# A Comparison Study of Evolutive Algorithms for Solving the Partitioning Problem in Distributed Virtual Environment Systems

P. Morillo [a], J. M. Orduña [b], M. Fernández [a]

[a] *Instituto de Robótica. Universidad de Valencia.*

[b] *Departamento de Informática. Universidad de Valencia.*
*Av. Vicent Andrés Estellés, s/n. 46100 - Burjassot (Valencia). SPAIN.*

**Abstract**

Fast Internet connections and the widespread use of high performance graphic cards are making Distributed Virtual Environment (DVE) systems very common nowadays. However, there are several key issues in these systems that should still be improved in order to design a scalable and cost-effective system. One of these key issues is the partitioning problem. This problem consists of efficiently assigning clients (3-D avatars) to the servers in the system. In this paper, we present a comparison study of different modern heuristics for solving the partitioning problem in DVE systems, as an alternative to the ad-hoc heuristic proposed in the literature. Performance evaluation results show that some of the heuristic methods can greatly improve the performance of the partitioning method, particularly for large DVE systems. In this way, efficiency and scalability of DVE systems can be significantly improved.

*Key words:* Distributed Virtual Environment, Evolutive Computation

## 1 Introduction

The widespread use of both fast Internet connections and also high performance graphic cards have made possible the current growth of Distributed Virtual Environment (DVE) systems. These systems allow multiple users,

*Email addresses:* Pedro.Morillo@uv.es (P. Morillo), Juan.Orduna@uv.es (J. M. Orduña), Marcos.Fernandez@uv.es (M. Fernández).

working on different computers that are interconnected through different networks (and even through Internet) to interact in a shared virtual world. This is achieved by rendering images of the environment as the user would perceive them if he was located at that point of 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 notified to the neighboring avatars in the shared virtual environment. DVE systems are currently used in different applications [32], such as collaborative design [31], civil and military distributed training [25], e-learning [4] or multi-player games [20,32,1,15].

Designning an efficient DVE system is a complex task, since these system show an inherent heterogeneusness. Such heterogeneusness appears in several elements:

**Hardware.** Each client computer controlling an avatar may have installed differenthardware: A very different range of resources like processor speed, memory size, and graphic card technology can be specified for different client computesr.

**Connection.** Different connections can be found in a single system. From shared medium topologies like Ethernet or Fast-Ethernet to other network connections like ISDN, fiber-optic or ATM can be simultaneously found in some DVE's.

**Communication rate of avatars.** Depending on the application, different communication rates of avatars can be found. For example, the communication rate of avatars in a collaborative 3D environment may greatly differ from the comunication rate of avatars in a 3D virtual military battle.

Additionally, other factors help to increase the complexity of designing an efficient DVE system. Each of them have become nowadays an open research field:

**Data Model.** This concept describes some conceivable ways of distributing persistent or semi-persistent data in a DVE [24]. Data can be managed in a replicated, in a shared or in a distributed methodology.

**Communication Model.** Network bandwidth determines the size and performance of a DVE. The system behavior is related to the way that all the scene clients are connected. Broadcast, peer-to-peer or unicast schemes define different network latency values for exchanging information between avatars.

**View Consistency.** This problem has been already defined in other computer science fields such as database management [3]. In DVE systems, this problem consists of ensuring that all avatars sharing a virtual space with common objects have the same local vision of them.

**Message Traffic Reduction.** Keeping a low amount of messages allows DVE systems to efficiently scale with the number of avatars in the system. Traditionally, techniques like dead-reckoning described in [32] offered some level of independence to the avatars. With network support, broadcast or multicast solutions [10,19] decrease the number of messages used to keep a consistent state of the system.

Most of the issues described above are related to the *partitioning problem* or *p-problem*. This problem consists of efficiently distributing the workload (avatars) among different servers in the system [21]. The partitioning problem may seriously affect the overall performance of the DVE system, since it determines not only the workload that each server must suport, but also the inter-server communication requirements (and therefore the network traffic).

Some methods for solving the partitioning problems have been already proposed [23,?,33]. These methods provide efficient solutions even for large DVE systems. However, there are still some features in the proposed methods that can still be improved. For example, different heuristic search methods can be used for finding the best assignment of clients to servers, instead of using ad-hoc heuristics. In this paper, we present a comparison study of several heuristics for solving the partitioning problem in DVE systems. We have implemented five different heuristics, ranging over most of the current taxonomy of heuristics: Genetic Algorithms (GA) [13], two different implementations of Simulated Annealing [17], Ant Colony Systems (ACS) [7], and Greedy Randomized Adaptive Search (GRASP) [6]. Performance evaluation results show that the execution cost of the partitioning algorithm (in terms of execution times) can be dramatically reduced, while providing similar or even better solutions than the ones provided by the ad-hoc heuristic proposed in [23].

The rest of the paper is organized as follows: Section 2 describes the partitioning problem and the existing proposals for solving it. Section 3 describes the proposed implementations of the heuristics considered for this study. Next, Section 4 presents the performance evaluation of the proposed heuristics. Finally, Section 5 presents some concluding remarks.

## 2 The Partitioning Problem in DVE Systems

### *2.1 Architectures for DVE Systems*

Several architectures have been traditionally used for simulating a large set of avatars sharing the same virtual world. Internet multi-player games as Quake [1] and Kali [15] or educational systems as VES [4] are examples of client-server systems (fig. 1b). In these applications, each client computer has a single connection to the only existing server in the system. This server maintains the global state of the simulation, but it becomes a single point of failure in the system. Instead of sending messages to a central server, in peer-to-peer architectures(fig. 1a) avatars exchange messages directly . Several systems have been developed with this architecture, such as NPSNET [10]. Although these systems obtain low latenccies, they do not properly scale. When the number of avatars greatly increases, clients are not able to handle the amount of messages from other avatars and simultaneously offering an interactive 3D virtual world to the user. In order to improve scalability, peer-server and server-network architectures group sets of avatars. Following a peer-server scheme (fig. 1d), systems like ATLAS [19] reduces the volume of information using multicast messaging. However, this architecture will be useful only when multicast protocols are fully available in Internet [?].

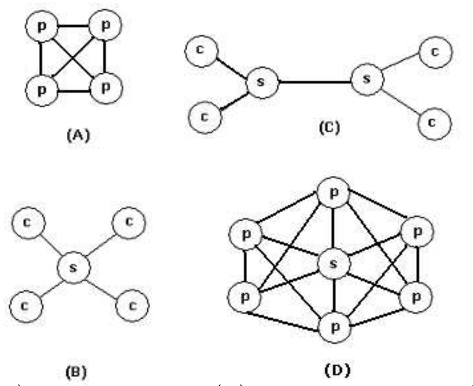

Fig. 1. Architectures: (a) peer-to-peer, (b) server-network, (c) client-server and d) peer-to-server

Architectures based on networked servers are becoming a de-facto standard for DVE systems [32,21,33]. In these architectures, the control of the simulation relies on several interconnected servers. Figure 1c) depicts how multi-platform client computers are attached to only one of the servers of the simulation.

In this architecture, when a client modifies an avatar, it also sends an updating message to its server, that in turn must propagate this message to other servers and clients. Servers must render different 3D models, perform positional updates of avatars and transfer control information among different clients. Thus, each new avatar represents an increasing in both the computational requirements of the application and also in the amount of network traffic. When the number of connected clients increases, the number of updating messages must be limited in order to avoid avoid a message outburst. In this sense, concepts like areas of influence (AOI) [32], locales [2] or auras [12] 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 avatar must notify his movements (by sending an updating message) only to those avatars located in that neighborhood. Depending on their origin and destination avatars, messages in a DVE system can be intra-server or inter-server messages (see figure 2). Inter-server messages are those messages involving two or more servers. On the contrary, intra-server messages are those messages exchanged between avatars whose client computers are attached to the same server. In order to design a scalable DVE systems, the number of intra-server messages must be maximized. Effectively, when clients send intra-server messages they only concern a single server. Therefore, they are minimizing the computing, storage and communication requirements for maintaining a consistent state of the avatars in a DVE system.

