demonstrated by running SOEO-LBF for the same test system considered in Fig. 7, with ϖ varied in the range $[2, J]$. We also increase the skewness parameter ζ from 0.271 to 0.4 and then to 1. Let $A_{\max} = \max_{m \in \mathcal{F}} A_m$ and $A_{\min} = \min_{m \in \mathcal{F}} A_m$. With a higher value of ζ, the ratio $A_{\max}/A_{\min}$ is larger. For this particular test system, $A_{\max}/A_{\min} = 3.98$, 6.92, and 100.54 when $\zeta = 0.271$, 0.4, and 1, respectively. Fig. 8 presents the RBP results of the optimal solutions after 1,000 generations. Fig. 9 provides the average CPU time required on a 3.0-GHz Pentium 4 machine for the 10 independent runs of SOEO-LBF in each experiment. The quantities are plotted on a logarithmic scale due to the large measured range.

These results confirm that SOEO-LBF with a large ϖ converges to solutions of statistically equivalent quality as those obtained from SOEO-LBF with a small ϖ, especially when ζ is not large. Even when $\zeta = 1$, it is sufficient to restrict ϖ to 5 in all cases, with different C values. On the other hand, with large ϖ, SOEO-LBF requires considerable CPU time. This is mainly due to the cumbersome RBP evaluation for feasible chromosomes typically having a large number of file copies for certain movie titles. It is also attributed to the large population size required to ensure that $\alpha \geq 99$ percent, as shown in Table 3. Note that even for such a small size system, we have only managed to run SOEO-LBF for $\zeta = 1$, with ϖ up to 7. This is ascribed to the

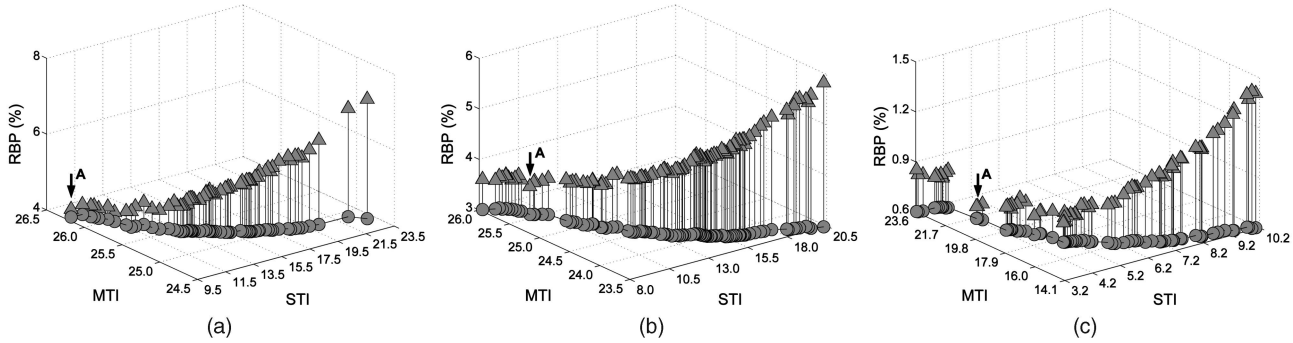

Fig. 10. Rank-1 solutions in the last population of MOEO. (a) $\zeta = 0.271$. (b) $\zeta = 0.4$. (c) $\zeta = 1$. The solution with a minimal RBP is marked by "A."

