Journal of ELECTRICAL ENGINEERING, VOL. 54, NO. 9-10, 2003, 244–249

JOINT OPTIMIZATION OF SERVER LOCATION AND STORAGE ALLOCATION IN MULTIMEDIA–ON–DEMAND NETWORK

Sun-Jin Kim — Mun-Kee Choi [*]

In this paper, we propose a genetic algorithm (GA) to design a non-hierarchical and decentralized Multimedia-On-Demand (MOD) network architecture. To optimize the MOD network resource based on cost analysis including installation cost for servers, program storage cost, and transmission cost, the joint optimization of both server location and storage allocation was considered. In order to improve solution quality and computational efficiency in applying the proposed methods to the problem, genetic representation, evaluation function, genetic operators and procedure are devised. The results of extensive computational simulations showed that the proposed algorithm provided high quality solutions within a reasonable amount of computation time.

K e y w o r d s: joint optimization, genetic algorithm, multimedia-on-demand, location-allocation problem

1 INTRODUCTION

Multimedia-On-Demand (MOD), which includes wider concept than Video-On-Demand (VOD) even if VOD has used pervasively until now, is an interactive service that provides various contents and programs to users connected to a network anywhere and anytime. It is generally recognized that MOD service will cover a large region to provide many of interactive multimedia services. However, this service will require a high-bandwidth network and a large amount of storage capacity. So far different aspects of VOD optimization have been discussed in [1–6]. Various topologies for VOD networks are proposed, which can be classified either into hierarchical [2, 3, 6, 7] or non-hierarchical architecture [4, 8].

In most researches, storage allocation problems have been studied under the constraint of storage cost and network cost (*ie* communication cost). This problem in [4] was also investigated by the decomposition procedure in which an original problem is divided into smaller problems. In [8], the problem of server location and storage allocation in non-hierarchical MOD network has been studied. In this paper, modifying an approach used in [8], we deal with the joint optimization of both server location and storage allocation subject to the tradeoffs among installation cost for servers, program storage cost, and transmission cost. The objective of this problem is to minimize the sum of the involved costs. However, the problem cannot be solved exactly within reasonable computation time, even for a moderate number of variables. To solve fairly large-sized problems, we employ a heuristic method called genetic algorithm (GA), which has been proven to be efficient and powerful in a wide variety of combinatorial optimization problems and applications [9, 10]. In order to for GA to find a high quality solution of the problem considered, we devise the following components: genetic representation, evaluation function, genetic operators and procedure, which are critical for solution quality and computational efficiency in applying the proposed GA. Extensive computational simulations are carried out on various bench-marking problems, showing that the proposed method provide high quality solutions within reasonable time periods of computations.

2 THE PROBLEM DESCRIPTION AND FORMULATION

Consider a non-hierarchical and decentralized architecture for a MOD network as shown in Fig. 1, where we assume.

(a) The network consists of n interconnected central offices (COs) and every MOD subscriber is connected to a CO (usually, the closest one).
(b) For each CO, at most one sever can be installed, so that the total number of server is less than or equal to number of COs. COs without server play a role of switching the required service between subscribers and servers.
(c) The installation cost of a server in any CO is same.
(d) Transmission and storage cost are linearly proportioned to the transmission distance and amount of data, respectively.
(e) Storage capacity in a CO has a limitation.
(f) Link capacity for MOD service between two COs are unlimited.
(g) The expected demand of every program at each CO is evaluated on the basis of the expected number of the subscribers and program preference probability.

