A Sexual Elitist Genetic Algorithm for Providing QoS in Distributed Virtual Environment Systems *

S. Rueda, P. Morillo, J. M. Orduña
Departamento de Informática
Universidad de Valencia
E-mail: Juan.Orduna@uv.es

J. Duato
DISCA
Universidad Politécnica de Valencia
E-mail: jduato@gap.upv.es

Abstract

Architectures based on networked servers have become a de-facto standard for Distributed Virtual Environment (DVE) systems. These systems allow a large number of remote users to share a single 3D virtual scene. In order to provide quality of service in a DVE system, clients should be assigned to servers taking into account system throughput and system latency. This highly complex problem is known as the quality of service (QoS) problem.

This paper proposes an elitist sexual genetic algorithm for solving the QoS problem in Distributed Virtual Environment systems. Performance evaluation results show that, due to its ability of both finding good search paths and keeping diversity escaping from local minima, this nature inspired technique can provide significantly better solutions than other heuristic methods like GRASP or SA with shorter execution times. Therefore, the proposed implementation of GA search method can improve the QoS offered by DVE systems.

1. Introduction

Distributed Virtual Environment (DVE) systems have experienced a spectacular growth over the last few years. These systems allow multiple users, working on different client computers that are interconnected through different networks (and even through the Internet) to interact in a shared virtual world. This is achieved by rendering images of the environment as they would be perceived by the user if he was located at that point in the virtual environment. Each user is represented in the shared virtual environment by an entity called *avatar*, whose state is controlled by the user. Since DVE systems support visual interactions between multiple avatars, every change in each avatar must be propagated to other avatars in the shared virtual environment. DVE systems are currently used in many different applications [20], such as collaborative design [4], civil and military distributed training [13], e-learning [19] or multiplayer games [9].

Architectures based on networked servers are becoming a de-facto standard for DVE systems [20, 11, 21]. In these architectures, the control of the simulation relies on several interconnected servers. Client computers are attached exclusively to one of the servers in the system. When a user modifies an avatar, the client computer controlling this avatar sends an updating message to the client computers controlling other avatars, in order to achieve that all avatars have a consistent and actualized view of the virtual world. When the number of connected clients increases, the number of updating messages must be limited in order to avoid a message outburst. In this sense, concepts like areas of influence (AOI) [20], locales [1] or auras [5] have been proposed for limiting the number of neighboring avatars that a given avatar must communicate with. All these concepts define a neighborhood area for avatars, in such a way that a given client computer controlling a given avatar i must notify all the movements of i (by sending an updating message) only to the client computers that control the avatars located in the neighborhood of avatar i. The avatars in the AOI of avatar i are denoted as *neighbor avatars* of avatar i. Figure 1 shows an example of a DVE system whose architecture is based on networked servers. In this example, the AOI of each avatar is represented as a circumference.

The *partitioning problem* [10] has been shown as the main issue in the design of efficient DVE systems based on networked servers. It consists of efficiently distributing the workload among the different servers in the system (by assigning avatars to servers). In a previous paper, we proposed a nature inspired technique for solving the partitioning problem [15]. Also, we have developed a partitioning method that provides a significant throughput increase to DVE systems [14, 16, 18]. Nevertheless, the most important performance measures in DVE systems (as in any

*Supported by the spanish MCYT under grant TIC2003-08154-C06-04

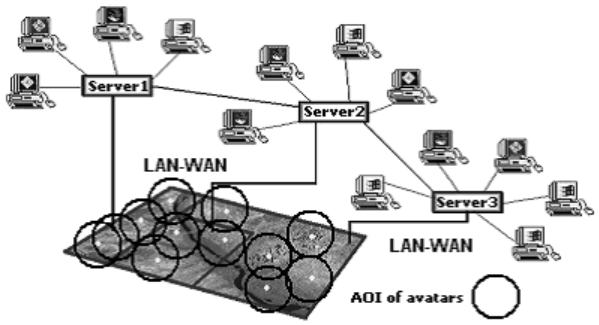

Figure 1. Examples of AOI in a DVE system whose architecture is based on networked servers.

client-server system) are not only throughput but also latency. Latency can be defined as the time interval from the instant when any neighbor of a given avatar i makes a movement until the instant when avatar i is notified of that movement. Latency represents *Quality of Service (QoS)* provided to users by the system, since it determines how fast changes in the virtual world are notified to the proper client computers.

