A Genetic Approach for Adding QoS to Distributed Virtual Environments

S. Rueda [a], P. Morillo [a], J. M. Orduña [a,*], J. Duato [b]
[a] *Departamento de Informática. Universidad de Valencia. SPAIN*
[b] *DISCA. Universidad Politécnica de Valencia. SPAIN*

Abstract

Distributed Virtual Environment (DVE) systems have been designed last years as a set of distributed servers. 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 properly assigned to servers taking into account system throughput and system latency. The latter one is composed of both network and computational delays. This highly complex problem is known as the quality of service (QoS) problem. In this paper, we study the implementation of a genetic algorithm (GA) for solving the QoS problem in DVE systems. Performance evaluation results show that, due to its ability of both finding good search paths and keeping diversity, this nature inspired technique can provide significantly better solutions than other heuristic methods while requiring shorter execution times. Therefore, the proposed implementation of GA search method can actually improve the QoS offered by DVE systems.

Key words: Genetic algorithms, Distributed virtual environments, Quality of service

1. Introduction

The advent of affordable, high-bandwidth Internet connections, together with the widespread use of high performance graphic cards, have allowed a dramatic growth of Distributed Virtual Environment (DVE) systems during last years. DVE 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 world. 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 [1], such as collaborative design [2], civil and military distributed training [3], e-learning [4] or multi-player games [5].

Architectures based on networked servers are becoming a de-facto standard for DVE systems [1,6]. 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 provide a consistent and updated view of the virtual world to all avatars. When the number of connected clients increases, the number of updating messages must be limi-

* Corresponding author. Tel +34 96 3160424
Email addresses: Silvia.Rueda@uv.es (S. Rueda), Pedro.Morillo@uv.es (P. Morillo), Juan.Orduna@uv.es (J. M. Orduña), jduato@gap.upv.es (J. Duato).

ted in order to avoid a communication outburst. In this sense, concepts like areas of influence (AOI) [1], locales [7] or auras [8] 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. Other approaches use three tiered architectures [9,10], data filtering [11] or distributed cache management [12] in order to minimize the impact of network traffic on the performance of the DVE system.

Depending on their origin and destination avatars (client computers), messages in a DVE system can be intra-server or inter-server messages. In order to design a scalable DVE system, the number of inter-server messages should be minimized. Since intra-server messages only require a single server, the more intra-server messages there are, the less resources the application will require to exchange the same number of messages. Fig. 1 shows an example of a DVE system with a multi-server architecture, composed of three servers. It also shows an example of both intra-server and inter-server communication. In this example, avatars are represented as dots and the AOI of each avatar is represented as a circumference.

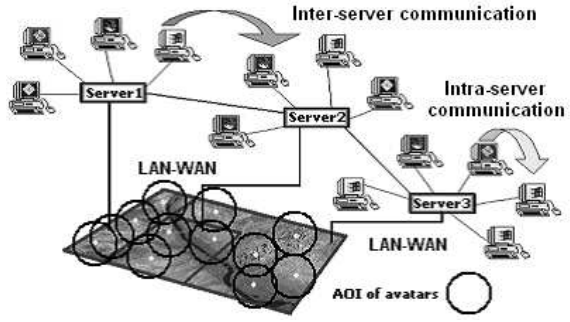

Fig. 1. An example of multi-server architecture for a DVE system.

The *partitioning problem* [6] 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 previous works [13–15], we developed a partitioning method that provides a significant throughput increase to DVE systems. Nevertheless, the most important performance measures in DVE systems (as in any distributed, client-server system) are not only throughput but also latency. Latency can be defined in DVE systems 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% [13]), 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 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 [16]. In this study, we implemented and tested different heuristic search methods based on Simulated Annealing (SA) and Greedy Randomized Adaptive Search (GRASP) techniques. These heuristic methods were capable of improving the QoS offered by DVE systems.