[*] School of Business, Information and Communications University, 58-4, Hwaam-Dong, Yuseong-Gu, Daejeon, Republic of Korea, 305-732, E-mails: kimsj@icu.ac.kr, mkchoi@icu.ac.kr

First version of this paper was presented at SympoTIC ´03, 26-28 October 2003, Bratislava, Slovakia

ISSN 1335-3632 © 2003 FEI STU

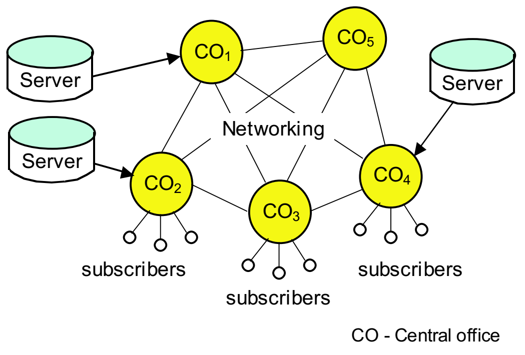


Fig. 1. Non-hierarchical MOD network

With a given MOD network, three kinds of network cost, *ie*, server installation cost, storage cost and transmission cost, are considered. If the installation cost is much lower than the other costs, server may be located at every CO and then the program requested can be served by the server to which the requester is connected. On the contrary, if the installation cost is much higher than the other costs, it is beneficial to install only a few servers and to transmit programs to the requester from one of servers that store them. Therefore, it is necessary to design MOD network architecture considering tradeoffs between the network costs.

In order to formulate this problem, the following notations and decision variables are introduced:

Notations

K	: number of total programs for service
k	: index of MOD programs $k = 1, \dots, K$
i, j	: index of central office $i, j = 1, \dots, n$
C_F	: installation cost of a single server
C_S	: storage cost per program in a single server
C_{ij}	: transmission cost per program between CO i and CO j
$D_k(i)$	: expected demand of program k in CO i
M	: number of multiple accesses
$[x]^+$	: smallest integer greater than or equal to x

Decision variables

$x_k(i,j)$: amount of transmission for program k from CO i to CO j

$$y(i) = \begin{cases} 1 & \text{if a server is installed in CO } i, \\ 0 & \text{otherwise.} \end{cases}$$

Minimize

$$\sum_{i=1}^{n} C_F \cdot y(i) + \sum_{i=1}^{n}\sum_{k=1}^{K}\Big\{C_s \cdot \Big[\sum_{j=1}^{n} x_k(i,j)/M\Big]^+\Big\} + \sum_{i=1}^{n}\sum_{k=1}^{K}\sum_{\substack{j=1\\ j\neq i}}^{n} C_{ij} \cdot x_k(i,j) \qquad (0)$$

Subject to

$$\sum_{i=1}^{n} x_k(i,j) = D_k(j)\,,\ k = 1, \dots, K\,,\ j = 1, \dots, n \quad (1)$$

$$\sum_{k=1}^{K}\Big[\sum_{j=1}^{n} x_k(i,j)/M\Big]^+ \le C(i)\,, \quad i = 1, \dots, n \quad (2)$$

$$y(i) = 0 \ \text{ or } \ 1 \qquad i = 1, \dots, n \quad (3)$$

$$x_k(i,j) \ge 0\,,\ \text{and Integer } k = 1, \dots, K \quad i = 1, \dots, n \ \ j = 1, \dots, n \quad (4)$$

The objective of above formulation is to minimize the sum of the costs considered. The first, second, and third term in the objective function represent respectively the server installation cost, program storage cost in servers, and program transmission cost between COs. The constraint (1) ensures that the expected demands of all the programs are met. The constraint (2) describes that storage amount in each CO should be restricted to its capacity. The constraints (3) and (4) just restate the definitions of the decision variables involved.

3 THE PROPOSED GENETIC ALGORITHM

The problem considered belongs to the location-allocation problem. If a vector of location variables $y(i)$ is specified, the problem is reduced to the transportation problem so that we easily obtain optimal values of allocation variables $x_k(i,j)$ using well-known methods. Therefore, the search procedure can be restricted to the space of location variables. In Section 3, when we refer to a solution, this should be interpreted as an assignment of 0s and 1s to the location variables. To solve this problem, a genetic algorithm (GA) is employed.

The GA is a stochastic procedure that imitates the biological evolutionary process of genetic inheritances and the survival of the fittest [9, 10]. The GA is an iterative procedure. During each iteration, a finite set, called a population, of individuals is maintained. Each individual represents a potential solution to the problem. The fitness of each individual is measured according to an evaluation function (evaluation step). Then, a new population is formed by selecting fitter individuals (selection step). Some individuals of the new population are altered by applying genetic operators (recombination step). The above process is repeated until some termination criteria are met.