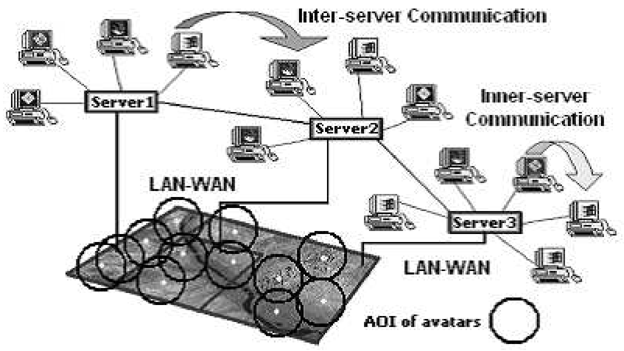

Fig. 2. A multi-server architecture for a basic DVE

The *partitioning problem* consist of efficiently distributing the workload (assigning avatars) among the different servers in the system. Lui and Chan have shown the key role of finding a good assignment of avatars to servers in order to ensure both a good frame rate and a minimum network traffic in DVE systems [21,23]. They propose a quality function, denoted as $C_p$, for evaluating each partition (assignment of avatars to servers). This quality function takes into account two parameters. One of them consists of the computing workload generated by clients in the DVE system, and it is denoted as $C_p^W$. In order to minimize this parameter, the computing workload should be proportionally shared among all the servers in the DVE system, according to the computing resources of each server. The other parameter of the quality function consists of the overall number of inter-server messages, and it is denoted as $C_p^L$. In order to minimize this parameter, avatars sharing the same AOI should be assigned to the same server. Thus, quality function $C_p$ is defined as

$$C_p = W_1\, C_p^W + W_2\, C_p^L \qquad (1)$$

where $W_1 + W_2 = 1$. $W_1$ and $W_2$ are two coefficients that weight the relative importance of the computational and communication workload, respectively. These coefficients should be tuned according to the specific features of each DVE system. Using this quality function (and assuming $W_1 = W_2 = 0.5$) Lui and Chan propose a partitioning algorithm that re-assigns clients to servers

[23]. The partitioning algorithm should be periodically executed for adapting the partition to the current state of the DVE system as it evolves (avatars can join or leave the DVE system at any moment, and they can also move everywhere within the simulated virtual world). Lui and Chan also have proposed a testing platform for the performance evaluation of DVE systems, as well as a parallelization of the partitioning algorithm [23].

The partitioning method proposed by Lui and Chan, known as LOT or *linear optimization technique*, currently provides the best results for DVE systems. However, it uses an ad-hoc heuristic. We propose a comparative study of several heuristics, ranging over most of the current taxonomy of heuristics, in order to determine which one provides the best performance when applied to the partitioning problem in DVE systems. In this study, we propose the same approach of Lui-Chan: using the same quality function, we will obtain an initial partition (assignment) of avatars to servers, and then we will test the implementation of each heuristic to provide a near optimal assignment.

## *2.2 Linear Optimization Technique (LOT)*

Linear Optimization Technique (LOT) was initially published by Lui and Chan in [21] and revisited in [23] from theirs ideas published in [22] about graph theory.

This ad-hoc approach models the 3-D virtual scene as a graph. Each avatar is modelled by a node and two avatars are linked by an edge if their areas of interest (AOI) collide. Using $C_p$ as the evaluation function, LOT provides an efficient partition of this graph following three steps. The first step is named the *recursive bisection procedure* (RBP), and it consists of using a divide-and-conquer procedure that provides an initial partition. Then, the *layering partitioning procedure* (LP) and the *communication refinement partitioning* (CRP) are applied on that initial partition of the graph. Each of these procedures performs workload balancing and minimizes the number of inter-server messages, respectively.

Figure 3 shows the partitions that this method would provide when applied to a small DVE system. In this example, a DVE composed by 10 avatars is simulated with three identical servers. Nodes and edges obtained from the associated graph has been labelled. The label of a node (avatar) represents a estimation of the workload generated by this avatar to the server where it is going to be assigned. Ranging from 1 to 10, the label of each edge represents the nearness of these two avatars. Figure 3 a) represents the result obtained by a RBP phase. Although the number of avatars assigned to each server seems to be balanced, the workload that those avatars generate must be also uniform in order to achieve actual workload balancing. By adding the labels of all the nodes assigned to the same server, we obtain that the workload assigned to each server is 16, 7 and 12 units of workload, respectively. Then, CRP balances (figure 3 b)) the existing workload in sets of 12, 11 and 12 units. At that point, the number of inter-server messages generated by these sets of avatars is reduced by LP, as shown in figure 3 c). Since strategies of CRP and LP techniques could be opposed, this couple of steps is repeated three times.

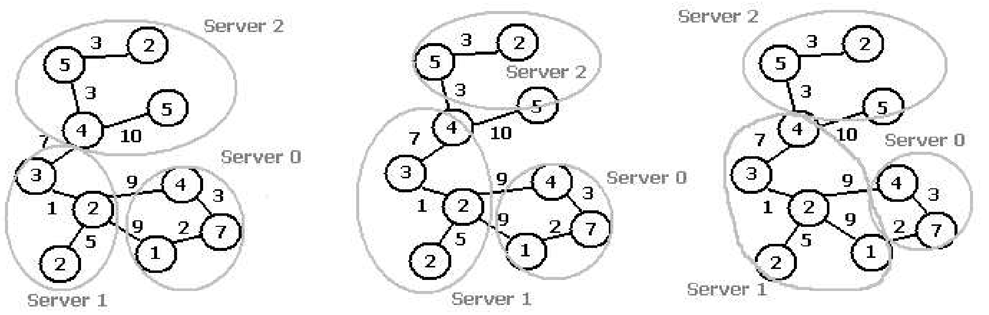

Fig. 3. Partition of a DVE performed by LOT method: (a) RBP (B) CRP and c) LP

### *2.3 Other Approaches to the Partitioning Problem*

In addition to LOT technique, two other relevant approaches have been also proposed for solving the partitioning problem. One of them divides the whole virtual scene into hexagonal cells, and each cell is mapped to a multicast group [?]. With this division, all avatars located in the same cell share the same multicast address. Since the size of AOI is bigger than the area of the cells, avatars can listen different multicast addresses but they can only send information to the cell where they are located.

Figure 4 shows an example of this technique. In this figure, the represented avatar is assigned to group A and also listen multicast information from groups B, C, E, F and G. In order to ensure and efficient management, this algorithm generate a group leader for each cell. The oldest avatar is selected as the group leader, using a timestamp mechanism. This leader controls how avatars join and leave the multicast group, and he also implements a flow control mechanism for the messages.

In spite of its apparent scalability, this approach presents several problems. First, it does not obtain good performance when avatars are located following a non-uniform distribution. Second, system performance is very low when avatars can fastly cross the virtual scene. This is due to the associated cost of leaving and joining different multicast groups. Finally, multicast is not fully implemented within the Internet.

Another different approach was proposed by Tam in [33]. In this method, dynamic concepts associated to avatars like AOI, aura or locale are rejected. This partitioning method divides the scene in square cells. The side length of the cells $S$ is related to the radius of the AOI of avatars, following the next relation:

$$\frac{S}{2} \leq AOI \leq S \tag{2}$$

A graph representation of the virtual scene is obtained from the volume of avatars contained in each cell, as also shown in Figure fig. 4. Next, this graph is divided in partitions and each partition is assigned to a server of the DVE. In order to accomplish this division an exhaustive and a greedy algorithms are compared. Using a quality different from $C_p$, these algorithms take also into account the number of inter-server messages and workload balancing among the servers. Although this approach provides a fast way of solving the partitioning problem, the performance of the static partitioning is quite low when avatars show a clustered distribution. In this case, the servers controlling the crowded areas are overloaded, increasing the overall cost of the quality function.