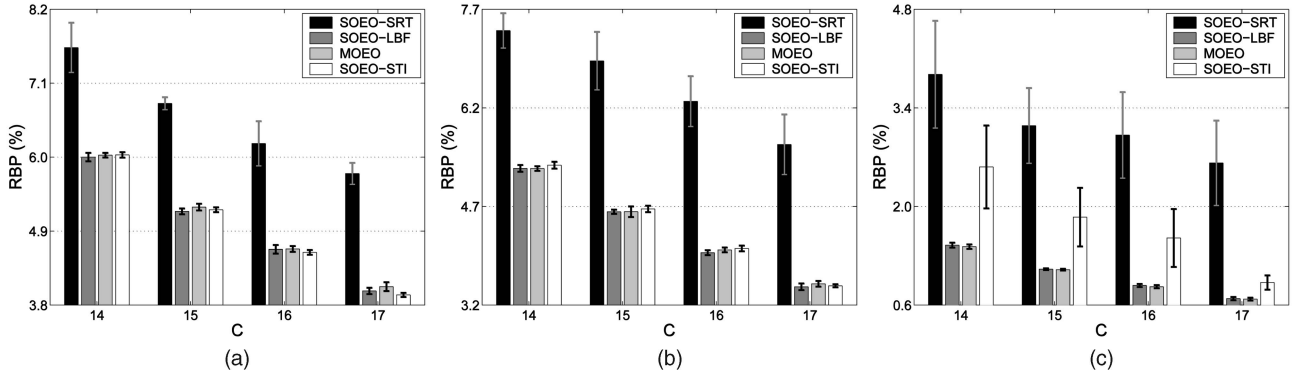

Fig. 11. Performance comparison among SOEO-SRT, SOEO-LBF, MOEO, and SOEO-STI. (a) $\zeta = 0.271$. (b) $\zeta = 0.4$. (c) $\zeta = 1$.

exponential growth of CPU time, as can be observed in Fig. 9.

8.3 Efficiency of Performance Indices

We now investigate the efficiency of MTI and STI when applied to solve the file assignment problem using MOEO. Our particular interest is to see if MOEO can obtain solutions of comparable quality but require less CPU time when compared to SOEO-LBF. We again use the test system TS-1 for this purpose. We run MOEO for 1,000 generations in each experiment, with a particular value on C and ζ that we have considered before. We set ϖ to J in this context, since the evaluation of the two easy-to-compute performance indices does not involve the enumeration of any combination set, as done with the RBP calculation.

The widely spread nondominated front of rank-1 solutions in the last population, as depicted in each plot in Fig. 10 for $C = 14$, clearly indicates a conflicting relationship between MTI and STI. It is interesting to observe that when $\zeta = 0.271$, the solution with a minimal RBP (marked by "A" in the plot) turns out to be the one with the smallest STI. This indicates that in situations where ζ is not large, it could be sufficient to use SOEO with STI alone (i.e., SOEO-STI) to find good-quality file assignments. This observation is confirmed in Fig. 11, where we compare the solution quality among SOEO-LBF, MOEO, and SOEO-STI. For SOEO-LBF, we use the results presented in Fig. 8 at $\varpi = 10$ for $\zeta = 0.271$ and $\zeta = 0.4$ and the ones at $\varpi = 7$ for $\zeta = 1$, since we have not managed to obtain results with a larger ϖ due to the exponential growth of CPU time. The results in Fig. 11 demonstrate the sufficient accuracy of MOEO in all cases. Despite the failure in all cases when $\zeta = 1$, SOEO-STI is sufficiently accurate for $\zeta = 0.271$ and $\zeta = 0.4$.

Such an interesting property of STI is essentially due to the disk storage space constraint and the nature of the greedy file allocation method. With limited disk storage space, the greater the number of file copies replicated for certain movie titles, the greater the number of other movie titles that would have only one single file copy. We have seen in Fig. 8 that when ζ is not large, the optimal solutions are, in general, those where no movie title has more than two file copies. This results in fewer single-copy movies and, hence, less proportion of the movie traffic for single-copy movies. Consequently, such solutions are, in general, associated with the smallest STI, provided that the single-copy movie traffic is evenly distributed among the disks, which is more or less ensured by the greedy file allocation method [11].