In order to design a GA for a particular problem, the following components must be determined: 1) a genetic representation for potential solutions to the problem; 2) an evaluation function to compute the fitness of each solution; 3) a selection scheme to generate a new population; 4) genetic operators to alter the individuals; 5) and various genetic parameters to control the evolutionary process.

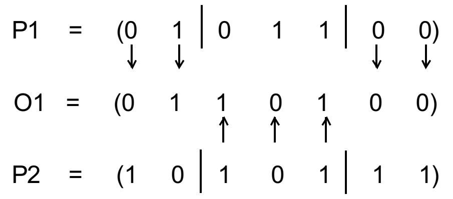


Fig. 2. Two-point crossover

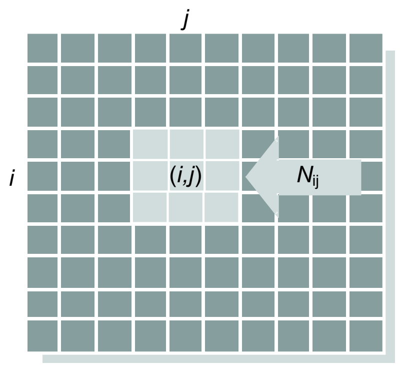


Fig. 3. Population and neighborhood structure

```
begin
  t    0;
  begin
    initialize P(t);
    evaluate P(t);
  end
  while (not termination condition) do
    begin
      set N(t) by selecting arbitrary area from P(t);
      select N'(t) for reproduction;
      recombine N'(t) to yield C(t) by crossover;
      evaluate C(t) and replace Ni(t) with C(t);
      apply mutation for N(t) to yield C(t);
      evaluate C(t) and replace N'(t) with C(t);
      t    t+1
    end
```

Fig. 4. The proposed procedure of GA

The representation is an encoding of potential solutions to the problem. The evaluation function reflects the objective of the optimization. Two kinds of genetic operators are used in most GAs: crossover operators that combine genetic material from two individuals, and mutation operators which randomly alter some composition of an individual. These operators, in conjunction with the representation schemes, greatly influence the performance of GAs. The genetic parameters are experimentally determined to fine-tune the evolutionary process. A GA should strike a balance between the exploitation of good individuals to facilitate the search, and the exploration of search space to prevent a premature convergence to local optima.

In this problem, a vector of location variables $y(i)$ is an individual as a potential solution and the objective function (0) in the formulation is used as an evaluation function. It is important to make a genetic operator in order to transmit good genetic characteristics of parents to their offsprings. Here, the two-point crossover operator, which is usually used in evolutionary algorithms [9, 10], is employed as follows:

- Two crossover points are randomly selected in both parents.
- The elements in the fore and back parts of the two crossover points in one parent of P1 are copied into an offspring in the same position as they appear in P1.
- And then, the elements between the two crossover points in the other parent of P2 are copied into an offspring in the same position as they appear in P2.
- The other offspring can be created by switching the roles of the two parents.

An example is given in Fig. 2, where crossover points are marked by '$|$'. In evolutionary algorithms, a mutation operator acts on a single parent and produces an offspring by introducing small changes in order to ensure the diversity of potential solutions and to prevent a premature convergence to local optima [9, 10]. For this problem, some individuals are randomly chosen with the individual mutation rate of P_m and mutation operation is applied to the individuals with the gene mutation rate of P_g.

Suppose $P(t)$ is the parents in the current generation t. In the evolutionary process, maintaining diverse populations is necessary for the long-term success of any evolutionary algorithm. A genetic search with a diverse population can continue utilizing recombination to produce a new structure, and thus avoid becoming trapped at local optima. To promote diversity and search efficiency, in this paper, the evolutionary system is based on the localized neighborhood interactions within populations. Population forms a two-dimensional toroidal square lattice. Although the size and shape of a neighborhood can of course be defined in many different ways, the structure of 3×3 neighborhood is used here for localized evolution. Therefore, the neighborhood area of $N(t)$ at the current generation t should be set when N_{ij} denote the neighborhood including individual (i, j) and its eight neighbors in population such as Fig. 3.