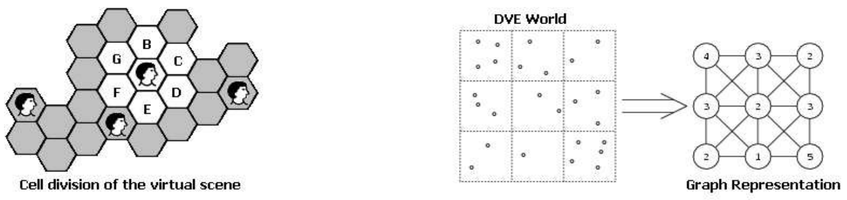

Fig. 4. Other approaches: cell division and grpah representation

## 3 Metaheuristic Procedures

Evolutive computing has become a important paradigm in the development of high-quality solution for NP-Complete problems. This paradigm, which models computational problems as natural processes, is used to provide solutions for complex problems that in most cases could not be solved by using deterministic computing techniques. Problems such as multi-objective optimization have been successfully adapted using these strategies [5]. In this section, we present five implementations of different heuristics for solving the partitioning problem in DVE systems. Following the approach presented by Lui and Chan (and using the same quality function $C_p$), the idea is dynamically applying a heuristic search method that provides a good assignment of clients to servers as the state of the DVE system changes. All the presented solutions has been adapted in the same way. That is, a initial partition obtains good assignments only for several avatars, while the evolutive algorithm itself performs successive refinements of the initial partition that lead to a near optimal partition of the DVE system. Also, the evolutive method is used periodically for updating the obtained partition to the current state of the DVE system (avatars change their locations, new avatars can join the system and some avatars can leave the system at any time).

In this section, we describe the implementation of each heuristic search method and the tuning of its parameters for solving the partitioning problem. Concretely, this tuning has been performed on the DVE evaluation test described by Lui and Chan in [23]. This test defines two kinds of DVE system: a SMALL virtual world, composed of 13 avatars and 3 servers and also a LARGE world, composed of 2500 avatars and 8 servers. However, for the sake of shortness we only present here the results obtained for large worlds. Effectively, the purpose of solving the partitioning problem is to provide scalable DVE systems. Therefore, the partitioning method must efficiently work in LARGE virtual worlds. On other hand, since the performance of the method may heavily depend on the location of avatars, this evaluation test also considers three different distributions of avatars: uniform, skewed, and clustered distribution.

## *3.1 Obtaining the Initial Population*

All of the implemented heuristics start from an initial partition (assignment) of the $n$ avatars in the DVE system. We tested several clustering algorithms for obtaining this initial partition. Although they are not shown here due to space limitations, we obtained the best results for a *density-based algorithm* (DBA) [9].

This algorithm divides the virtual 3-D scene in square sections. Each section is labelled with the number of avatars that it contains ($na$), and all the sections are sorted (using Quick-sort algorithm) by their $na$ value. The first $S$ sections in the sorted list are then selected and assigned to a given server, where $S$ is the number of servers in the DVE system. That is, all the avatars in a selected region are assigned to a single server. The next step consists of computing the mass-center ($mc$) of the avatars assigned to each server. Using a round-robin scheme, the algorithm then chooses the closest free avatar to the $mc$ of each server, assigning that avatar to that server, until all avatars are assigned.

The proposed implementation of the DBA method consists of the following steps (expressed as pseudo-code statements):

```
program Initial_Partition (avatar, Int n, Int S)

const
   n_sections = 25x25

type Cell
    sum,idx: Int
```

```
var
    assigned,represent:Int[]
    pivot,elect,ncentr:Int
    min_dis,dist_tmp  :Real
    na                :Cell[n_sections]

begin
    DivideSceneInSquareSections(n_sections)
    for i:=0 to n_sections do
        na[i].sum:=CountAvatarsInSection(i)
        na[i].idx:=i
    end_for
    QuickSort (na)
    for i=0 to S do
        representant [i]= ObtainMC (na[i].idx)
    end_for;
    pivot := 0, elect := -1, ncentr:=0
    for i:=0 to n do
        elect:=-1,min_dis := 100000
        for  j:=0 to n do
           if (assigned[j] = NOT_ASSIGNED)
                dist_tmp:= EuclideanDistance(avatar[j],represent([ncentr])
                if (dist_tmp < min_dis)
                    pivot   := j
                    min_dis := dist_tmp
                endif
           endif
        end_for
        avatar[pivot].assignment := assigned[pivot] := ncentr
        ncentr : (ncentr + 1) mod S
    end_for
end
```

Since the assignment of avatars follows a round-robin scheme, this algorithm provides a good balancing of the computing workload (the number of avatars assigned to each server does not differ in more than one). On other hand, avatars that are grouped in a small region and close to the mass-center of a server will be assigned to that server by the density-based algorithm. Additionally, since these avatars are located so closely, they will probably will share the same AOI. Therefore, the density-based algorithm also provides an initial partition with low inter-server communication requirements for those avatars.

However, the assignment of avatars equidistant (or located far away) from the mass-center of the servers is critical for obtaining a partition with minimum inter-server communication requirements (minimum values of the quality function $C_p$), particularly for large virtual worlds with only a few servers. Density-based algorithm inherently provides good assignments for clustered avatars, but it does not properly focus on the assignment of these critical avatars. Each of the following evolutive methods should be used at this point to search a near optimal assignment that properly re-assigns these avatars.

### *3.2 Genetic Algorithms (GA)*

This heuristic consists of a search method based on the concept of evolution by natural selection [13]. GA starts of an initial population (the initial partition) and then it evolves a certain number of *generations* (iterations), providing an *evolved population* (final solution). The proposed implementation for solving the partitioning problem, proposed initially in [29], starts with a population composed of a set of elements called *genomes* or *chromosomes.* The number of chromosomes is the number of partial solutions that each iteration must maintain. Each chromosome is defined by a descriptor vector containing a given assignment of avatars to servers. The length of this vector is equal to the number of *border avatars* (BA) within the DVE system. BA is an avatar that although it lies within the AOI of another avatar, it is not assigned to the same server. These kinds of avatars offers a higher probability of giving a successful permutation, since they minimize inter-server communication.

Starting from the initial population, this approach applies an auto-fertilization mechanism for generating new chromosomes [16]. This mechanism is based on a single-point random crossover, where each generation is found by exchanging some elements of the restricted population in such a way that in each of the $N$ chromosomes two border avatars assigned to different servers are randomly chosen and exchanged. Thus, an iteration performed on a population of $N$ chromosomes will produce a new population of $2N$. From these $2N$ chromosomes, the $N$ elements with the lower value of $C_p$ will be chosen.

GA is also capable of escaping from local minima due to the *mutation* process. Once the child-vector has been obtained, mutation involves changing at random the server assigned to one of the elements of the population. Some other mutations such as the modification of several elements or the crossover between the characteristics of pairs of chromosomes have also been tested. But after several tests it has been found that the mutation of just one "bit" is the one offering the best results. It is important to mention that the mutation is a random process controlled by a parameter. This parameter needs to be carefully chosen for each specific experiment in order to achieve solutions with low $C_p$ System Cost.

Next code describes the behavior of the proposed approach based on GA. It has been expressed as pseudo-code statements:

```
program GA (Int chromo, Int iterations, Real mut_rate)

var
   B,temp_cost,Cp_GA :Real
   av_i,av_j,av_k    :Integer

begin
   Initial_Partition (DBA)
   B := ObtainBorderAvatars()
   Cp_GA := Compute_Cp()
   For i:=0 to iterations do
       For j:=0 to chromo do
            SelectAndCopyChromosome(j)
            Choose2DifRamdomAvatars(B,av_i,av_j)
            ExchangeServerAssignment(av_i,av_j)
            temp_cost := Compute_Cp()
            if (HaveItoMutate(mut_rate))
                Chose1RandomAvatar(av_i)
                ForceServerAssigment(av_i)
            endif
            AddChromosomeToPopulation(this)
       end_for
       CWPSortPopulationByCp(2*chromo)
       Cp_GA := SelectBestIndividuals(chromo)
   end_for
end
```

Figure 5 shows a example of generation of new avatars in a small DVE system where 6 BA have been obtained from the full set of avatars. These avatars define a chromosome, and they can be assigned to any of the 3 servers in the DVE system. This figure represents a possible crossover and mutation on this chromosome.