Once the partitioning method has ensured that the system throughput is maximized (it has provided a partition where the estimated percentage of CPU utilization in all the DVE servers is under 99% [14]), then the computing resources can still be used to decrease the average system time response provided to avatars. This improvement should be carried out also by the partitioning method, since it is really a trade-off between system throughput and system latency. The problem of solving the partitioning problem ensuring both that the system is below its saturation point and that time the average latency provided to avatars is minimized is known as the QoS problem. This problem can be modeled as finding a partition that minimizes a quality function. In a previous paper, we performed a comparison study of some heuristic methods applied to the QoS problem in DVE systems [17]. In this study, we implemented and tested Simulated Annealing and GRASP techniques. These heuristic methods were capable of improving the QoS offered by DVE systems.

In this paper, we present the implementation, tuning, and evaluation of a sexual elitist variation of Genetic Algorithms (SEGA) for solving the QoS in Distributed Virtual Environment Systems. Performance evaluation results show that this nature inspired technique is able to improve the QoS provided by DVE systems with shorter execution times than GRASP or SA techniques. Therefore, this implementation of GA technique can be considered as the best heuristic method for providing QoS in DVE systems.

The rest of the paper is organized as follows: Section 2 details the problems to be solved for achieving QoS in DVE systems . Section 3 describes the implementation and tuning of SEGA. Next, Section 4 presents the performance evaluation of the proposed method. Finally, Section 5 presents some concluding remarks.

2. Background

The term *Quality of Service (QoS)* has been extensively used in wide area network (WAN) environments to describe the ability of some systems to conform with some specific user requirements of latency, jitter delays, traffic peaks, etc. [22, 2]. However, the term QoS can be applied to any system, and it means that the system not only supports a given client but it also fulfills some specific requirements of that client.

The QoS problem has been already described in DVE systems, and some strategies have been proposed for solving it [3, 7]. Approaches like [7] use latency compensating methods in order to repair the effects of high network jitter. Adaptive rendering strategies like [3] or [20] modify the resolution of the 3-D models depending on the client connection speed. However, none of these strategies takes into account the non-linear behavior of DVE systems with the workload assigned to each server, as described in [14]. Therefore, these strategies cannot guarantee that the performance provided to avatars will not degrade beyond any threshold value.

The QoS problem can be expressed in DVE systems as latency constraints. A DVE system can only offer QoS to clients if it is working below its saturation point and at the same time the average round-trip delay for the messages sent by each avatar (denoted as ASR, for *average system response*) is lower than 250 ms. [7]. However, the ASR provided to a given avatar i depends on which servers are the neighbor avatars i assigned to. If avatar i is assigned to server s, then the ASR for avatar i linearly decreases with the number of neighbor avatars of i that are assigned server s . A partitioning method that provides QoS to avatars should maximize the number of neighbor avatars assigned to the same server. At the same time, it should balance the workload (avatars) in order to keep the system away from saturation. Additionally, it must not migrate more than 30% of avatars in the system [8]. Therefore, the QoS problem consists of finding the best partition complying with all these three requirements. In a previous paper we defined a quality function that measures how a given partition fulfills these requirements [17]. In order to make this paper self-contained, in this section we will briefly describe this function.

Equation 1 represents the evaluation function proposed

for providing QoS, composed of three terms. This function measures the quality of each possible partition (assignment of n avatars to s servers). The partitioning method consists of performing a heuristic search to find the partition with the lowest value of F_{QoS} as possible. F_{QoS} function is defined as

$$F_{QoS} = \sum_{i=0}^{s} h_{cpu}(i) + \sum_{i=0}^{n} h_{asr}(i) + n_m \qquad (1)$$

This function is the sum of three different functions or terms. The term $h_{cpu}(i)$ is a function of the estimated percentage of CPU utilization in each server i. Figure 2 shows the behavior of this function, that is exponential. This term will make F_{QoS} to assign a poor (high) value to any partition where at least one of the DVE servers is estimated to be saturated or close to saturation.

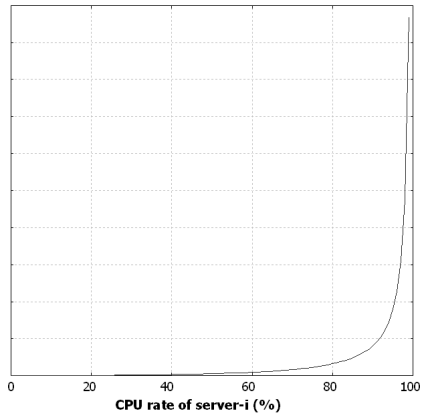

Figure 2. Behavior of $h_{cpu}(i)$

The term $h_{asr}(i)$ measures the estimated ASR provided to each avatar in the system by a given partition. Figure 3 shows the behavior of this term, that is composed of two sections. Function $h_{asr}(i)$ penalizes partitions where the estimated ASR of avatars is higher than 250 ms.