Based on fitness, a roulette wheel method, *ie* , the better individual has the larger survival probability in the next generation, is used for the selection of parents and individuals for replacement [9, 10]. $N'(t)$ is selected for the reproduction process according to the fitness in $N(t)$, and the new offspring, $C(t)$, are created by applying crossover operation for $N'(t)$ according to the crossover rate and by applying mutation operation for $N(t)$ according to mutation rate. When $N'(t)$ or $N(t)$ is replaced with $C(t)$, the elite individual is not replaced during the reproduction process. The procedure of the proposed algorithm is shown in Fig. 4.

4 SIMULATION DESIGN AND RESULTS

To verify the performance of the proposed GA, computational simulations were carried out on various benchmarking problems. In constraint (1), the expected demands of program k in CO j, $D_k(j)$, should be com-

Table 1. Problem parameter setting

Parameters	Values
Storage cost per program (C_s)	1
Transmission cost per program (C_{ij})	Randomly chosen from 2,3 and 4
Installation cost per server (C_F)	1000, 2500, 4000
Number of multiple accesses (S)	1
Number of programs for service (K)	200
Mean service time	1.0
Busy hour request attempts per user	0.1
Service blocking probability	0.005

puted. For this purpose, we have used the *Erlang B* model. In the model, traffic intensity can be calculated by the multiplication of MOD subscribers, busy hour request attempts per users, program preference probability, and mean service time, divided by service time unit. Blocking probability to a program stored in a server is also needed in order to determine the demands.

Table 2. Bench-marking problems

x-COs1)		
Problems	Number of multiple accesses	Difference in #of subscribers
Nx1)-11	1	No2)
Nx-12	10	No
Nx-13	20	No
Nx-14	30	No
Nx-21	1	Medium3)
Nx-22	10	Medium
Nx-23	20	Medium
Nx-24	30	Medium
Nx-31	1	Large4)
Nx-32	10	Large
Nx-33	20	Large
Nx-34	30	Large

1) x: the number of COs, which has 5, 7, 10, 20, 30 and 50 respectively

2) No: the number of subscribers per CO is 10,000

3) Medium: 10,000 and 2,500 are alternatively assigned as the number of subscribers in each CO

4) Large: 20,000, 15,000, 10,000, 5,000 and 2,500 are alternatively assigned as the number of subscribers in each CO

The preference probability for the program with preference order k is denoted by p_k and calculated by (5) proposed by [2]. The parameter D_{HP} is the ratio between the preference probability $(k-1)$-th and k-th. D_{HP} is greater than one because all the programs are sorted in a decreasing order with respect to their preference probability. In this paper, D_{HP} is set to 1.3 on an average even though it differs according to the number of COs.

$$p_k = \frac{p_{k-1}}{D_{HP}}\,, \quad k > 1\,, \quad p_1 = \frac{1-(1/D_{HP})}{1-(1/D_{HP})^k}\,. \qquad (5)$$

The problem parameters for MOD networks are first set, as shown in Tab. 1. Bench-marking problems are made according to the number of COs, installation cost per server and difference in the number of subscribers such as Tab. 2.

Table 3. Parameters for GA

Parameters	Values
Crossover rate	0.7
Mutation rate (P_m, P_g)	0.1, 0.1
Population Size Neighborhood Size	100 9
Termination criteria for algorithm (# of generations according to # of COs)	70 / 120 / 180 / 400 / 700 / 1500

The GA was coded in C++ and each experiment was repeated 10 times for every bench-marking problem. Preliminary experiments were performed to determine a proper parameter value, different values are taken with respect to the bench-marking problems such as Tab. 3.

The enumeration method, which is used as a reference method, is that the best solution is selected after evaluating all possible solutions. The proposed GA is compared with it in terms of solution quality and computation time. Relatively small-sized problems were implemented, since enumeration method failed to find a solution to large-sized problems within reasonable computer time.