Since the QoS problem in DVE systems can be stated as a problem of balancing both communications and computational load, in this paper we show how a genetic approach can improve the currently existing solutions to this problem. Concretely, we present in this paper the implementation, tuning, and evaluation of a Genetic Algorithm for solving the QoS in Distributed Virtual Environment Systems (QSGA). Performance evaluation results show that this nature inspired heuristic 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 a suitable 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 im-

plementation and tuning of QSGA. 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. [17–21]. 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 [22,23]. Approaches like [23] use latency compensating methods in order to repair the effects of high network jitter. Adaptive rendering strategies like [22] or [1] 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 [13]. 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. [23]. However, the ASR provided to a given avatar i depends on which servers are the neighbor avatars of 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 to server s. A partitioning method designed to provide 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 [10]. 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 [16]. In order to make this paper self-contained, in this section we will briefly describe this function.

Equation 1 represents the evaluation function proposed measuring the QoS, composed of three terms. This function measures the quality of a 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

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

This function is the sum of three different functions or terms. The term $h_{\mathrm{cpu}}(i)$ is a function of the estimated percentage of CPU utilization in each server i. Fig. 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.

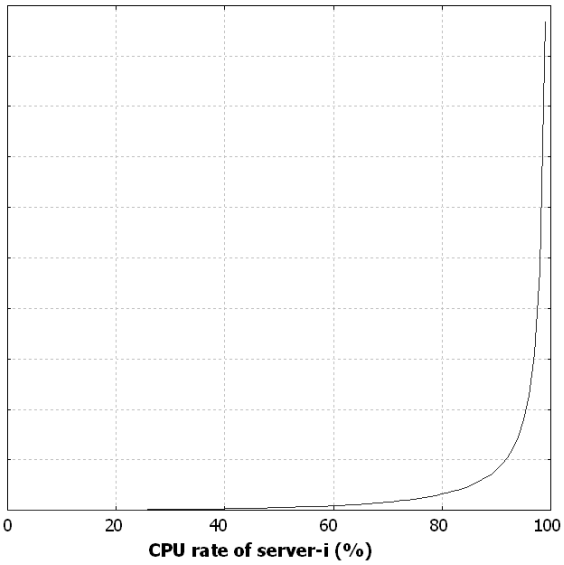

Fig. 2. Behavior of $h_{\mathrm{cpu}}(i)$

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

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

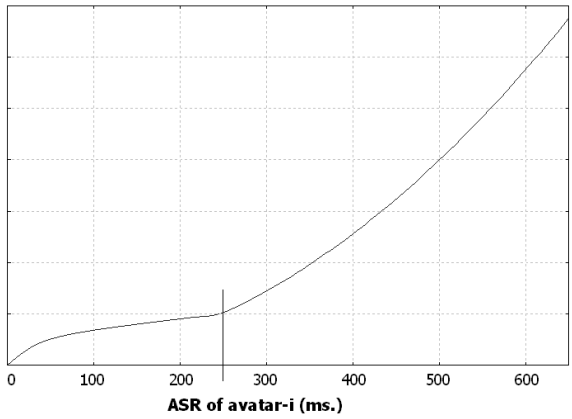

Fig. 3. Behavior of $h_{\mathrm{asr}}(i)$

sections, as shown in Fig. 4. This function penalizes partitions that migrates more than 30% of avatars.

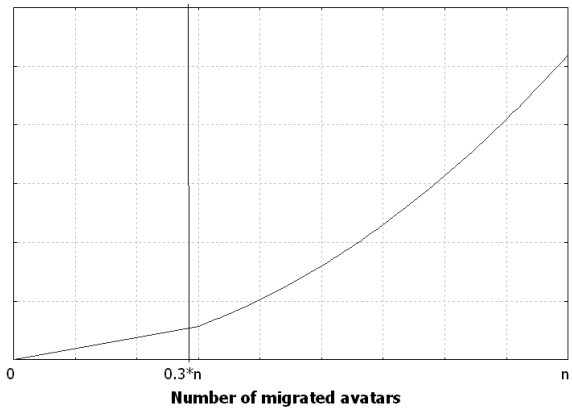

Fig. 4. Behavior of n_{m}.

Thus, the QoS problem in DVE systems consists of finding 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 [17], heuristic procedures can provide near-optimal solutions that actually improve the QoS offered by DVE systems.

3. A Genetic Algorithm for providing QoS: QSGA

Genetic Algorithms (GAs) are heuristic search methods based on the concept of evolution by natural selection [24]. A GA starts from an initial population, composed of P *chromosomes*, that evolves following certain rules for optimizing a function until reaching a convergence condition. Each iteration of our GA consists of generating a new population from the existing one by recombining or even mutating chromosomes. A chromosome contains a *genotype* or string representing an individual (a particular solution of the problem) and also a *phenotype* or features that the genotype represents. We will use this additional information in the phenotype for tuning the behavior of the algorithm.