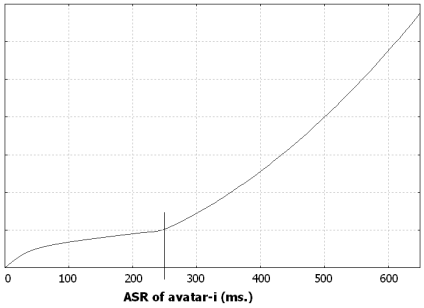

Figure 3. Behavior of $h_{asr}(i)$

Finally, the term n_m measures the number of avatars that should be migrated in order to obtain a given partition. This function is also composed of two sections, as shown in Figure 4. This function penalizes partitions that migrates more than 30% of avatars.

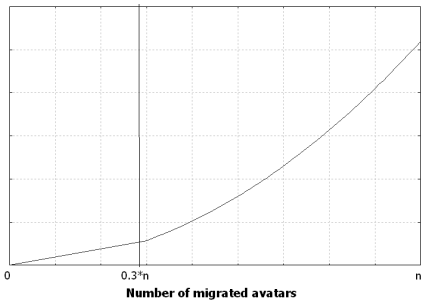

Figure 4. Behavior of n_m

Thus, the QoS problem in DVE system is reduced to find a partition (assignment of the existing n avatars to the s existing servers in the DVE system) with the lowest value of F_{QoS} as possible. Because of the high complexity of this problem, labeled as NP-hard in other systems [22], different approaches based on metaheuristic procedures were proposed [17].

3. A Sexual Elitist Genetic Algorithm

In this section we describe the particular implementation of Genetic Algorithms that we have used to solve the QoS problem in DVE systems, denoted as *Sexual Elitist Genetic Algorithm (SEGA)*. Genetic Algorithms (GA) consists of a search method based on the concept of evolution by natural selection [6]. GA starts from an initial population, made of P *chromosomes*, that evolves following certain rules, until reaching a convergence condition that maximizes a fitness function. Each iteration of the algorithm consists of generating a new population from the existing one by recombining or even mutating chromosomes. A chromosome can contain a *genotype* or particular solution of the problem and also a *phenotype* or additional information for tuning the behavior of the algorithm.

In SEGA implementation, the genotype consists of a list with the pairs avatar-server. If there are N avatars in the system, this list contains N pairs, each one designating the server where that avatar is assigned to. The phenotype consists of information about the estimated workload that each avatar adds (in terms of the CPU utilization) to the server where it is assigned to. Also, the phenotype indicates if each avatar is a *boarder avatar* or not. If a given avatar is assigned to server s, then this avatar is a boarder avatar if any of its neighbor avatars (the avatars within its AOI) is

assigned to another server different from s. We use F_{QoS} as the fitness function to be (in our case) minimized.

Most of heuristic methods are based on the random generation of an initial population. However, if the initial population has been correctly defined, then the heuristic method easily obtains a good approximation to the global optimum. In this case the algorithm should maintain a certain level of structural diversity among all the chromosomes, in order to avoid the premature convergence of the search [12]. Therefore, the initial population in SEGA is provided by the ALB partitioning method [16]. This load balancing method provides a balanced partition of the DVE system, ensuring low initial values of the term $\sum_{i=0}^{s} h_{cpu}(i)$. The GA algorithm must keep the workload balanced while providing QoS to the maximum number of avatars as possible.

Each iteration consists of generating a *descendant* generation of chromosomes, starting from an *ancestor* generation. The way that the algorithm provides the next generation determines the behavior of SEGA. We have chosen a sexual reproduction technique [12], in such a way that each descendant is generated starting from two ancestors. However, in order to provide a high diversity, we have also used non homogeneous hybrid derivation (certain rate of asexual reproduction). Additionally, we use *elitism* in each iteration, and so the name of the heuristic method. The term elitism means that some individuals (chromosomes) of a given population are directly passed to the next generation without suffering any variation [12]. In each SEGA iteration, an intermediate population of $P + N_{elitist}$ chromosomes is generated, where the $N_{elitist}$ chromosomes are the ones with the best fitness function in the previous iteration. At the end of the iteration, the new generation will be composed of the P chromosomes with the best fitness function in the intermediate population.