The representative results are shown in Tab. 4. The first column identifies the bench-marking problems. The next four columns show the total costs, which are the values of dividing every resulting objective value by 100. The sixth column indicates the standard deviation (SD) of solutions obtained by using GA. The computation time for enumeration method and GA appears in the seventh and the eighth column, respectively. A comparison in solution quality between GA and enumeration method is shown in the last column. Efficiency is calculated by $\{1-$(average of the GA $-$ optimal solution) / optimal solution$\} \times 100$. The computation time and efficiency are graphed in Figs. 5 and 6, respectively, when the network consists of 10 and 20 COs. The results show that the proposed GA finds out an optimal solution in almost all the trials for small-sized problems.

In order to investigate the performance of the GA for relatively large-sized problems, simulations were also performed. The result is shown in Tab. 5. In the sixth column, difference is calculated by $\{$(the worst solution $-$ the best solution) / the best solution$\} \times 100$. In results, the standard deviation of the solutions is relatively small,

Table 4. Performance comparison between Enumeration and GA

Problems	E	Proposed GA				Computational time (sec.)		Efficiency
		Best	Worst	Average	SD	E	GA	(%)
N5-1	133.9	133.9	133.9	133.9	0.0	0.0	0.1	100.0
N5-2	208.9	208.9	208.9	208.9	0.0	0.0	0.1	100.0
N5-3	267.6	267.6	267.6	267.6	0.0	0.0	0.1	100.0
N5-4	115.1	115.1	115.1	115.1	0.0	0.0	0.1	100.0
N5-5	178.2	178.2	178.2	178.2	0.0	0.0	0.1	100.0
N5-6	218.6	218.6	218.6	218.6	0.0	0.0	0.1	100.0
N5-7	133.7	133.7	133.7	133.7	0.0	0.0	0.1	100.0
N5-8	198.3	198.3	198.3	198.3	0.0	0.0	0.1	100.0
N5-9	243.3	243.3	243.3	243.3	0.0	0.0	0.1	100.0
N7-1	182.3	182.3	182.3	182.3	0.0	0.1	0.4	100.0
N7-2	287.3	287.3	287.3	287.3	0.0	0.1	0.4	100.0
N7-3	350.6	350.6	350.6	350.6	0.0	0.1	0.4	100.0
N7-4	154.5	154.5	154.5	154.5	0.0	0.1	0.4	100.0
N7-5	225.0	225.0	225.0	225.0	0.0	0.1	0.4	100.0
N7-6	282.7	282.7	282.7	282.7	0.0	0.1	0.4	100.0
N7-7	201.2	201.2	201.2	201.2	0.0	0.1	0.4	100.0
N7-8	289.8	289.8	289.8	289.8	0.0	0.1	0.4	100.0
N7-9	356.1	356.1	356.1	356.1	0.0	0.1	0.4	100.0
N10-1	273.0	273.0	273.0	273.0	0.0	0.9	0.7	100.0
N10-2	423.0	423.0	423.0	423.0	0.0	1.1	0.7	100.0
N10-3	526.2	526.2	530.6	526.6	1.4	1.0	0.7	99.9
N10-4	223.2	223.2	226.1	223.5	0.9	1.0	0.7	99.9
N10-5	326.9	326.9	326.9	326.9	0.0	1.0	0.7	100.0
N10-6	387.1	387.1	387.1	387.1	0.0	1.0	0.7	100.0
N10-7	273.2	273.2	273.2	273.2	0.0	0.9	0.7	100.0
N10-8	404.0	404.0	404.0	404.0	0.0	1.0	0.7	100.0
N10-9	490.7	490.7	494.6	491.1	1.2	0.9	0.7	99.9
N20-1	552.1	552.1	552.1	552.1	0.0	1739.1	7.3	100.0
N20-2	852.1	852.1	852.1	852.1	0.0	1739.4	7.3	100.0
N20-3	1048.1	1048.1	1053.9	1049.7	2.0	1739.1	7.7	99.9
N20-4	454.9	454.9	461.4	456.9	2.4	1739.1	7.4	99.6
N20-5	655.8	655.8	661.8	656.4	1.9	1739.1	7.6	99.9
N20-6	770.7	770.7	770.7	770.7	0.0	1739.2	7.8	100.0
N20-7	354.7	354.7	354.7	354.7	0.0	1739.1	7.4	100.0
N20-8	490.5	490.5	494.6	492.8	1.4	1739.4	7.5	99.5
N20-9	537.7	537.7	545.2	538.9	2.6	1739.1	7.6	99.8