In our particular case, the genotype consists of an array defining the pairs avatar-server. If there are N avatars in the system, this array contains N elements, 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 to which it is assigned to. This information can be estimated starting from both the movement rate of the avatar and the number of neighbors in its AOI [25]. 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 will use F_{QoS} as the fitness function to be 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 [26]. Therefore, the initial population in QSGA is provided by the ALB partitioning method [14]. This load balancing method provides a balanced partition of the DVE system, ensuring low initial values of the term $\sum_{i=0}^{s} h_{\mathrm{cpu}}(i)$. The GA must keep the workload balanced while providing QoS to the maximum number of avatars as possible.

Each iteration consists of generating an *offspring* generation of chromosomes, starting from a *parent* generation. The way that the algorithm provides the next generation determines the behavior of the GA. In order to choose the variation of GA that fits the best to the QoS problem, we have considered two different criteria: the utilization of the boarder avatars to perform the search, and the use of the ALB algorithm for constructing the initial population. The utilization of the boarder avatars consists of exclusively exchanging boarder avatars in each iteration. We have empirically evaluated the four possi-

ble combinations of these two criteria.

For this evaluation (as well as for all the evaluations shown in this paper) we have considered two different DVE configurations, denoted as MEDIUM1 and MEDIUM2 [16]. MEDIUM1 is composed of 250 avatars and 3 servers, and MEDIUM2 is composed of 700 avatars and 10 servers. However, for the sake of shortness we only present here the result for the MEDIUM2 configuration. The results obtained for MEDIUM1 configuration were very similar. For evaluation purposes we have measured both the system cost provided by each variation of GA (the values of F_{QoS}) and also the execution times required by the algorithm in each case.

In order to evaluate the performance of a DVE system, usually three different avatar distributions in the virtual world have been suggested in the literature: uniform, skewed and clustered [6]. 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 evaluated the four possible combinations of the two criteria under these three distributions of avatars in the virtual world.

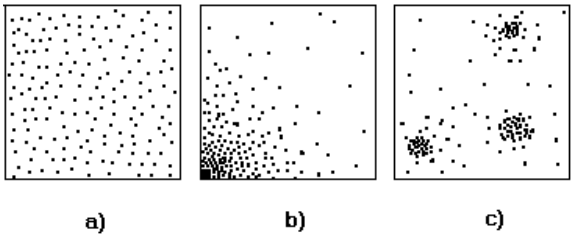

Fig. 5. Distributions of avatars in a 2-D virtual world: a) Uniform b) Skewed c) Clustered.

Fig. 6 shows the F_{QoS} values corresponding to the partitions provided by the four variations of GA that we have considered. We have denoted these variations as follows: Boarder Care for those cases exclusively using boarder nodes in the recombination operator, and No Boarder Care for those cases using any avatars, Low Init for those cases using the ALB for providing the initial partition, and High Init for those cases where the initial population has been randomly generated.

Fig. 6 shows that the No Boarder Care alternatives provide worse results than the Boarder Care ones. The reason for this behavior is that boarder avatars generate most of the inter-server traffic, and this traffic requires most of both the computational

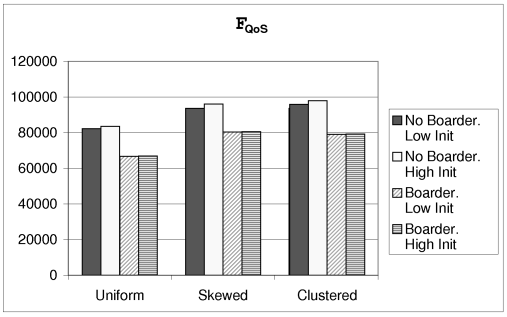

Fig. 6. System costs provided by different variations of GA

and communication bandwidth. Therefore, focusing on the boarder avatars leads to better results than focusing on the other avatars. Also, this figure shows that the use of the partition provided by the ALB algorithm is not significant for the Boarder Care alternatives.