In the case of sexual reproduction, in each iteration the first ancestor for the $i-th$ chromosome of the population is the $i-th$ chromosome of the population in the previous iteration. The second ancestor is randomly selected among the 50% of the previous population with the best fitness function. The $i-th$ chromosome of the population is then obtained by applying one point crossover, multi-point crossover or uniform crossover to the ancestors [12]. In the case of asexual reproduction, the ancestor itself is selected as the descendant.

Once the descendants of iteration t are obtained, if the finishing condition is not reached then a recombination process is performed on all the chromosomes of that descendant. This recombination process consists of randomly selecting two boarder avatars and exchanging the servers they are assigned to. This process helps to keep diversity while exclusively exploring highly probable solutions. Finally, a mutation can be performed on the resulting descendant. It consists of randomly selecting an avatar in a chromosome and changing its server. The whole process performed in each iteration can be expressed as the following pseudocode statements (where Genotype Gt is the resulting population of the previous iteration t, composed of P chromosomes):

```
Iteration t+1 (Genotype Gt)
CONST
   Nelitist /* Num. of elitist chromosomes */
   P        /* Num. of chromosomes in genotype */
   N        /* Num. of avatars in DVE system */
   Sexuality   /* Sexuality rate */

TYPE
   chromosome : int[N]

VAR
   int i
   Anc1, Anc2 : chromosome  /* Ancestors */
   Desc : chromosome        /* Descendant */

begin
   Copy_Nelitist_best_of_Gt_to Gi()
   For i:=Nelitist to P+Nelitist do
      Anc1 := Gt[i]
      a := Reproduction_select(Sexuality)
      if a = 0  then /* 0 = Sexual, 1 = Asexual */
           Anc2 := Random_select_from(Gi)
           crossover := Random_select_crossover()
           case(crossover)
              one-point:
                 Desc := 1point_cross(Anc1,Anc2)
              multipoint:
                 Desc := mpoint_cross(Anc1,Anc2)
              uniform:
                 Desc := unif_cross(Anc1,Anc2)
           end_case
      else
           Desc := Anc1
      end_if
      if (NOT converg_condition(Desc)) then
          recombination(Desc)
      end_if
      if (random()< mutation_rate) then
          mutation(Desc)
      end_if
      Gi[i] := Desc;
   end_for
   Evaluate_And_Sort(Gi);
   For i:=0 to P do
      Gt+1[i] := Gi[i];
   end_for
end
```

In order to establish the convergence condition (finishing condition of the algorithm), we have considered three parameters. The first one is the standard deviation of the fitness function (F_{QoS}). When all the individuals in the population are very similar, it will hardly provide solutions better than the current one. Therefore, we will stop the algorithm if this parameter is below a threshold value. The second parameter is the number of consecutive iterations performed without improving the current best fitness function. If we

cannot find a better chromosome (solution) in a given number of iterations, we assume that the current solution is the best one. Finally, we have also limited the total number of iterations to a given value. When any of these three conditions is is fulfilled, then the algorithm finishes.

The proposed implementation has several parameters to be tuned, in order to obtain the maximum performance of the algorithm. These parameters are the following ones: the number of chromosomes in the population (P), the number of iterations, the mutation rate, number of elitist chromosomes ($Nelitist$), the sexuality rate (the percentage of iterations in which sexual reproduction is used), the minimum standard deviation allowed for the convergence condition, and the maximum number of iterations allowed without improving the fitness function. We have empirically tuned these seven parameters in two different DVE configurations, denoted as MEDIUM1 and MEDIUM2 [17]. MEDIUM1 is composed of 250 avatars and 3 servers, and MEDIUM2 is composed of 700 avatars and 10 servers. However, due to space limitations, we only present here the result for MEDIUM2 configuration. The results obtained for MEDIUM1 configuration were very similar. For tuning purposes we have measured both the system cost provided by SEGA (the values of F_{QoS}) and also the execution times required by the algorithm in each case. Due to space limitations, the figures showing the required execution times are neither presented. Unless explicitely stated, most of them show a linear correspondence with the parameter tuned in each case.