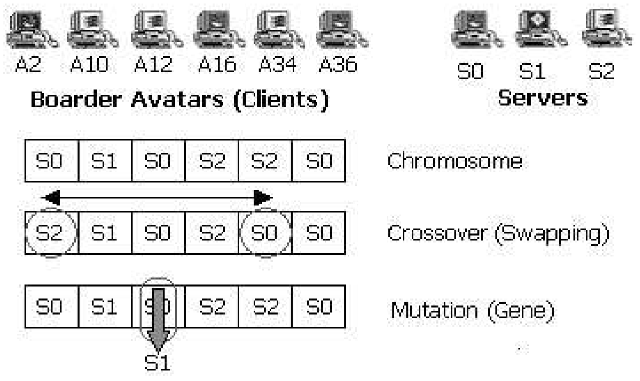

Fig. 5. Generation of new individuals in GA approach

The main parameters to be tuned in GA method are the population size $P$, the number of iterations $N$ and the mutation rate $M$. Figure 6 depicts the values of que quality function $C_p$ (denoted as system cost) as the number of generations and individuals grows. In both cases $C_p$ decreases significantly while the algorithm does not reach a threshold value. From this threshold value (close to 300 for the number of generations and 15 for the population size) the quality of the obtained solutions remains constant, showing the impossibility of finding better solutions in this search domain. Therefore it has no sense spending more time in searching new partitioning solutions.

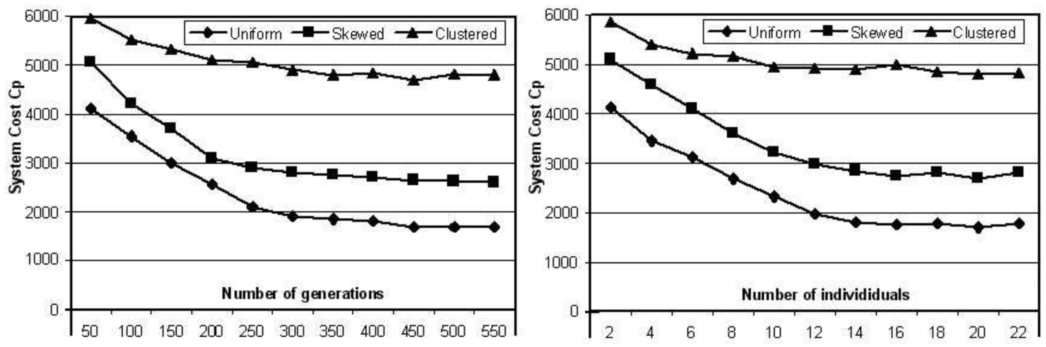

Fig. 6. Values of the quality function $C_p$ for different number of generations and population sizes

Figure 7 shows the tuning of the mutation rate in a given DVE system. The behavior of the algorithm is different for this parameter. In this case, the algorithm is able to provide a high-quality solution when mutation rate is close to 1%.

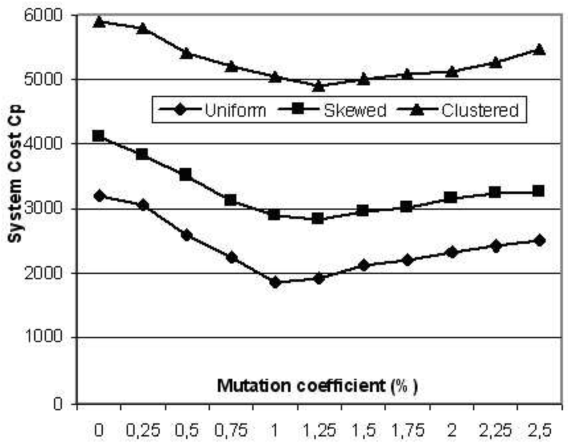

Fig. 7. Values of the quality function $C_p$ for different mutation coefficients

If GA approach uses values much lower than 1% for this parameter, the system can be trapped in a local minimum. In the opposite case, when GA selects rates higher than this threshold value, then the search method spends too much CPU time testing useless solutions.

As a conclusion of this tuning phase, we can state that for this particular apllication GA method provides good results for a population of 15 individuals, a mutation probability of 1% and 100 iterations as stop criterion.

Ant Colony Systems (ACS) search method is based on the behavior shown by ant colonies when searching possible paths to their food [7]. The use a hormone called *pheromone* to communicate among them. The path that a given ant follows when searching food depends on the amount of pheromone each possible path contains. Additionally, when a given ant chooses a path to the food, she adds pheromone on that path, thus increasing the probability for the ants behind her to choose the same path. This system makes the food search to be initially random. Nevertheless, since the ants that choose the shortest path will add pheromone more often, the probability for choosing the shortest path increases with time (positive feedback). On the other hand, pheromone evaporates at a given rate. Therefore, the associated pheromone for each path decreases if that path is not used during certain period of time (negative feedback). Evaporation rate determine the ability of the system for escaping from local minima.

ACS search method has been implemented in different ways as it has been used for solving several discrete optimization problems [8]. Concretely, we propose a new implementation of the ACS search method, to be used for solving the partitioning problem in DVE systems. This implementation was originally proposed in [27]. As the rest of evolutive approaches, this implementation follows two steps. First, the DBA initial partition (described in section 3.1), obtains good assignments only for several avatars. At that point ACS algorithm performs successive refinements of the initial partition that lead to a near optimal partition.

The first step in the ACS method is to select the subset of border avatars from the set of all the avatars in the system. A given avatar is selected as a border avatar if it is assigned to a certain server $S$ in the initial partition and any of the avatars in its AOI is assigned to a server different from $S$. For each of the border avatars, a list of candidate servers is constructed, and a level of pheromone is associated to each element of the list. This list contains all of the different servers that the avatars in the same AOI are assigned to (including the server that the avatar is currently assigned). Initially, all the elements in the list of candidate servers have associated the same pheromone level.

ACS method consists of a population of *ants*. Each ant consists of performing a search through the solution space, providing a given assignment of the $B$ border avatars to servers. The number of ants $N$ is a key parameter of the ACS method that should be tuned for a good performance of the algorithm. Each iteration of the ACS method consists of computing $N$ different ants (assignments of the $B$ border avatars). When each ant is completed, if the resulting assignment of the $B$ border avatars produces a lower value of the quality function $C_p$, then this assignment is considered as a partial solution, and a certain amount of pheromone is added to the servers that the border avatars are assigned to in this assignment (just the same way that each ant adds pheromone to the search path that she follows). Otherwise, the ant (assignment) is discarded. When each iteration finishes (the $N$ ants have been computed), the pheromone level is then equally decreased in all the candidate servers of all of the border avatars, according to the evaporation rate (the pheromone evaporates at a given rate). ACS method finishes when all the iterations have been performed.

In the process described above, each ant must assign each border avatar to one of the candidate servers for that avatar. Thus, a *selection value* is computed for each of the candidate servers. The selection value $S_v$ is defined as

$$S_v = \alpha \times pheromone \; + \; \beta \times C_p \tag{3}$$

where *pheromone* is the current pheromone level associated to that server, $C_p$ is the resulting value of the quality function when the border avatar is assigned to that server instead of the current server, and $\alpha$ and $\beta$ are weighting coefficients that must be also tuned. The server with the highest selection value will be chosen by that ant for that border avatar.