Fig. 7 shows the execution times required for providing the partitions whose F_{QoS} values are shown in Fig. 6. This figure shows that the execution times required to provide the partitions is linearly related to the quality of the provided partitions.

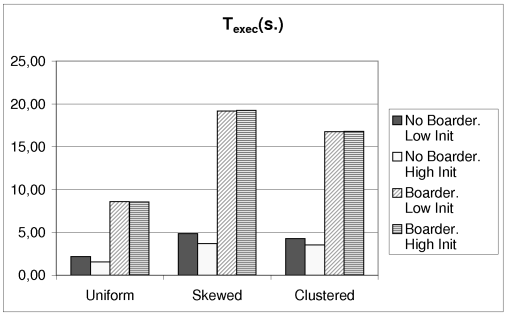

Fig. 7. Execution times required by different variations of GA.

Taking into account the results shown in fig. 6 and 7, we have chosen the combination of Boarder Care and Low Init alternatives for implementing the GA. We have denoted this implementation as QSGA (for Quality of Service Genetic Algorithm). Effectively, for solving the QoS problem we have chosen a recombination technique [26], in such a way that each offspring is generated starting from two parents. However, in order to provide a high diversity,

we have also used non homogeneous hybrid derivation (certain rate of recombination absence in reproduction). Additionally, we use *elitism* in each iteration, in the sense that that some (the best) individuals of a given population are directly passed to the next generation without suffering any variation [26,**?**]. The pseudocode for this algorithm is shown below. In each QSGA 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. Additionally, we have introduced a selective recombination operator on the boarder nodes, in order to avoid that the population can prematurely converge.

In the case of recombination process, in each iteration the first parent for the $i-th$ chromosome of the population is the $i-th$ chromosome of the population in the previous iteration. The second parent 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, two-point crossover or uniform crossover to the parents [26].

One-point crossover technique consists of randomly selecting one crossing point. From the beginning of the offspring to the crossing point the corresponding genotype is copied from the first parent, and from the crossing point up the offspring is copied from the corresponding genotype of the second parent. Fig. 8 shows an example of the one-point crossover. These three methods are randomly selected in our algorithm.

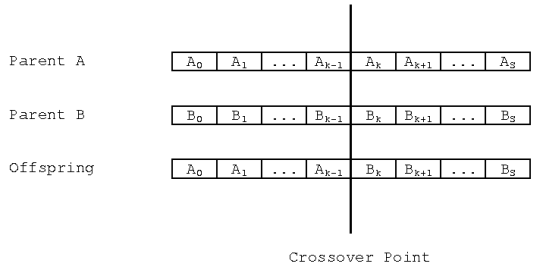

Fig. 8. One-Point Crossover.

Two-point crossover consists of randomly selecting two crossing points in the offspring chromosome. Then, the first part in this offspring is copied from the corresponding genotype in the first parent, the second part (from the first to the second crossing point) is copied from the corresponding genotype in the second parent, and the third part of the offspring is copied again from the corresponding genotype in the first parent. Fig. 9 shows an example of the two-point crossover.

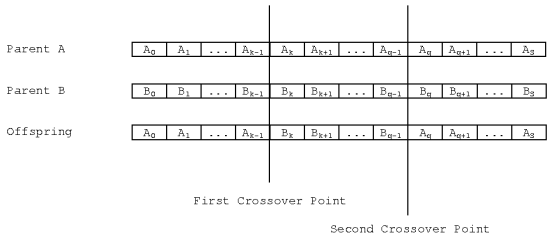

Fig. 9. Two-Point Crossover.

Finally, uniform crossover consists of randomly selecting each assignment in the offspring either from the corresponding assignment in the first parent or from the corresponding assignment in the second parent. Fig. 3 shows an example of the uniform crossover.

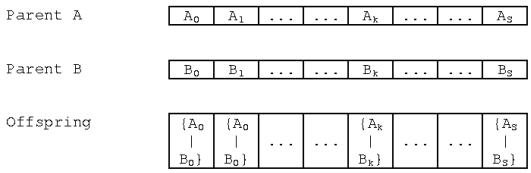

Fig. 10. Uniform Crossover.

In the case of recombination absence, the parent itself is selected as the offspring.