In order to evaluate the performance of a given partitioning method, usually 3 different avatar distributions in the virtual world have been suggested in the literature: uniform, skewed and clustered [10]. The reason for using different distributions is that they generate a different workload. Fig. 5 shows an example of these avatar distributions in a 2-D virtual world. In this figure, the virtual world is a square and avatars are represented as black dots. We have tuned and evaluated SEGA method for these three distributions of avatars in the virtual world.

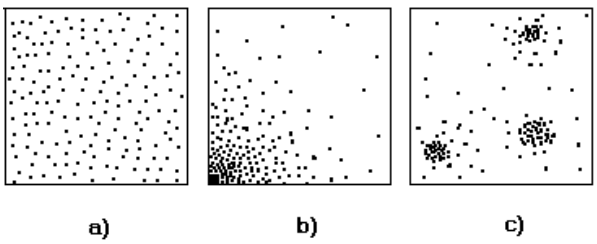

Figure 5. Distributions of avatars in a 2-D virtual world: a) Uniform b) Skewed c) Clustered

Figure 6 shows the values of F_{QoS} reached with SEGA for different population sizes. This figure shows on the X-axis the number of chromosomes P, while on the Y-axis it shows the F_{QoS} values obtained for each one of the distributions considered. It can be seen that the three plots describe a curve with descending slope, until a value of 100 chromosomes. From this value up, F_{QoS} values hardly change as the number of chromosomes increases. Therefore, we have used 100 as the number of chromosomes P.

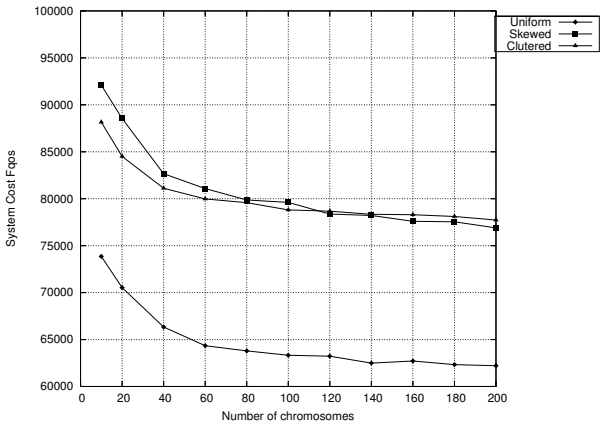

Figure 6. System costs obtained for different population sizes

Figure 7 shows the values of F_{QoS} provided by SEGA when different numbers of iterations are performed. Again, the values of F_{QoS} describe a curve with descending slope in all the plots, until a value of 300 iterations is reached. From that value up, all the plots are almost a flat line. This behavior shows that the algorithm finds another convergence condition before reaching a total threshold limit of 300 iterations. Therefore, we have used this value as the total limit of iterations.

Figure 8 shows the values of F_{QoS} provided by SEGA when different mutation rates are used in each iteration. This figure clearly shows that this parameter does not have any effect on the behavior of the algorithm. We have chosen a value of 0.05 for this parameter.

Figure 9 shows the values of F_{QoS} provided by SEGA when elitism is applied on different numbers of chromosomes (when $Nelitist$ variable varies). This figure clearly shows that the system cost greatly decreases as the elitism rate increases, until a value of 50%. From that value up, the system cost increases and then remains constant. Thus, we have chosen an elitism rate of 50% of the population. This means that 50 chromosomes from a population of 100 chromosomes will be directly copied from Gt to Gi in each iteration.