On other hand, when a partial solution is found then the pheromone level must be increased in those servers where the border avatars are assigned to in that solution. The pheromone level is increased using the following formula:

$$pheromone \; = \; pheromone \; + \; Q \times \frac{1}{C_p} \tag{4}$$

Following this description, the proposed implementation of the ACS search method consists of the following steps:

```
program ACS (Int Ants, Int iterations, Real evap_rate)

const
   alpha = 1.0
   beta  = 7.0
   Q     = 1000

var
   temp_sol   :Real[Number_of_Avatars]
   L          :Integer[]
   B,Cp_ACS,temp_cost :Real

begin
   Initial_Partition (DBA)
   B := ObtainBorderAvatars()
   Cp_ACS := Compute_Cp()
   For i:=0 to iterations do
       For j:=0 to N do
           For k:=0 to B do
               L:=ChooseServer(alpha,beta,Q)
           end_for
           temp_sol := Compose_Solution(B)
           temp_cost:= Obtain_Cp(temp_sol)
           if (temp_cost < Cp_ACS)
               Cp_ACS := temp_cost
               IncreasePheromone (B,Q)
           endif
       end_for
       DecreasePheromone(evap_rate)
   end_for
end
```

Like in GA approach, there are some parameters in ACS search method that must be properly tuned. In particular, the values for the number of ants $N$, the pheromone evaporation rate and the number of iterations that ACS method must perform should be tuned.

Figure 8 shows the values of the quality function $C_p$ (denoted as *system cost*) reached by the ACS method when different number of ants and iterations are considered. It shows that $C_p$ decreases when the number of iterations increases, until value of 25 iterations is reached. The same behavior manifests when the number of ants is grown. In this case is 100 the top value. From that point, system cost $C_p$ slightly increases or remain constant, depending on the considered distribution of avatars. The reason for this behavior is that the existing pheromone level keeps the search method from finding better search paths even when more iterations are performed. Thus, the number of iterations and ants selected for ACS method has been 25 and 100 respectively.

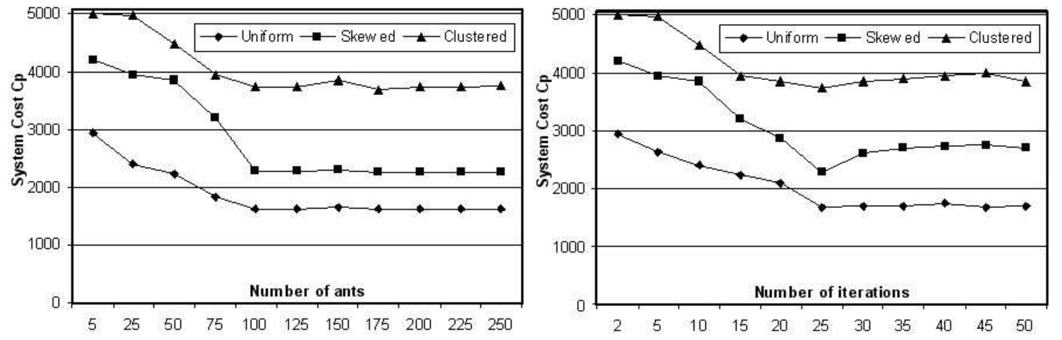

Fig. 8. Values of the quality function $C_p$ for different number of iterations.

Finally, figure 9 shows the values of $C_p$ reached by the ACS method when different pheromone evaporation rates are considered. This figure shows on the x-axis the percentage decreasing in pheromone level that all candidate servers suffer after each iteration. It shows that for all the considered distributions $C_p$ decreases when the evaporation rate increases, until a value of 1% is reached. The reason for this behavior is that for evaporation rates lower than 1% the pheromone level keeps the search method from escaping from local minima, thus decreasing performance. From that point, system cost $C_p$ also increases, since pheromone evaporation is too high and the search method cannot properly explore good search paths. Thus a coefficient of 1% has been selected as the optimal value of evaporation rate.

Additionally, we have performed empirical studies in order to obtain the best values for $\alpha$, $\beta$ and $Q$ coefficients. Although the results are not shown here for the sake of shortness, we it has obtained the best behavior of the ACS method for $\alpha = 1.0$, $\beta = 7.0$ and $Q = 1000$. These are the values that the pseudo-code shown above uses for ACS algorithm.

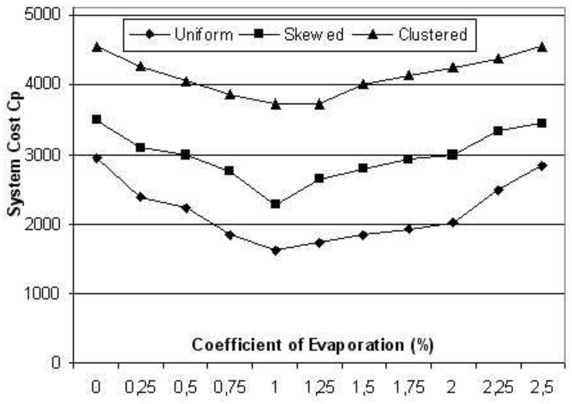

Fig. 9. Values of the quality function $C_p$ for different evaporation rates.

### *3.4 Simulated Annealing (SA)*

SA was initially proposed for solving complex optimization problems [17]. Although some authors do not consider SA as an evolutive technique, it can be modeled as a simplified version of GA approach. SA has shown to be effective in the solution of a large set of applications such as staff scheduling problems, timetabling problems and locomotive allocation problem in railway networks [18].

This heuristic search method is based on a thermodynamic theory establishing that the minimum energy state in a system can be found if the temperature decreasing is performed slowly enough. Simulated Annealing (SA) is a heuristic search method that always accepts better solutions than the current solutions, and also accepts worse solutions according to a probability system based on the system temperature. SA starts with a high temperature (a high probability of accepting a worsening movement), and in each iteration system temperature is decreased. Thus, SA can leave local minima by accepting worsening movements at intermediate stages. The search method ends when system temperature is so low that worsening movements are practically impossible. Since the method cannot leave local minima, it cannot find better solutions neither (the algorithm ends when certain amount of iterations $N$ are performed without finding better solutions).

The proposed partitioning method based on SA was previously presented in [28]. As the rest of evolutive procedures, this approach starts from an initial population. This initial population has been obtained with DBA algorithm. In SA method each iteration consists of randomly selecting two different border avatars assigned to different servers. Then, the servers where this two border avatars are assigned to are exchanged. If the resulting value of the quality function $C_p$ is higher than the previous one plus a threshold $T$, that change is rejected. Otherwise, it is accepted (the search method must decrease the value of the quality function $C_p$ associated with each assignment). The threshold value $T$ used in each iteration $i$ of the search depends on the rate of temperature decreasing $R$, and it is defined in this implementation as

$$T = R - \left(\frac{R \times i}{N}\right) \tag{5}$$

where $N$ determines the finishing condition of the search. When $N$ iterations are performed without finding a partition that decreases the value of quality function $C_p$, then the search finishes.