Once the offsprings of iteration t are obtained, if the finishing condition is not reached then a recombination operator is performed on all the chromosomes of that offspring. This recombination process consists of randomly selecting two boarder avatars and exchanging the servers they are assigned to. This process allows to keep diversity while exclusively exploring highly probable solutions. Finally, a mutation can be performed on the resulting offspring. 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 pseudo-code statements (where

Genotype Gt is the resulting population of the previous iteration t, composed of P chromosomes):

```
Iteration t_plus_1 (Genotype Gt)
CONST
  Nelitist /* Num. of elitist chromosomes */
  P      /* Num. of chromosomes in genotype */
  N      /* Num. of avatars in DVE system */
  Recomb  /* recombination rate */

TYPE
  chromosome : int[N]

VAR
  int i
  Anc1, Anc2 : chromosome  /* parents */
  Desc : chromosome      /* offspring */

begin
  Copy_Nelitist_best_of_Gt_to(Gi)
  For i:=Nelitist to P+Nelitist do
    Anc1 := Gt[i]
    a := Reproduction_select(Recomb)
   if a = 0 then /* 0 = Recombination , 1 = No recombination */
        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_plus_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 fulfilled, then the algorithm finishes.

In order to obtain the maximum performance of the algorithm, several parameters must be tuned in the proposed implementation. These parameters are the following ones: the number of chromosomes in the population (P), the number of iterations (Max. Iterations), the mutation rate, number of elitist chromosomes ($Nelitist$), the recombination rate (the percentage of iterations in which recombination is performed), the minimum standard deviation allowed for the convergence condition (diversity threshold), the maximum number of iterations allowed without improving the fitness function (Max repetitions), and the maximum number of iterations. We have empirically tuned these seven parameters in both MEDIUM1 and MEDIUM2 configurations. However, we do not present the tuning data in this paper for the sake of shortness. Instead, Table 1 shows the tuning values that provided the best QSGA performance for the MEDIUM2 configuration. We used these values for the performance evaluation shown in Section 4.

Parameter	Value
Population size (P)	100
Elitism percentage ($\%N_{elitist}$)	50
Mutation rate	0.05
Recombination rate	0.75
Max. Iterations	300
Max Repetitions	50
Diversity threshold	0.005

Table 1
Results of tuning QSGA parameters.

4. Performance Evaluation

In this section, we present the performance evaluation of the genetic algorithm described in the previous section. 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 [16]. We have empirically tuned these other heuristics in the same MEDIUM1 and MEDIUM2 configurations described in the previous

section. Since the considered heuristic methods are stochastic procedures, we have performed 100 executions of each method and we have computed the average value for each of the performance measures shown in the following tables (these average values have been rounded due to space limitations). Therefore, each value in the tables represents the average value obtained after 100 different executions. The standard deviation of the different executions was not higher than 20% of the shown values in any case.

Table 2 shows the results obtained for MEDIUM1 and MEDIUM2 configurations when avatars are uniformly located in the virtual world. Each column in the table shows the results for a given partitioning algorithm. 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% [15]. The second row shows the system cost (in terms of quality function F_{QoS} values) 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[16]. 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.

	MEDIUM1			MEDIUM2		
	SA	GRASP	QSGA	SA	GRASP	QSGA
Max Ut. (%)	8	9	8	17	19	16
F_{QoS}	17753	17352	17495	69633	62048	61558
QoS	239	249	245	526	619	621
$\Gamma(P_0)$	13	15	13	54	101	101
T_{exe} *(s.)*	0.45	0.37	0.47	7,68	7,31	9,07

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