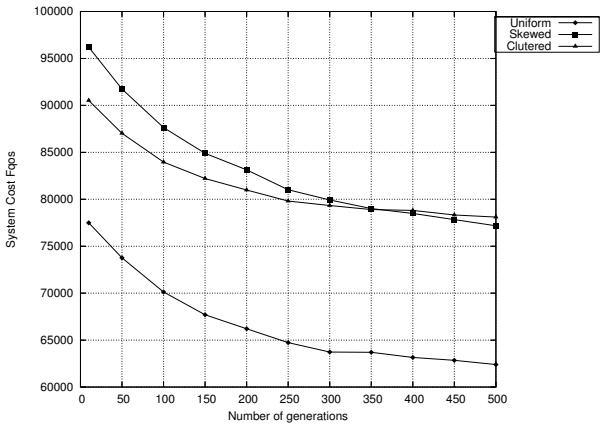

Figure 7. System costs obtained for different number of iterations

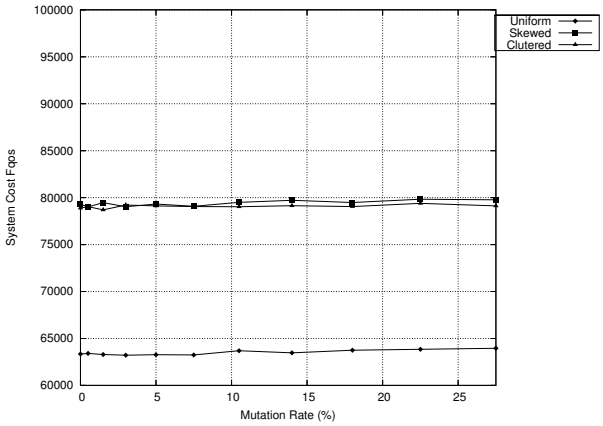

Figure 8. System costs obtained for different mutation rates

Figure 10 shows the values of F_{QoS} provided by SEGA for different sexuality rates (percentages of iterations where sexual reproduction is used). This figure shows the same behavior for the three distributions of avatars. The F_{QoS} values provided by SEGA decreases as the percentage of sexual derivations increases. However, the slope of the three plots progressively decreases as this percentage increases. We have chosen 75% as the best sexuality rate (percentage of sexual reproduction), in order to provide population diversity.

Due to space limitations, the tuning of the other two parameters (the minimum standard deviation allowed for the convergence condition, and the maximum number of iterations allowed without improving the fitness function) are not presented here. We have chosen a value of 0.005 as the minimum standard deviation for the population. A population with a lower value of standard deviation is considered as the final population. In the same way, we have chosen a value of 50 as the maximum number of iterations allowed without improving the fitness function. If the algorithm reaches this number of consecutive iterations without decreasing the current minimum value of F_{QoS}, then the algorithm finishes. Using these values for these seven parameters, we have obtained the best performance of SEGA.

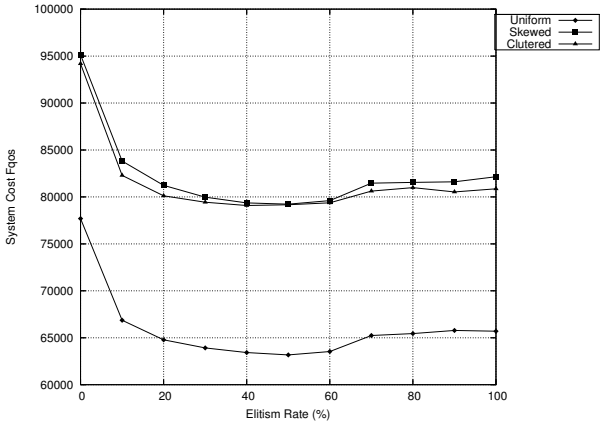

Figure 9. System costs obtained for different elitism rates

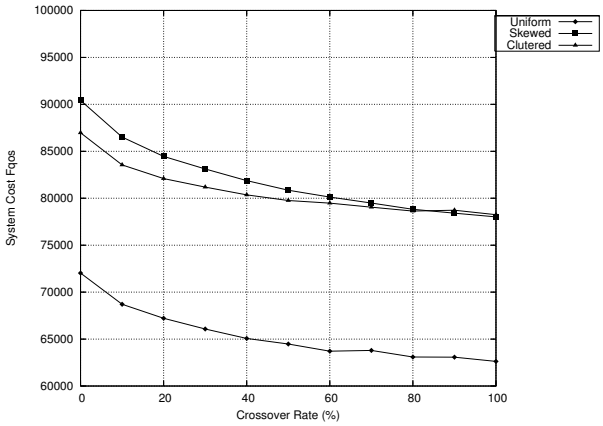

Figure 10. System costs obtained for different sexuality rates

4. Performance Evaluation

In this section, we present the performance evaluation of the heuristic method described in the previous section when it is used for solving the QoS problem in DVE systems. For comparison purposes, we have also tested another two heuristic methods, SA and GRASP, that have been adapted to the QoS problem in DVE systems [17]. We have empirically tuned these other heuristics in the same MEDIUM1 and MEDIUM2 configurations described in the previous section. However, due to space limitations, we present here the result for MEDIUM2 configuration. The results obtained for MEDIUM1 configuration were very similar.

