An Evolutive Approach to the Partitioning Problem in Distributed Virtual Environment Systems

P. Morillo, M. Fernández, J. M. Orduña

Resumen— **Fast Internet connections and the widespread use of high performance graphic cards are making Distributed Virtual Environment 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 DVE system. One of these key issues is the partitioning problem. This problem consists of efficiently assigning clients to the servers in the system.**

This paper proposes a new implementation of the Ant Colony System (ACS) search method for solving the partitioning problem in DVE systems. ACS method is a heuristic search method based on evolutionary computation. Performance evaluation results show that, due to its ability of both finding good search paths and escaping from local minima, the proposed method can provide better solutions with shorter execution times than the methods proposed in the literature, particularly for large DVE systems. Therefore, the proposed implementation of ACS search method improves scalability and efficiency of DVE systems.

I. Introduction

High-speed connections to Internet and the widespread use of high performance graphic cards have made possible the current growth of Distributed Virtual Environment (DVE) systems. Theses systems allow multiple users (working on different computers that are interconnected through different networks) to interact in a shared virtual world. This is achieved by rendering images of the environment as if they were perceived by the user. Each user is represented in the shared virtual environment by an entity called *avatar*, whose state is controlled by the user input. Since DVE systems support visual interactions between multiple avatars, every change in each avatar must be propagated to the rest of the avatars in the shared virtual environment.

DVE systems are currently used in different applications [19], such as collaborative design [18], civil and military distributed training [14], e-learning [15] or multi-player games [1], [11]. Due to their different requirements, communication rate of avatars may significantly differ among these different applications. On other hand, DVE systems are inherently heterogeneous. Effectively, each avatar can be controlled by a different computer system, with different resources. Additionally, both clients and servers can be interconnected through different networks using different technologies. Such heterogenous systems must deal with high communication and computational requirements as the number of avatars in the DVE system increases. Thus, since servers must render different 3D models, perform the positional updates and transfer control information among different clients, each new avatar represents an increasing in the computational requirements of the application. Additionally, it also represents an increasing in the amount of network traffic.

One of the key issues in the design of a scalable DVE system is the *partitioning problem* [12]. It consists of efficiently assigning the workload (avatars) among different servers in the system. Since the partitioning problem may seriously affect the issues described above, it can determine the overall performance of the DVE system. Some methods for solving the partitioning problems have been already proposed [12], [19]. These methods provide efficient solutions even for large scale DVE systems. However, there are still some features in the proposed methods that can be improved. For example, different heuristic search methods can be used for finding the best assignment of clients to servers.

In this paper, we propose a new heuristic search method for solving the partitioning problem in DVE systems. The proposed method is based on Ant Colony Systems (ACS), an evolutionary computation algorithm. This heuristic search method uses positive feedback to improve the use of good search paths, while using negative feedback to escape from local minima. Performance evaluation results show that, due to its ability of both finding good search paths and escaping from local minima, the proposed method can achieve better assignments while requiring shorter execution time than other methods proposed in the literature.

The rest of the paper is organized as follows: Section II describes the partitioning problem and the existing proposals for solving it. Section III shows the proposed method for solving the partitioning problem, based on ACS search, and also the new implementation proposed for this search method. Next, Section IV presents the performance evaluation of the proposed search method. Finally, Section V presents some concluding remarks and future work to be done.

II. The Partitioning Problem in DVE Systems

Architectures based on networked servers are becoming a de-facto standard for DVE systems [19],

Supported by the Spanish CICYT under Grant TIC2000–1151–C07–04

Departamento de Informática, Universidad de Valencia. e-mail: Juan.Orduna.@uv.es.

[13], [9]. In these architectures, the control of the simulation relies on several interconnected servers. Multi-platform client computers must be attached to one of these servers. 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. 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) [19], locales [2] or auras [10] have been proposed for limiting the number of neighboring avatars that a given avatar must communicate with.