The next code shows the described implementation based on SA:

```
program SA (Int iterations, Real dec_t_rate)

var
   B,temp_cost,Cp_SA :Real
   delta_sup         :Real
   av_i,av_j,        :Integer

begin
   Initial_Partition (DBA)
   B := ObtainBorderAvatars()
   Cp_SA := Compute_Cp()
   For i:=0 to iterations do
      Choose2DifRamdomAvatars(B,av_i,av_j)
      ExchangeServerAssignment(av_i,av_j)
      temp_cost := Compute_Cp()
      delta_sup := dec_t_rate - (dec_t_rate*i/interations)
      if (temp_cost - delta_sup < Cp_SA)
         Cp_SA := temp_cost
      else
         ExchangeServerAssignment(av_i,av_j)
   end_for
end
```

In order to improve the performance of SA method, we measured the impact of exchanging groups of avatars, instead of exchanging one avatar in each iteration. Table 1 compares the results (in terms of both the value for the quality function $C_p$ and also in terms of execution times) of exchanging groups of 2, 3 and 4 avatars. We tested each option under three different distributions of avatars in the virtual world (uniform, skewed and clustered). These distributions are detailed in section 4. On ther hand, these results have been obtained for a LARGE world composed by 2500 avatars and 8 servers. This table shows that the best option for SA method is to exchange the lowest number of avatars as possible in each permutation. Therefore, we exchanged a single avatar in all the simulations performed in our study.

| | **2 avatars** | | **3 avatars** | | **4 avatars** | |
|---|---|---|---|---|---|---|
| | $C_p$ | Time (s.) | $C_p$ | Time (s.) | $C_p$ | Time (s.) |
| **Uniform** | 1707,62 | 6,35 | 1808,38 | 6,51 | 1817,90 | 7,02 |
| **Skewed** | 2628,46 | 13,79 | 2826,44 | 14,32 | 2992,30 | 15,82 |
| **Clustered** | 4697,61 | 29,62 | 5046,27 | 30,25 | 5283,705 | 33,98 |

Table 1
Different types of exchanges performed on SA approach

On other hand, the two key issues for properly tuning this heuristic search method are the number of iterations $N$ and the temperature decreasing rate $R$ [18]. Figure 10 shows the performance (in terms of $C_p$ values) obtained with SA algorithm for a LARGE world when the number of iterations increases.

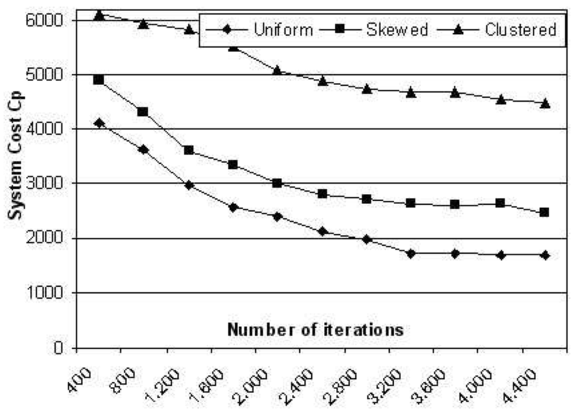

Fig. 10. Values of the quality function $C_p$ for different number of iterations

From figure 10 we can conclude that performing more iterations results in providing better values of $C_p$. However, the slope of this plot decreases from a certain number of iteration. We have considered the value of 3000 iterations as that point, and we have tested GA method with this number of iterations.

System temperature shows a different behavior in terms of $C_p$. Figure 11 shows the values of $C_p$ obtianed when the temperature decreasing rate is modified in a LARGE world. It clearly shows that the quality of the obtained solutions do no follow a lineal progression. Effectively, since the temperature decreasing rate allows SA approach to escape from local minima, a threshold value appears when this parameter is modified. As this rate comes closer to 1.15 algorithm abandons local minima much more fast, and therefore the quality of the obtained solution increases. Beyond this value the risk of accepting inefficient exchange of avatars is too much high and thus the algorithm is unable to find right paths.

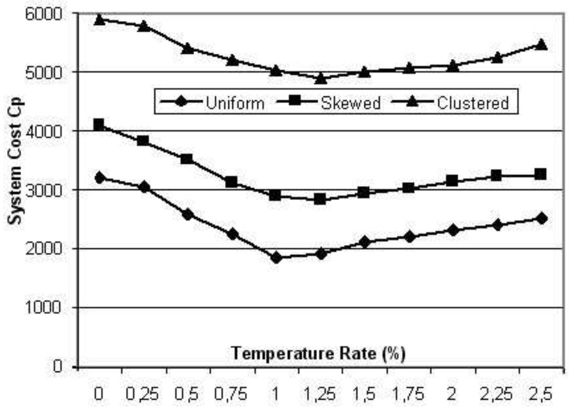

Fig. 11. Values of the quality function $C_p$ for different temperature decreasing rate

### *3.5 Greedy Randomized Adaptive Search (GRASP)*

Greedy Randomized Adaptive Search (GRASP) is a constructive technique designed as a multi-start heuristic for combinatorial problems [11]. Although GRASP is not an evolutive technique, it has been elected as a metaheuristic method to be compared with LOT and the set of evolutive approaches. Several problems as circuit partitioning, set covering or graph planarization have been successfully addressed by GRASP algorithms [30]. GRASP implementations are very robust and it is unusual to find examples where this method performs badly.

GRASP implementation starts with a initial partition of the DVE. Unlike the evolutive techniques described above, the initial partition used by GRASP does not provide any assignment for the border avatars. In this case, GRASP exploits an important property of the DBA method presented in section 3.1. In

DBA algorithm the assignment of avatars to servers is calculated iteratively. The avatar with the lowest distance to a given server of the DVE (following a round-robin scheme) is assigned in each iteration. In order to measure these distances, servers are represented by mass-centers (see DBA algorithm). Therefore, as avatars are assigned they are clustered around DVE servers.

Because of DBA obtains a perfect balancing of avatars ($C_P^W$), system cost ($C_p$) depends only (see equation refCp) on the evaluation of inter-server communications ($C_P^L$). Thus, the best partitioning solutions are obtained when neighboring avatars are assigned to the same server. This property makes avatars which are located far away or equidistant of different mass-centers to be critical. These avatars are denoted as *critical avatars*, In order to properly assign critical avatars it is necessary to spend more resources. In order to avoid suchspending, a different version of DBA algorithm is used. This version, denoted as DBA-R, does not assign critical avatars. These avatars are the inputs to GRASP algorithm.

Figure 12 shows an example of a DBA execution. In this example 66 avatars are simulated with 3 servers in a DVE system. Each server received a balanced group of 18 elements each located in a rounded pattern. The remaining 12 avatars have been intentionally non-assigned.

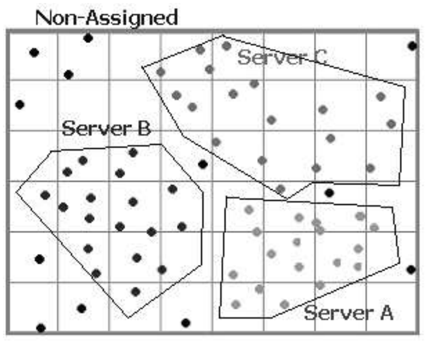

Fig. 12. Result from a initial solution obtained by DBA-R algorithm

The GRASP-based method for solving the partitioning problem in DVE systems consists of several iterations [26]. Each iteration assigns a critical avatar and consists of two steps: construction and local search. The construction phase builds a feasible solution choosing one border avatar by iteration, and the local search also provides a server assignment of that border avatar in the same iteration, following the next procedure: First, the resulting cost $C_p$ of adding each non-assigned critical avatar to the current (initial) partition is computed. Since each border avatar can be assigned to different servers, the cost for assigning each border avatar to each server is computed, forming the *list of candidates (LC)* (each element in this list has the form (*non-assigned border avatar, server, resulting cost*). This list is sorted (using Quick-sort algorithm) by the resulting cost $C_p$ in descendent order, and then is reduced to its top quartile. One element of this reduced list of candidates (RLC) is then randomly chosen (construction phase). Next, an extensive search is performed in the AOI of that selected avatar. That is, all the possible assignments of the avatars in the AOI of the selected avatars are computed, and the assignment with the lowest $C_p$ is kept.