and the difference between the worst and the best case is less than 1 % in addition to computation time does not increase exponentially as the number of COs increases. One can conclude that, for large-sized problems, the proposed algorithm is able to achieve a high quality solution with a modest growth in computational effort.

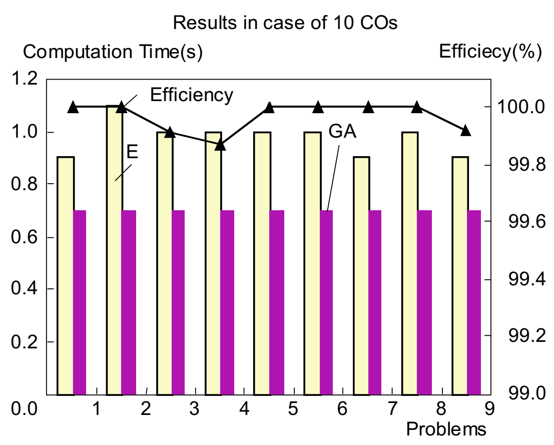
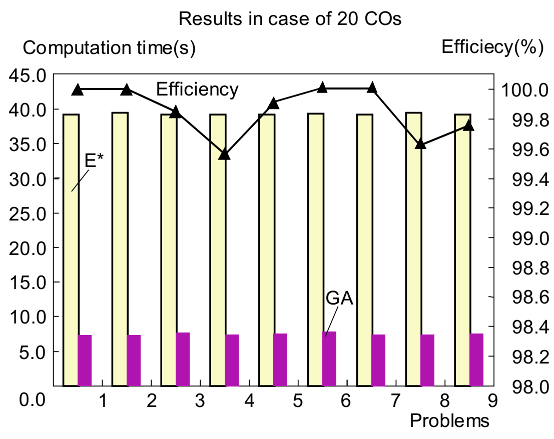


Fig. 5. The results in case of 10 COs



*: values by subtracting 1700 from every resulting computation time value

Fig. 6. The results in case of 20 COs

5 CONCLUSIONS

In this paper, a genetic algorithm application to non-hierarchical and decentralized MOD network design was proposed. Both server location and storage allocation were considered under network resource optimization, based on cost analysis including server including installation cost for servers, program storage cost, and transmission cost. The comparisons made for various problem modifications were used for verifying the efficiency of the proposed methods. High quality solutions were received within reasonable time periods of computations. The simulations produced have shown improved the quality of solution as well as the efficiency of computation.

Although the GA to design MOD network is developed, it can be used to reallocate resource when network is affected by factors such as the release of new programs, increase or decrease of subscribers and the changes of involved network costs. Therefore, the amount of program stored in server can be adjusted periodically without changing the location of server, or the location of newly installed server can be decided in any case.

Additionally, the attractive feature of the proposed algorithm is its reasonable flexibility. It can be applied to solve many variants of optimization criteria and restrictions. as well as multiple location-allocation types of problems.