Table 1 shows the results obtained for a MEDIUM2 configuration when avatars are uniformly located in the virtual world. Each column of the table shows the results for a given partitioning algorithm. For comparison purposes, the first column shows the results for the ALB algorithm [16]. This algorithm is not oriented to provide QoS, and it only provides a balanced initial partition. The rest of the columns show the results for SA, GRASP and SEGA methods. Each row in the table shows a different performance measure. The first row shows the maximum estimated percentage of CPU utilization that any server will have with the resulting partition provided by each method. This measure must not be greater than 99%. The second row shows the system cost (in terms of the quality function F_{QoS} associated to the resulting partition) provided by each method. The third row shows the estimated number of avatars that will be provided with QoS, according to the resulting partition provided by each method[17]. The fourth row shows the number of avatars that have to be migrated in order to arrive to the resulting partition from the current (initial) partition. Finally, last row shows the execution time (in seconds) required by each method in order to provide the resulting partition.

	ALB	SA	GRASP	SEGA
Max Ut. (%)	17	17	19	16
F_{QoS}	73851	69633	62048	61558
QoS	496	526	619	621
$\Gamma(P_0)$		54	101	101
T_{exe} *(s.)*		7,68	7,31	9,07

Table 1. Results for a MEDIUM2 DVE system with a uniform distribution of avatars

Table 1 shows that all the considered methods manage to keep the system below the saturation point, since the maximum estimated utilization percentage does not reach 20%. These values also indicate that a uniform distribution of avatars generate a low load. The $\Gamma(P_0)$ row shows that the three methods provide good partitioning efficiency, since all of them provide a final partition where the number of avatars to be migrated is below the threshold of one third of the population (in the case of MEDIUM2, 700/3). This table also shows that for a uniform distribution of avatars, SEGA method does not provide a significant performance improvement in regard to GRASP or SA heuristic methods. Although it provides a final partition with a slightly lower value of F_{QoS}, and it manages to provide QoS to two avatars more than GRASP method does, the required execution time is longer than the one required by GRASP or SA methods.

Table 2 shows the results for the MEDIUM2 configuration when avatars are located in the virtual world following a skewed distribution. In this case, the maximum utilization rates increase in regard to the ones in Table 1, showing that in this distribution the workload generated with the same number of avatars is greater. Table 2 clearly shows how for this distributions of avatars SEGA is able to provide a partition with higher values of F_{QoS} than the ones provided by GRASP or SA, increasing in a 24% the number of avatars provided with QoS when compared with GRASP method, and increasing it in a 48% when compared to SA method. Additionally, it requires the shortest execution time in order to provide the resulting partition.

	ALB	SA	GRASP	SEGA
Max Ut. (%)	58	53	58	54
F_{QoS}	99408	88426	82857	75326
QoS	211	290	347	431
$\Gamma(P_0)$		194	81	169
T_{exe} *(s.)*		84,08	20,07	18,43

Table 2. Results for a MEDIUM2 DVE system with a skewed distribution of avatars

Finally, Table 3 shows the results for the MEDIUM2 configuration when avatars are located in the virtual world following a skewed distribution. For this distributions of avatars, the maximum utilization rates still increases in regard to Table 2. This means that this is the distribution where avatars generate the greatest workload. In this case, SEGA is capable of providing a partition with a F_{QoS} that is 5% lower than the one provided by GRASP and 8.5% lower than the one provided by SA method. As a result, the number of avatars provided with QoS increases to a value of 436, while for the other two methods it is 415 and 349, respectively. Nevertheless, the most spectacular improvement is done in terms of execution times. The execution time required by SEGA method is around half the time required to execute GRASP method.

All these results shows that the performance of SEGA method increases as the distribution of avatars in the virtual

	ALB	SA	GRASP	SEGA
Max Ut. (%)	79	69	72	70
F_{QoS}	99690	92677	88960	84774
QoS	314	349	415	436
$\Gamma(P_0)$		95	82	120
T_{exe} *(s.)*		48,89	43,66	23,40

Table 3. Results for a MEDIUM2 DVE system with a clustered distribution of avatars

world generates a greater workload.

5. Conclusions

In this paper, we have proposed an elitist sexual genetic algorithm for solving the QoS problem in Distributed Virtual Environment systems. We have compared the performance of the proposed method with the performance obtained with other heuristic methods applied to the same problem.

The results show that although the proposed method does not improve the performance of the DVE system when the workload is low, it properly scales with the workload generated by avatars, therefore providing the best performance for the QoS problem as the DVE system gets more loaded. Since the purpose of the method is to provide QoS to avatars as longer as possible (regardless of the distributions of avatars in the virtual world), these results validate SEGA method as the best heuristic method for providing QoS in DVE systems.