Next code describes how the GRASP approach works when n avatars are assigned to S servers in a DVE system:

```
program GRASP (Int threshold)

type New_sol
    idx_av,idx_ser: Int
    new_cost      : Real
    avatar,server : Int

var
   tmp_cost,Cp_GRASP :Real
   Non_assig         :Integer
   list              :New_sol[]

begin
   Initial_Partition (DBA-R,threshold)
   non_assig := n - threshold
   For i:=0 to non_assig do
      for j:=0 to n do
        for k:=0 to S do
            tmp_cost=TestSolution(j,k)
            AddToList(list,tmp_cost,j,k)
        end_for
      end_for
      QuickSort(list)
      ReduceToFirsQuartile(list)
      ChooseRandomElement(list,avatar,server)
      Cp_GRASP = tmp_cost = 10000
      for j:=0 To AvatarsInAOI(avatar) do
          for k:=0 to S do
            tmp_cost=TestSolution(j,k)
            AddToList(list,tmp_cost,j,k)
            if (tmp_cost < Cp_GRASP)
               Cp_GRASP := tmp_cost
               savej := j
               savek := k
            end_for
          end_for
      end_for
      AssingSolutionServer(savej,savek)
  end_for
```

```
end
```

The quality of the solutions provided by GRASP search method depends on the quality of the elements in the RLC, and the range of solutions depends on the length of the RLC. Thus, the main parameter to be tuned in this case is the number of *non-assigned N* or *critical* avatars that the initial partition must leave.

Figure 13 shows the results in this tuning phase in order to compose an intermediate solution. In this example a LARGE world composed by 2500 avatars are assigned to 8 servers. The avatars are located following a uniform distribution. This figure represents the variations of two performance measures as the number of critical avatars (iterations) is increased. The quality of the obtained solutions $C_p$ and the execution time for GRASP algorithm have been elected as performance measures.

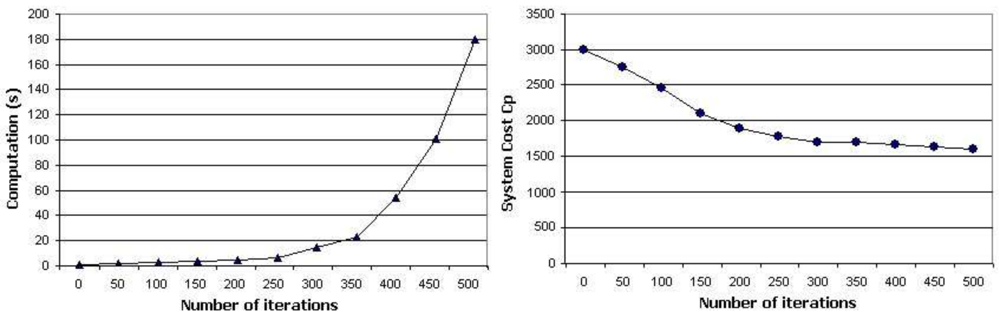

Fig. 13. Variation of the performance measures for different threshold values

Figure 13 shows that as the number of critical avatars increases the quality of the provided solutions also increases ($C_p$ values decreases), but the execution time for GRASP algorithm (labelled as computations) also increases. We chose in this case a compromise solution of 250 iterations. It is worth mention that a larger number of iterations result in higher execution times and do not reach significant solutions.

## 4 Performance Evaluation

In this section, we present the performance evaluation of the heuristics described in the previous section when they are used for solving the partitioning problem in DVE systems. Following the evaluation methodology shown in [23], we have empirically tested these heuristics in two examples of a DVE system: a small world, composed by 13 avatars and 3 servers, and a large world, composed by 2500 avatars and 8 servers. We have considered two parameters: the value of the quality function $C_p$ for the partition provided by the search method and also the computational cost, in terms of execution time, required by the search method in order to provide that partition. For comparison purposes, we have also implemented the *linear optimization technique* (LOT) [23]. This method currently provides the best results for the partitioning problem in DVE systems. In the case of small worlds we have also performed an exhaustive search through the solution space, obtaining the best partition as possible. The hardware platform used for the evaluation has been a 1.7 GHz Pentium IV with 256 Mbytes of RAM.

Since the performance of the heuristic search methods may heavily depend on the location of avatars in the virtual world, we have considered three different distributions of avatars: uniform, skewed, and clustered distribution. Figure 14 shows an example of how avatars would be located in a 2-D world when following each one of these distributions.

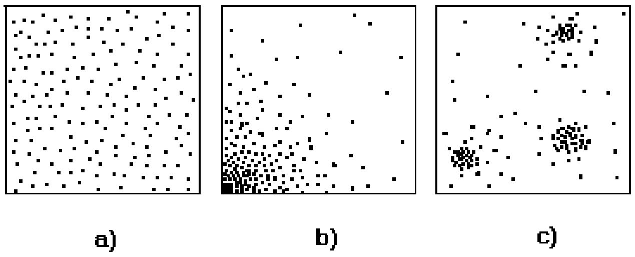

Fig. 14. Distributions of avatars: (a) uniform, (b) skewed, and (c) clustered

Table 2 shows the $C_p$ values corresponding to the final partitions provided by each heuristic search method for a small virtual world, as well as the execution times required for each method in order to obtain that final partition. It can be seen that all of the heuristics provide better (lower) $C_p$ values than the LOT search method for a uniform distribution of avatars. For the skewed and clustered distributions, most of the heuristics also provides better $C_p$ values than the LOT search method, and some of them (GA and SA methods) even provide the minimum value. However, the execution times required by most of the heuristics are longer than the ones required by the LOT method. Only GRASP method provides worse $C_p$ values than the LOT method, but it requires much shorter execution times. Although these results does not clearly show which heuristic provides the best performance, they validate any of the proposed heuristics as an alternative to the LOT search method.

However, in order to design a scalable DVE system the partitioning method must provide good performance when the number of avatars in the system increases. That is, it must provide a good performance specially for LARGE virtual worlds. Table 3 shows the required execution times and the $C_p$ values obtained by each heuristic search method for a large virtual world. In this case, all of the heuristics provides similar values of $C_p$ than the one provided by LOT method for the uniform distribution, while requiring much shorter execution times. When non-uniform distributions of avatars are considered, then all of the heuristics provide much better $C_p$ values than the LOT method and they also require much shorter execution times than the LOT method. In

| | **Uniform dist.** | | **Skewed dist.** | | **Clustered dist.** | |
|---|---|---|---|---|---|---|
| | *Time (s.)* | $C_p$ | *Time (s.)* | $C_p$ | *Time (s.)* | $C_p$ |
| **Exhaustive** | 3.411 | 6.54 | 3.843 | 7.04 | 4.783 | 7.91 |
| **LOT** | 0.0009 | 6.56 | 0.001 | 8.41 | 0.0011 | 8.89 |
| **SA** | 0.004 | 6.82 | 0.005 | 7.46 | 0.005 | 7.91 |
| **ACS** | 0.0007 | 6.59 | 0.003 | 7.61 | 0.0024 | 8.76 |
| **GA** | 0.002 | 6.54 | 0.003 | 7.04 | 0.005 | 7.91 |
| **GRASP** | 0.0002 | 7.42 | 0.0002 | 8.63 | 0.0003 | 11.88 |

Table 2
Results for a small DVE system

particular, ACS method provides the best $C_p$ values for non-uniform distributions, requiring also the shortest execution time in the case of a clustered distribution.

| | **Uniform dist.** | | **Skewed dist.** | | **Clustered dist.** | |
|---|---|---|---|---|---|---|
| | *Time (s.)* | $C_p$ | *Time (s.)* | $C_p$ | *Time (s.)* | $C_p$ |
| **LOT** | 30.939 | 1637.04 | 32.176 | 3460.52 | 43.314 | 5903.80 |
| **SA** | 6.35 | 1707.62 | 13.789 | 2628.46 | 29.62 | 4697.61 |
| **ACS** | 5.484 | 1674.08 | 14.05 | 2286.16 | 23.213 | 3736.69 |
| **GA** | 6.598 | 1832.21 | 14.593 | 2825.64 | 29.198 | 4905.93 |
| **GRASP** | 6.622 | 1879.76 | 13.535 | 2883.84 | 26.704 | 5306.24 |

Table 3
Results for a large DVE system

These results show that the performance of the partitioning algorithm can be significantly improved by simply using any of the proposed heuristics instead of the LOT method, thus increasing the scalability of DVE systems. In particular, ACS method provides the best performance as a partitioning algorithm for LARGE worlds.