This property of STI is further justified in Table 4 by the extensive numerical results obtained from the five test systems for $\zeta = 0.271$. In all cases, the disk model uses $C = 14$ and $N = 30$. The movie file size is randomly generated. Due to the exponential growth of CPU time required for a large ϖ, we are not able to carry out SOEO-LBF experiments for the large size test systems if we shall set $\varpi = J$. For the purpose of performance comparison, we have chosen to set $\varpi = \min(J, \lceil A_{\max}/A_{\min} \rceil)$. This leads to $\varpi = 4$ for TS-1, $\varpi = 6$ for TS-2 and TS-3, and $\varpi = 8$ for TS-4 and TS-5. This is a conservative upper bound, since we have seen in Fig. 8 that, even if $A_{\max}/A_{\min} = 100.54$ when $\zeta = 1$, it is sufficient to restrict ϖ to 5. Table 4 again confirms the efficiency of MTI and STI. Although MOEO converges slowly in the large size test systems and hence requires more generations to obtain

comparable solutions, SOEO-STI succeeds in finding solutions of statistically equivalent quality within the same number of generations but with significantly reduced CPU time by up to two orders of magnitude when compared to SOEO-LBF.

Finally, we implement SOEO-SRT by using the analytical model of the SRT system presented in [10] to demonstrate the inefficiency of SRT. We run SOEO-SRT for the transformed problem with purely random population handling. For all the experiments presented in Fig. 11 and Table 4, except those for TS-4 and TS-5, SOEO-SRT indeed obtains optimal solutions, which achieve disk load balancing in the context of the SRT system. However, such solutions yield very poor RBP performance in the more realistic LBF system. A large variation can be seen in Fig. 11 when $\zeta = 1$. Even in cases where $\zeta = 0.271$, we see in Table 4 that the RBP performance can be up to twice as bad. Moreover, with purely random population handling, SOEO-SRT cannot even find a feasible solution after 1,000 generations for TS-4 and TS-5. This again demonstrates the robustness of our proposed evolutionary approach by operating GAs solely within the search space of likely allocatable file replication instances to this difficult file assignment problem.

9 Conclusions

In this paper, we have presented an evolutionary approach for tackling a realistic and challenging file assignment problem for a large-scale VOD system. With the design goal of a computationally efficient stochastic search using GAs, we have proposed an elaborate transformation method for the problem. This essentially relies on a good performance greedy file allocation method so that a divide-and-conquer strategy can be adopted and ad hoc population handling methods can be designed to operate GAs within a drastically reduced search space and yet obtain good-quality file assignment solutions. We have further proposed two easy-to-compute performance indices, namely, MTI and STI. These two attributes can jointly estimate the quality of a file assignment solution with respect to RBP. By means of MTI and STI, we are able to circumvent the cumbersome RBP evaluation in each cycle of the evolution process and yet obtain file assignment solutions of comparable quality. We have also observed that in situations where the skewness of the movie popularity distribution is not large, the proposed evolutionary approach can be made more efficient by using STI alone, since it indicates with sufficient accuracy the optimal solutions to the file assignment problem in such circumstances.

Appendix

An accurate approximate analysis for RBP evaluation in an LBF system using the fixed-point method was presented in [10]. For the purpose of this paper, we briefly describe here how we derive the set of fixed-point equations.

Let $\xi_j^{(i)}$ be the stationary probability that disk j is in state i, or in other words, it delivers i video streams concurrently. Define $\vec{\xi}_j = \left(\xi_j^{(0)}, \xi_j^{(1)}, \ldots, \xi_j^{(N)}\right)$ and $\vec{\xi} = \left(\vec{\xi}_1, \vec{\xi}_2, \ldots, \vec{\xi}_J\right)$. Let $y_j^{(i)}(m)$ be the rate of requests for movie m, $m \in \Phi_j$, given that disk j is in state i. Let $y_j^{(i)}$ be the rate of requests for all movie files in Φ_j when disk j is in state i. We have $y_j^{(i)} = \sum_{m \in \Phi_j} y_j^{(i)}(m)$.