References

[1] D. Anderson, J. Barrus, and J. Howard. Building multi-user interactive multimedia environments at merl. *IEEE Multimedia*, 2(4), 1995.

[2] B. Caminero, C. Carrión, F. Quiles, J. Duato, and S. Yalamanchili. A hardware approach to qos support in cluster environments: The multimedia router mmr. In *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'03)*, 2003.

[3] Z. Choukair, D. Retailleau, and M. Hellstrom. Environment for performing collaborative distributed virtual environments with qos. In *Proceedings of the International Conference on Parallel and Distributed Systems (ICPADS'00)*, pages 111–118. IEEE Computer Society, 2000.

[4] J. S. Dias, R. Galli, and A. C. A. et al. mworld: A multiuser 3d virtual environment. *IEEE Computer Graphics*, 17(2), 1997.

[5] F. C. Greenhlagh. Awareness-based communication management in massive systems. *Distributed Systems Engineering*, 5, 1998.

[6] R. L. Haupt and S. E. Haupt. *Practical Genetic Algorithms*. Ed. Willey, 1997.

[7] T. Henderson and S. Bhatti. Networked games: a qos-sensitive application for qos-insensitive users? In *Proceedings of the ACM SIGCOMM 2003*, pages 141–147. ACM Press / ACM SIGCOMM, 2003.

[8] K. Lee and D. Lee. A scalable dynamic load distribution scheme for multi-server distributed virtual environment systems with highly-skewed user distribution. In *Proceedings of the 10th ACM Symposium on Virtual Reality Software and Technology (VRST 2003)*, pages 160–168. ACM, 2003.

[9] M. Lewis and J. Jacboson. Game engines in scientific research. *Communications of the ACM*, 45(1), 2002.

[10] J. C. Lui and M. Chan. An efficient partitioning algorithm for distributed virtual environment systems. *IEEE Trans. Parallel and Distributed Systems*, 13, 2002.

[11] J. C. Lui, M. Chan, and K. Oldfield. Dynamic partitioning for a distributed virtual environment. Technical report, Department of Computer Science. Chinese University of Hong Kong, 1998.

[12] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer, 1994.

[13] D. Miller and J. Thorpe. Simnet: The advent of simulator networking. *IEEE TPDS*, 13, 2002.

[14] P. Morillo, J. M. Orduna, M. Fernández, and J. Duato. On the characterization of distributed virtual environment systems. In *Euro-Par' 2003 - Lecture Notes in Computer Science 2790*, pages 1190–1198. ACM, Springer-Verlag, 2003.

[15] P. Morillo, J. M. Orduña, and M. Fernández. An acs-based partitioning method for distributed virtual environment systems. In *Proc. of 2003 Int. Parallel and Distributed Processing Symposium Workshops (IPDPS' 2003)*. IEEE Computer Society, 2003.

[16] P. Morillo, J. M. Orduña, M. Fernández, and J. Duato. An adaptive load balancing technique for distributed virtual environment systems. In *Proc. of Intl. Conf. on Parallel and Distributed Computing and Systems (PDCS'03)*, pages 256–261. IASTED, ACTA Press, 2003.

[17] P. Morillo, J. M. Orduña, M. Fernández, and J. Duato. A comparison study of metaheuristic techniques for providing qos to avatars in dve systems. In *ICCSA' 2004 - Lecture Notes in Computer Science 3044*, pages 661–670. Springer-Verlag, 2004.

[18] P. Morillo, J. M. Orduña, M. Fernández, and J. Duato. A fine-grain method for solving the partitioning problem in distributed virtual environment systems. In *Proc. of 16th. Intl. Conf. on Parallel and Distributed Computing and Systems (PDCS'04)*. IASTED, ACTA Press, 2004.

[19] T. Nitta, K. Fujita, and S. Cono. An application of distributed virtual environment to foreign language. *IEEE Education Society*, 2000.

[20] S. Singhal and M. Zyda. *Networked Virtual Environments*. ACM Press, 1999.

[21] P. Tam. Communication cost optimization and analysis in distributed virtual environment. Technical report, Department of Computer Science. Chinese University of Hong Kong, 1998.

[22] X. Yuan. Heuristic algorithms for multi-constrained quality of service routing. *IEEE Transactions on Networking*, 10(2):244–256, 2002.