## 5 Conclusions

In this work, we have proposed a comparison study of modern heuristics for solving the partitioning problem in DVE systems. This problem is the key issue that allows to design scalable and efficient DVE systems. We have evaluated the implementation of different heuristics, ranging over most of the current taxonomy of modern heuristics. We have tested the proposed heuristics when applied for both small and large DVE systems, with different distributions of the existing avatars in the system. We have compared these results with the ones provided by the Linear Optimization Technique (LOT), the partitioning method that currently provides the best solutions for DVE systems. For small virtual worlds, we can conclude that in general terms any of the implemented heuristics provides a partition with similar values of the quality function $C_p$, but the execution times required by the implemented heuristics are longer than the time required by the LOT search method. Although SA and GA methods provide the minimum value of the quality function, only GRASP method provides execution times shorter than the ones required by the LOT method for all the tested distributions of avatars. These results validate any of the proposed heuristics as an alternative to the LOT search method when considering small DVE systems. However, for large virtual worlds any of the proposed heuristics provides better $C_p$ values and requires shorter execution times than the LOT method for non-uniform distributions of avatars. In particular, ACS method provides the best results. Since a scalable DVE system must be able to manage large amounts of avatars, we can conclude that these results validates ACS search method as the best heuristic method for solving the partitioning problem in DVE systems.

## References

[1] M. Abrash, "Quake's game engine: The big picture" Dr. Dobb's.Journal.Spring. Year 1997.

[2] D.B. Anderson, J.W.Barrus, J.H.Howard, "Building multi-user interactive multimedia environments at MERL", in *IEEE Multimedia*, 2(4), pp.77-82, Winter 1995.

[3] P.A.Berstein, V.Hadzilacos and N.Goodman. "Concurrency, Control and Recovery in Database Systems" *Addison-Wesley.* 1997.

[4] C. Bouras, D. Fotakis, A. Philopoulos, "A Distributed Virtual Learning Centre in Cyberspace", Proc. of the 4th International Conference on Virtual Systems and Multimedia, VSMM98, Gifu-Japan, 18-20 November 1998

[5] C. Coello, G. Lamont, D. Van Veldhuizen "Evolutionary Algorithms for Solving Multi-Objective Problems",*Kluwer Academic Publishers*, 2002

[6] H. Delmaire, J.A. Daz, E. Fernndez, and M. Ortega, "Comparing new heuristics for the pure integer capacitated plant location problem", Technical Report DR97/10, Department of Statistics and Operations Research, Universitat Politecnica de Catalunya, Spain, 1997.

[7] M. dorigo, V. Maniezzo, A. Coloni, , "The Ant System: Optimization by a Colony of Operation Agent", in *IEEE Transactions on Systems, Man & Cybernetics*, Vol. 26, 1996.

[8] M.Dorigo, G. Di Caro (Editor), M. Sampels, "Ant Algorithms: Third International Workshop, Ants 2002" , , *Lecture Notes in Computer Science*, 2463, Springer-Verlag, 2002.

[9] R.Duda, P.Hart, D.Stork, "Pattern Classification", *Ed. Wiley Intescience, 2000*, pp. 567-580.

[10] J. Falby, M. Zyda, D. Pratt, and R. Mackey. "NPSNET: Hierarchical Data Structures for Real-Time Three-Dimensional Visual Simulation". *Computers & Graphics*, 17(1):6569, 1993.

[11] Thomas A. Feo, Mauricio G.C, "Greedy Randomized Adaptive Search Procedures", *Resende Journal of Global Optimization*, 1995.

[12] C. Greenhalgh, "Awareness-based communication management in the MASSIVE systems", *Distributed Systems Engineering*, vol. 5, no. 3, pp. 12937, 1998

[13] Randy L. Haupt, Sue Ellen Haupt, "Practical Genetic Algorithms", *Ed. Willey*, 1997.

[14] J.C.Hu, I.Pyarali, D.C.Schmidt, "Measuring the Impact of Event Dispatching and Concurrency Models on Web Server Performance Over High-Speed Networks", *Proc. of the 2nd. IEEE Global Internet Conference*, November.1997.

[15] Kali networked game support software: *http://www.kali.net*

[16] K.E. Kinnear, "Alternatives to Automatically Function Definition", *In Advances in Genetic Programming*, MIT Press, pp 119-141, 1994

[17] S. Kirkpatrick, C. D. Gelatt and M.P. Vecchi,"Optimization by Simulated Annealing", *Science*, Number 4598, 13 May 1983

[18] C. Koulamas, S.R. Antony, R.Jaen, "A Survey of Simulated Annealing Applications to Operations Research Problems ", *Omega, International Journal of Management Science* Vol. 22, 41-56. 1994

[19] D. Lee, M. Lim, and S. Han, "ATLAS - A Scalable Network Framework for Distributed Virtual Environments", *ACM Collaborative Virtual Environments(CVE2002)*, Bonn Germany, September-October, 2002, pp.47-54.

[20] Michael Lewis and Jeffrey Jacboson, "Game Engines in Scientific Research", in *Communications of the ACM*, Vol 45. No.1, January 2002.

[21] John C.S. Lui, M.F.Chan, Oldfield K.Y, "Dynamic Partitioning for a Distributed Virtual Environment", *Department of Computer Science*, Chinese University of Hong Kong, 1998.

[22] John C.S. Lui, W.K. Lam, "General Methodology in Analysing the Performance of Parallel/Distributed Simulation under General Computational Graphs", *Third International Conference on the numerical Solution of Markov Chain*, September 6-10, 1999.

[23] Jonh C.S. Lui, M.F. Chan, "An Efficient Partitioning Algorithm for Distributed Virtual Environment Systems", *IEEE Trans. Parallel and Distributed Systems*, Vol. 13, March 2002

[24] M. Macedonia, "A Taxonomy for Networked Virtual Environments", *IEEE Multimedia*, 4(1) 48-56. January-March 1997.

[25] D.C. Miller, J.A. Thorpe, "SIMNET: The advent of simulator networking", in *Proceedings of the IEEE*, 83(8), pp. 1114-1123. August, 1995.

[26] P. Morillo and M.Fernández, "A GRASP-based algorithm for solving DVE partitioning problem", *Proc. of 2003 Int. Parallel and Distributed Processing Symposium (IEEE-IPDPS' 2003).* Nice, France. April, 2003 13.

[27] P. Morillo, M. Fernández and J.M.Orduña, "An ACS-Based Partitioning Method for Distributed Virtual Environment Systems", *Proc. of 2003 Int. Parallel and Distributed Processing Symposium Workshops (IEEE-IPDPS: NIDISC' 2003).* Nice, France. April, 2003.

[28] P. Morillo, M. Fernández and J.M.Orduña , "Comparison Study of Modern Heuristics for Solving the Partitioning Problem in Distributed Virtual Environment Systems", *Proc. of Int. Conference on Computational Science and its applications (ICCSA' 2003). Lecture Notes in Computer Science 2669.* Montreal, Canada, May, 2003.

[29] P. Morillo, M. Fernández and N.Pelechano, "A grid representation for Distributed Virtual Environments", *Proc. of 2003 1st European Across Grids Conference.* Santiago de Compostela, Spain, February, 2003.

[30] M. Resende, C. Ribeiro, "Greedy Randomized Adaptive Search Procedures",*Handbook of Metaheuristics, Kluwer Academic Publishers*, pages 219-249. Dordrecht, The Netherlands, 2002.

[31] J.M. Salles Dias, Ricardo Galli, A. C. Almeida et al. "mWorld: A Multiuser 3D Virtual Environment", in *IEEE Computer Graphics*, Vol. 17, No. 2, March-April 1997.

[32] S. Singhal, and M.Zyda, "Networked Virtual Environments", *ACM Press, New York, 1999.*

[33] P.T. Tam, "Communication Cost Optimization and Analysis in Distributed Virtual Environment", *M. Phil second term paper, Technical report RM1026-TR98-0412.* Department of Computer Science & Engineering.The Chinese University of Hong Kong. 1998.