The state-transition process of disk j is modeled as a birth-death process [35], with the birth rate $y_j^{(i)}$, $i = 0, 1, \ldots, N-1$, and the death rate $i\hat{\mu}_j^{(i)}$, $i = 1, 2, \ldots, N$. For disk j, $j = 1, 2, \ldots, J$, the reduced load request arrival rate $y_j^{(i)}$ when disk j is in state i is given by

$$y_j^{(i)} = \sum_{m \in \Phi_j} \sum_{h=1}^{n_m} \left(\frac{\lambda p_m}{h}\right) \sum_{\mathcal{S} \in \Upsilon(\Omega_m - \{j\},\, h-1)} \prod_{u \in \mathcal{S}} \xi_u^{(i)} \cdot \prod_{v \in \Omega_m - \{j\} - \mathcal{S}} \sum_{k=i+1}^{N} \xi_v^{(k)}, \tag{13}$$

where $\Upsilon(\Omega_m - \{j\}, h-1)$ defines the set of all possible combinations of choosing $h-1$ disks out of $\Omega_m - \{j\}$, given that disk j is in Ω_m. We also have

$$\frac{1}{\hat{\mu}_j^{(i)}} = \frac{1}{y_j^{(i-1)}} \sum_{m \in \Phi_j} \frac{y_j^{(i-1)}(m)}{\mu_m} \tag{14}$$

and

$$\xi_j^{(i)} = \frac{N!}{i! \prod_{k=i}^{N-1} \frac{y_j^{(k)}}{\hat{\mu}_j^{(k+1)}}} \xi_j^{(N)}, \tag{15}$$

where

$$\sum_{i=0}^{N} \xi_j^{(i)} = 1. \tag{16}$$

The system of (13), (14), (15), and (16) forms a set of fixed-point equations:

$$\vec{\xi} = f(\vec{\xi}), \tag{17}$$

which can often be solved efficiently by the successive substitution method [36]. By the disk independence assumption [10] and having obtained $\vec{\xi}$ by solving (17), the RBP of the LBF system is computed by

$$\mathrm{RBP}_{\langle \mathbf{n}, \mathbf{\Omega}, \mathbf{\Phi} \rangle} = \sum_{m \in \mathcal{F}} p_m \prod_{j \in \Omega_m} \xi_j^{(N)}. \tag{18}$$

Acknowledgments

The authors would like to thank the anonymous referees for their valuable comments. The work described in this paper was partially supported by grants from the City University of Hong Kong (project No. 7001458) and partially supported by the Australian Research Council (ARC). Part of the work presented in this paper was done while Jun Guo and Yi Wang were visiting the Department of Electronic Engineering, City University of Hong Kong.

References

[1] W.D. Sincoskie, "System Architecture for a Large-Scale Video-on-Demand Service," *Computer Networks and ISDN Systems*, vol. 22, no. 2, pp. 155-162, 1991.