Table 2 shows that the considered methods manage to keep the system below the saturation point, since the maximum estimated utilization percentage does not reach 20% in any column. These values also indicate that a uniform distribution of avatars generate a low level of workload. 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 (250/3 for MEDIUM1 configuration and 700/3 for MEDIUM2 configuration). This table also shows that for a uniform distribution of avatars, QSGA method does not provide a significant performance improvement with regard to GRASP or SA heuristic methods. Although it provides a final partition with a slightly lower value of F_{QoS}, the required execution time is longer than the one required by GRASP or SA methods for MEDIUM2 configuration. Similar results are obtained for MEDIUM1 configuration. However, it is very unlikely that all avatars in a DVE system show a uniform movement pattern. DVE systems usually have certain "hot points" where avatars tend to head for [27]. In other cases, like 3-D networked games, these hot-points are game resources (energy, weapons, etc.) that dynamically appear and disappear, and only avatars located within a given radius of the hot-point tend to approach these points [28]. That is, non uniform distributions like the skewed or clustered distributions are more realistic than the uniform distribution of avatars.

Table 3 shows the results for MEDIUM1 and MEDIUM2 configurations when avatars are located in the virtual world following a skewed distribution. In this case, the maximum utilization rates increase with regard to the ones in Table 2, showing that in this distribution the workload generated with the same number of avatars is greater. Table 3 clearly shows how for this distribution of avatars and for MEDIUM2 configuration QSGA is able to provide a partition with lower (better) values of F_{QoS} than the ones provided by GRASP or SA. For this configuration QSGA is capable of increasing the number of avatars provided with QoS in a 24% when compared with GRASP method, and increasing it in a 48% when compared to SA method. However, for MEDIUM1 configuration QSGA provides QoS to a lower number of avatars than GRASP method. Nevertheless, it requires the shortest execution time in order to provide the resulting partition for both configurations. These results suggest that QSGA provides comparatively better results as the size of the DVE system grows, adding scalability to the DVE system.

Finally, Table 4 shows the results for both configurations when avatars are located in the virtual world following a clustered distribution. For this distribution of avatars, the maximum utilization rates still increase with regard to Table 3. This means that this is the distribution where avatars generate the greatest workload. In this case, for MEDIUM2 con-

	MEDIUM1			MEDIUM2		
	SA	GRASP	QSGA	SA	GRASP	QSGA
Max Ut. (%)	26	29	24	53	58	54
F_{QoS}	20322	19370	19796	88426	82857	75326
QoS	222	234	232	290	347	431
$\Gamma(P_0)$	54	55	49	194	81	169
T_{exe} *(s.)*	3.77	2.08	1.67	84.08	20.07	18.43

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

figuration QSGA 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 this value is 415 and 349, respectively. Nevertheless, the most spectacular improvement is done in terms of execution times. The execution time required by QSGA is around half the time required to execute GRASP method. However, the results for MEDIUM1 configuration are similar or even worse (in terms of number of avatars provided with QoS) than those obtained with GRASP. As for the case of the skewed distribution, these results show that the potential of QSGA can be fully exploited in large scale DVE systems.

	MEDIUM1			MEDIUM2		
	SA	GRASP	QSGA	SA	GRASP	QSGA
Max Ut. (%)	28	36	28	69	72	70
F_{QoS}	21307	19868	20354	92677	88960	84774
QoS	213	237	226	349	415	436
$\Gamma(P_0)$	69	67	66	95	82	120
T_{exe} *(s.)*	4.34	3.01	1.73	48.89	43.66	23.40

Table 4
Results for a DVE system with a clustered distribution of avatars.

All these results show that the performance of QSGA increases as the distribution of avatars in the virtual world generates a greater workload. Since the purpose of the method is to provide QoS to the maximum number of avatars and as longer as possible, these results validate QSGA as a very efficient heuristic method for providing QoS in DVE systems.

5. Conclusions and Future Work

In this paper, we have proposed a 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.

Due to its ability of both finding good search paths and keeping diversity, this technique can provide significantly better solutions than other heuristic methods with shorter execution times. Thus, the evaluation results show that although the newly 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. As a result, it provides 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 the maximum number of avatars as longer as possible (regardless of the distributions of avatars in the virtual world), these results validate QSGA as an efficient heuristic method that can actually improve the QoS offered by DVE systems.