Table 5. Performance analysis of the proposed method

Problems	Proposed GA					Computation
	Best	Worst	Average	SD	Difference(%)	time (sec.)
N30-1	817.8	817.8	817.8	0.0	0.0	9.1
N30-2	1267.8	1272.9	1268.3	1.6	0.4	9.1
N30-3	1528.3	1530.8	1528.6	0.8	0.2	9.6
N30-4	673.1	679.1	674.6	2.1	0.9	9.2
N30-5	969.5	969.5	969.5	0.0	0.0	9.4
N30-6	1119.1	1124.7	1120.0	1.7	0.5	9.7
N30-7	822.8	825.8	823.4	1.2	0.4	9.1
N30-8	1196.1	1196.1	1196.1	0.0	0.0	9.3
N30-9	1442.9	1442.9	1442.9	0.0	0.0	9.5
N50-1	1384.6	1384.6	1384.6	0.0	0.0	35.4
N50-2	2134.6	2140.3	2135.7	2.4	0.3	35.3
N50-3	2601.1	2606.9	2602.5	1.8	0.2	38.2
N50-4	1136.5	1142.8	1137.7	2.1	0.6	35.6
N50-5	1648.5	1651.6	1649.0	1.0	0.2	36.9
N50-6	1889.2	1894.8	1892.2	1.9	0.3	38.3
N50-7	1390.8	1395.4	1392.5	2.2	0.3	35.5
N50-8	2024.7	2026.2	2025.0	0.6	0.1	36.3
N50-9	2435.0	2438.7	2435.7	1.4	0.2	37.3

References

[1] GELMAN, A. D.—HALFIN, S.: Analysis of Resource Sharing in Information Providing Services, IEEE GLOBECOM'90, 312–316, 1990.

[2] De GIOVANNI, L.—LANGELLOTTI, A. M.—PATITUCCI, L. M.—PETRINI, L.: Dimensioning of Hierarchical Storage for Video on Demand Service, IEEE ICC'94, 1739–1743, 1994.

[3] HWANG, R. H.—SUN, Y. C: Optimal Video Placement for Hierarchical Video-on-Demand Systems, IEEE Transaction on Broadcasting **44**(4) (1998), 392–401.

[4] OUVEYSI, I.—WONG, K. C.—CHAN, S.—KO, K. T.: Video Placement and Dynamic Routing Algorithms for Video-On-Demand Networks, The Bridge to Global Integration, IEEE, 2, 658–663, 1998.

[5] PETIT, G. H.—DELODDERE, D. VERBIEST, W.: Bandwidth Resource Optimization in Video-On-Demand Network Architectures, IEEE GLOBECOM'94, 91–97, 1994.

[6] WONG, E. W. M.—CHAN, S.: Modeling of Video-on-Demand Networks with Server Selection, The Bridge to Global Integration. IEEE, 1, 54–59, 1998.

[7] SCHAFFA, F.—NUSSBAUMER, J. P.: On Bandwidth and Storage Tradeoffs in Multimedia Distribution Networks, IEEE INFORM'95, 1020–1026, 1995.

[8] KIM, Y. K.—KIM, J. Y.—KANG, S. S.: A Tabu Search Approach for Designing a Non-Hierarchical Video-on-Demand Network Architecture, Computers & Industrial Engineering **33** No. 3-4 (1997), 837–840.

[9] GOLDBERG, D. E.: Genetic Algorithm in Search Optimization & Machine Learning, Addison-Wesley, Readings, 1989.

[10] MICHALEWICZ, Z.: Genetic Algorithm + Data Structures = Evolution Programs (2nd edn), Springer-Verlag, Berlin, 1994.

Received 22 August 2003

Sun-Jin Kim has received the BS and MS degrees in Industrial Engineering from Chonnam National University, Korea, in 1996 and 1998, respectively. She is currently working toward the PhD degree in IT Business department at ICU (Information and Communications University), Korea. Her research interests include resource management, combinatorial optimization, telecommunication network analysis in communication network and evolutionary algorithms.

Mun-Kee Choi has received BS degree in Applied Mathematics from Seoul National University, Korea, MS degree in Industrial Engineering from Korea Advanced Institute of Science and Technology and PhD degree in Operations Research from North Carolina State University, USA at 1974,1978,1989 respectively. From 1978 to 1999, he worked network as a senior research staff in Electronics and Telecommunications Research Institute (ETRI). He is currently a professor of IT Business department at ICU (Information and Communications University), Korea. His current research interests are telecommunication system and networking technology, Network services and business model, and performance analysis of related issues.