[2] T.D.C. Little and D. Venkatesh, "Popularity-Based Assignment of Movies to Storage Devices in a Video-on-Demand System," *Multimedia Systems,* vol. 2, pp. 280-287, Jan. 1995.
[3] A.N. Mourad, "Issues in the Design of a Storage Server for Video-on-Demand," *Multimedia Systems,* vol. 4, pp. 70-86, 1996.
[4] D.N. Serpanos, L. Georgiadis, and T. Bouloutas, "MMPacking: A Load and Storage Balancing Algorithm for Distributed Multimedia Servers," *IEEE Trans. Circuits and Systems for Video Technology,* vol. 8, no. 1, pp. 13-17, Feb. 1998.
[5] K.S. Tang, K.T. Ko, S. Chan, and E. Wong, "Optimal File Placement in VOD System Using Genetic Algorithm," *IEEE Trans. Industrial Electronics,* vol. 48, no. 5, pp. 891-897, Oct. 2001.
[6] Y.W. Leung and R.Y.T. Hou, "Assignment of Movies to Heterogeneous Video Servers," *IEEE Trans. Systems, Man, and Cybernetics A,* vol. 35, no. 5, pp. 665-681, Sept. 2005.
[7] J.L. Wolf, P.S. Yu, and H. Shachnai, "Disk Load Balancing for Video-on-Demand Systems," *Multimedia Systems,* vol. 5, no. 6, pp. 358-370, Nov. 1997.
[8] S.L. Tsao, M.C. Chen, M.T. Ko, J.M. Ho, and Y.M. Huang, "Data Allocation and Dynamic Load Balancing for Distributed Video Storage Server," *J. Visual Comm. and Image Representation,* vol. 10, no. 2, pp. 197-218, 1999.
[9] Y. Zhao and C.C.J. Kuo, "Video-on-Demand Server System Design with Random Early Migration," *Proc. IEEE Int'l Symp. Circuits and Systems (ISCAS '03),* vol. 2, pp. 640-643, May 2003.
[10] J. Guo, E.W.M. Wong, S. Chan, P. Taylor, M. Zukerman, and K.S. Tang, "Performance Analysis of Resource Selection Schemes for a Large-Scale Video-on-Demand System," *IEEE Trans. Multimedia,* vol. 10, pp. 153-159, 2008.
[11] J. Guo, E.W.M. Wong, S. Chan, P. Taylor, M. Zukerman, and K.S. Tang, "Combination Load Balancing for Video-on-Demand Systems," to be published in *IEEE Trans. Circuits and Systems for Video Technology.*
[12] D.E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley, 1989.
[13] M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W.H. Freeman, 1979.
[14] T. Ibaraki and N. Katoh, *Resource Allocation Problems: Algorithmic Approaches.* MIT Press, 1988.
[15] D. Sitaram and A. Dan, *Multimedia Servers: Applications, Environments and Design.* Morgan Kaufmann, 2000.
[16] R. Zimmermann and S. Ghandeharizadeh, "Highly Available and Heterogeneous Continuous Media Storage Systems," *IEEE Trans. Multimedia,* vol. 6, no. 6, pp. 886-896, Dec. 2004.
[17] S.C. Liew and D.C.-Y. Tse, "A Control-Theoretic Approach to Adapting VBR Compressed Video for Transport over a CBR Communications Channel," *IEEE/ACM Trans. Networking,* vol. 6, no. 1, pp. 42-55, Feb. 1998.
[18] M. Krunz, R. Sass, and H. Hughes, "Statistical Characteristics and Multiplexing of MPEG Streams," *Proc. IEEE INFOCOM '95,* vol. 2, pp. 455-462, Apr. 1995.
[19] C.P. Costa, I.S. Cunha, A. Borges, C.V. Ramos, M.M. Rocha, J.M. Almeida, and B. Ribeiro-Neto, "Analyzing Client Interactivity in Streaming Media," *Proc. 13th Int'l Conf. World Wide Web (WWW '04),* pp. 534-543, 2004.
[20] A. Dan, D. Sitaram, and P. Shahabuddin, "Scheduling Policies for an On-Demand Video Server with Batching," *Proc. Second ACM Int'l Conf. Multimedia (Multimedia '94),* pp. 15-23, 1994.
[21] G.K. Zipf, *Human Behavior and the Principle of Least Effort: An Introduction to Human Ecology.* Addison-Wesley, 1949.
[22] P. Branch, G. Egan, and B. Tonkin, "Modeling Interactive Behavior of a Video-Based Multimedia System," *Proc. IEEE Int'l Conf. Comm. (ICC '99),* vol. 2, pp. 978-982, June 1999.
[23] K. Deb, "An Efficient Constraint Handling Method for Genetic Algorithms," *Computer Methods in Applied Mechanics and Eng.,* vol. 186, no. 2-4, pp. 311-338, June 2000.
[24] J.H. Holland, *Adaptation in Natural and Artificial Systems.* Univ. of Michigan Press, 1975.
[25] C.R. Reeves, "Using Genetic Algorithms with Small Populations," *Proc. Fifth Int'l Conf. Genetic Algorithms (ICGA '93),* pp. 92-99, 1993.
[26] *Handbook of Mathematical Functions with Formulas, Graphs and Mathematical Tables,* M. Abramowitz and I.A. Stegun, eds. Dover, 1972.
[27] D.E. Goldberg and K. Deb, "A Comparative Analysis of Selection Schemes Used in Genetic Algorithms," *Proc. First Workshop Foundations of Genetic Algorithms (FOGA '91),* vol. 1, pp. 69-93, 1991.
[28] K. Deb, *Multi-Objective Optimization using Evolutionary Algorithms.* John Wiley & Sons, 2001.
[29] C.A.C. Coello, D.A.V. Veldhuizen, and G.B. Lamont, *Evolutionary Algorithms for Solving Multiobjective Problems.* Kluwer Academic Publishers/Plenum, 2002.
[30] E. Zitzler, K. Deb, and L. Thiele, "Comparison of Multiobjective Evolutionary Algorithms: Empirical Results," *Evolutionary Computation,* vol. 8, no. 2, pp. 173-195, 2000.
[31] K. Deb, A. Pratap, S. Agrawal, and T. Meyarivan, "A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II," *IEEE Trans. Evolutionary Computation,* vol. 6, no. 2, pp. 182-197, Apr. 2002.
[32] K. Deb and T. Goel, "Controlled Elitist Non-Dominated Sorting Genetic Algorithms for Better Convergence," *Proc. First Int'l Conf. Evolutionary Multi-Criterion Optimization (EMO '01),* pp. 67-81, Mar. 2001.
[33] J. Guo, "Two Problems in Stochastic Service Systems," PhD dissertation, The Univ. of Melbourne, 2006.
[34] S.K. Bose, *An Introduction to Queueing Systems.* Kluwer Academic Publishers/Plenum, 2002.
[35] H. Akimaru and K. Kawashima, *Teletraffic: Theory and Applications,* second ed. Springer-Verlag, 1999.
[36] W.J. Stewart, *Introduction to the Numerical Solution of Markov Chains.* Princeton Univ. Press, 1994.