As a future work to be done, we plan to tackle the problem with a multi-objective algorithm.

Acknowledgments

This paper is supported by the Spanish MEC under grant TIC2003-08154-C06-04.

References

[1] S. Singhal, M. Zyda, Networked Virtual Environments, ACM Press, 1999.

[2] J. S. Dias, R. Galli, A. C. Almeida, C. A. C. Belo, J. M. Rebordao, mWorld: A multiuser 3d virtual environment, IEEE Computer Graphics and Applications 17 (2) (1997) 55–65.

[3] D. Miller, J. Thorpe, SIMNET: The advent of simulator networking, Proceedings of the IEEE 83 (8) (1995) 1114–1123.

[4] T. Nitta, K. Fujita, S. Cono, An application of distributed virtual environment to foreign language, in: 30th Annual Frontiers in Education Conference, vol. 1, 2000, pp. F1G/9–F1G15.

[5] M. Lewis, J. Jacboson, Game engines in scientific research: Introduction, Communications of the ACM 45 (1) (2002) 27–31.

[6] J. C. Lui, M. Chan, An efficient partitioning algorithm for distributed virtual environment systems, IEEE

Trans. Parallel and Distributed Systems 13 (3) (2002) 193–211.

[7] D. B. Anderson, J. W. Barrus, J. H. Howard, C. Rich, C. Shen, R. C. Waters, Building multiuser interactive multimedia environments at merl, IEEE MultiMedia 2 (4) (1995) 77–82.

[8] F. C. Greenhlagh, Awareness-based communication management in massive systems, Distributed Systems Engineering 5 (3) (1998) 129.

[9] H. Abrams, K. Watsen, M. Zyda, Three-tiered interest management for large-scale virtual environments, in: VRST '98: Proceedings of the ACM symposium on Virtual reality software and technology, ACM Press, New York, NY, USA, 1998, pp. 125–129.

[10] K. Lee, 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), ACM, 2003, pp. 160–168.

[11] H. Trefftz, I. Marsic, M. Zyda, Handling heterogeneity in networked virtual environments, Presence: Teleoperators and Virtual Environments 12 (1) (2003) 37–51.

[12] M. V. Capps, The quick framework for task-specific asset prioritization in distributed virtual environments, in: VR '00: Proceedings of the IEEE Virtual Reality 2000 Conference, IEEE Computer Society, Washington, DC, USA, 2000, p. 143.

[13] P. Morillo, J. M. Orduña, M. Fernández, J. Duato, On the characterization of distributed virtual environment systems, in: Euro-Par 2003 - Lecture Notes in Computer Science 2790, ACM, Springer-Verlag, 2003, pp. 1190–1198.

[14] P. Morillo, J. M. Orduña, M. Fernández, 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), IASTED, ACTA Press, 2003, pp. 256–261.

[15] P. Morillo, J. M. Orduña, M. Fernández, J. Duato, Improving the performance of distributed virtual environment systems, IEEE Transactions on Parallel and Distributed Systems 16 (7) (2005) 637–649.

[16] P. Morillo, J. M. Orduña, M. Fernández, 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, Springer-Verlag, 2004, pp. 661–670.

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

[18] B. Caminero, C. Carrión, F. Quiles, J. Duato, 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, pp. 220–226.

[19] Z. Wang, J. Crowcroft, Quality-of-service routing for supporting multimedia applications, IEEE Journal of Selected Areas in Communications 14 (7) (1996) 1228–1234.
URL `citeseer.ist.psu.edu/wang96quality.html`

[20] S. Chen, K. Nahrstedt, Distributed quality-of-service routing in high-speed networks based on selective probing, in: LCN, 1998, pp. 80–89.
URL `citeseer.ist.psu.edu/499012.html`

[21] P. Bhaniramka, W. Sun, R. Jain, Quality of service using traffic engineering over MPLS: An analysis, in: Proceedings of 25th Annual IEEE Conference on Local Computer Networks (LCN 2000), 2000, pp. 238–241.

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

[23] T. Henderson, S. Bhatti, Networked games: a QoS-sensitive application for qos-insensitive users, in: Proceedings of the ACM SIGCOMM 2003, ACM Press / ACM SIGCOMM, 2003, pp. 141–147.

[24] R. L. Haupt, S. E. Haupt, Practical Genetic Algorithms, Ed. Willey, 1997.

[25] P. Morillo, J. M. Orduña, M. Fernández, J. Duato, On the characterization of avatars in distributed virtual worlds, in: EUROGRAPHICS' 2003 Workshops, The Eurographics Association, 2003, pp. 215–220.

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

[27] C. Greenhalgh, Analysing movement and world transitions in virtual reality tele-conferencing., in: European Conference on Computer Supported Cooperative Work (ECSCW 97), 1997, p. 313.

[28] M. Matijasevic, K. P. Valavanis, D. Gracanin, I. Lovrek, Application of a multi-user distributed virtual environment framework to mobile robot teleoperation over the internet, Machine Intelligence & Robotic Control 1 (1) (1999) 11–26.