Jun Guo received the BE degree in automatic control engineering from Shanghai University of Science and Technology, Shanghai, in 1992 and the ME degree in telecommunications engineering and the PhD degree in electrical and electronic engineering from the University of Melbourne, Melbourne, in 2001 and 2006, respectively. From 2003 to 2004, he was a research associate in the Department of Electronic Engineering, City University of Hong Kong. Since 2005, he has been with the Networks Research Group, School of Computer Science and Engineering, The University of New South Wales. His research interests include multicast in wired/wireless networks. He is a member of the IEEE.

Yi Wang received the BE degree from South China University of Technology, Guangzhou, China, in 2002, and the ME degree in telecommunications engineering from the University of Melbourne, Melbourne, in 2003. She is currently working toward the PhD degree in the School of Computer Science and Information Technology, RMIT University. From 2003 to 2004, she was a research assistant with the Department of Electrical and Electronic Engineering, The University of Melbourne. In 2004, she was a research associate in the Department of Electronic Engineering, City University of Hong Kong, Hong Kong. Her research interests include biometric recognition and indexing techniques. She is a student member of the IEEE.

Kit-Sang Tang received the BSc degree from the University of Hong Kong in 1988 and the MSc and PhD degrees from the City University of Hong Kong in 1992 and 1996, respectively. He is currently an associate professor in the Department of Electronic Engineering, City University of Hong Kong. From 2004 to 2005, he was the associate editor for the *IEEE Transactions on Circuits and Systems Part II.* He is currently the associate editor for the *Dynamics of Continuous, Discrete and Impulsive Systems Series B.* He has published more than 60 journal papers and four book chapters and is a coauthor of two books, focusing on genetic algorithms and chaotic theory. He is a member of the IEEE, the Nonlinear Circuits and Systems Technical Committee of the IEEE Circuits and Systems Society, and the Technical Committee on Optimal Control of the IFAC.

Sammy Chan received the BE and MEngSc degrees in electrical engineering from the University of Melbourne, Melbourne, in 1988 and 1990, respectively, and the PhD degree in communication engineering from the Royal Melbourne Institute of Technology in 1995. He was with the Telecom Australia Research Laboratories as a research engineer from 1989 to 1992 and as a senior research engineer and project leader from 1992 to 1994. Since December 1994, he has been with the Department of Electronic Engineering, City University of Hong Kong, where he is currently an associate professor. He is a member of the IEEE.

Eric W.M. Wong received the BSc and MPhil degrees in electronic engineering from the Chinese University of Hong Kong, Hong Kong, in 1988 and 1990, respectively, and the PhD degree in electrical and computer engineering from the University of Massachusetts, Amherst, in 1994. In 1994, he joined the City University of Hong Kong, where he is currently an associate professor in the Department of Electronic Engineering. His most notable research work involves the first workable model on state-dependent dynamic routing. Since 1991, the model has been used by AT&T to design and dimension its telephone network that uses real-time network routing. His research interests include the analysis and design of telecommunications networks, optical burst switching, and video on demand. He is a senior member of the IEEE.

Peter Taylor received the BSc (Hons) and PhD degrees in applied mathematics from the University of Adelaide in 1980 and 1987, respectively. After periods at the universities of Western Australia and Adelaide, he moved, at the beginning of 2002, to Melbourne. In January 2003, he was the inaugural professor of operations research at the University of Melbourne. He has been active on the organizing committees of many conferences, is the editor in chief of *Stochastic Models* and is an associate editor for *Queueing Systems*. He is one of the chief investigators of the Australian Research Council (ARC) Centre of Excellence in Mathematical and Statistical Modelling of Complex Systems. In addition, he was a key researcher in the CRC for Smart Internet Technology and has been the recipient of nine large grants from the ARC. His research interests include stochastic processes, applied probability, in particular applications in telecommunications, and the interaction of stochastic modeling with optimization and optimal control. He has published around 85 papers in internationally refereed journals and approximately 20 technical reports dealing with topics such as the theory of Markov chains, insensitivity theory, queuing networks, loss networks, matrix-analytic methods, network optimization, and stochastic Petri nets. In addition, he has several papers on the performance analysis and control of telecommunications systems. He has coauthors from 10 countries of the five continents.

Moshe Zukerman received the BSc degree in industrial engineering and management and the MSc degree in operations research from Technion—Israel Institute of Technology and the PhD degree in electrical engineering from the University of California, Los Angeles (UCLA) in 1985. He was an independent consultant with the IRI Corp. and a postdoctoral fellow at UCLA from 1985 to 1986. He was with Telstra Research Laboratories (TRL) as a research engineer from 1986 to 1988 and as a project leader from 1988 to 1997. From 1990 to 2001, he taught and supervised graduate students at Monash University. In 1997, he joined the University of Melbourne, where he is currently a professor and is responsible for promoting and expanding telecommunications research and teaching in the Electrical and Electronic Engineering Department. He served on the editorial boards of the *Australian Telecommunications Research Journal*, *Computer Networks*, and the *IEEE Communications Magazine*. He also served as a guest editor of two issues of the *IEEE Journal on Selected Areas in Communications*. He currently serves on the editorial boards of the *IEEE/ACM Transactions on Networking* and the *International Journal of Communication Systems*. He has more than 200 publications in scientific journals and conference proceedings. He is a coauthor of two award-winning conference papers. He is a fellow of the IEEE. He is the recipient of the 1990 Telstra Research Laboratories Outstanding Achievement Award.