Depending on their origin and destination avatars, messages in a DVE system can be intra-server or inter-server messages. 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.

Lui and Chan have shown the key role of finding a good assignment of clients to servers in order to ensure both a good frame rate and a minimum network traffic in DVE systems [12]. They propose a quality function, denoted as C_p, for evaluating each assignment of clients 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, 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 inter-server communication cost, denoted as C_p^L. In order to minimize this parameter, avatars sharing the same AOI should be assigned to the same server. Thus, the 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. Thus, if the servers in the DVE system are connected through low performance networks (i.e Internet) then the quotient W_1/W_2 must be close to zero. On the contrary, if the servers are interconnected using high performance networks, this quotient must be close to one. Using this quality function (and assuming $W_1 = W_2 = 0.5$), Lui and Chan propose a partitioning algorithm that periodically re-assigns clients to servers [12], in order to adapt the partition to the current state of the DVE system (avatars can join or leave the DVE system at any time, 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 [12].

Some other approaches for solving the partitioning problem have been also proposed. One of them [3] groups avatars following regular distributions. In order to ensure good performance, this algorithm generate a number of regular distributions equal to the number of servers in the DVE system. However, this proposal does not obtain good performance when avatars are located following a non-uniform distribution.

Another different approach rejects dynamic concepts associated to avatars like AOI, aura or locale [20]. This proposal divides the 3D virtual scene in a regular grid. A multicast group is created for each grid cell, in such a way that avatars sharing a cell also share multicast packets and are assigned to the same server. 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 areas of the clusters are overloaded, increasing the overall cost of the quality function.

Since the partitioning method proposed by Lui and Chan currently provides the best results for DVE systems, our proposal uses the same approach: using the same quality function, we will obtain an initial partition (assignment) of avatars to servers, and then we will modify this partition (using a new method), providing a near optimal assignment. As the state of the DVE system changes, the proposed method can be iteratively applied for properly adapting the current assignment.

III. A New Partitioning Method

In this section, we present a new method for solving the partitioning problem. Following the approach presented by Lui and Chan [12] (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 in the DVE system. Since the partitioning algorithm must provide solutions dynamically, the search method must be as flexible as possible. Therefore, we propose the use of Ant Colony Systems (ACS) [6], an evolutionary computation algorithm. This heuristic search method uses positive feedback to improve the use of good search paths, while using negative feedback to escape from local minima. These features makes ACS to be a very flexible search method.

As his name suggests, ACS is based on the behavior shown by ant colonies when searching possible paths to their food. They use a hormone called pheromone to communicate among them. The path 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 different problems [5], [17], [16]. Concretely, we propose a new implementation of the ACS search method, to be used for solving the partitioning problem in DVE systems. This implementation starts with an initial partition and assignment of avatars to the servers in the DVE system. This initial partition obtains good assignments only for several avatars, while the ACS algorithm itself performs successive refinements of the initial partition that lead to a near optimal partition. Also, ACS search method is used periodically to update 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).

We tested several clustering algorithms for obtaining the initial partition. Although they are not shown here for the sake of shortness, we obtained the best results for a *density-based algorithm* [7]. This algorithm divides the virtual 3D scene in square sections. Each section is labeled with the number of avatars that it contains (na), and all the sections are sorted (using Quicksort 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. 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. However, 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.

Nevertheless, the assignment of avatars equidistant or located far away from the mass-centers is critical for obtaining a partition with minimum inter-server communication requirements (and therefore 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. ACS method is used at this point to search a near optimal assignment that properly re-assigns these avatars.

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 solutions 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 (each ant adds pheromone to the search path 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

$$S_v = \alpha \times pheromone \; + \; \beta \times C_p \qquad (2)$$

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 α and β 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:

$$pheromone = pheromone + Q \times \frac{1}{C_p} \qquad (3)$$

We have performed empirical studies in order to obtain the best values for α, β and Q coefficients. Although the results are not shown here for the sake of shortness, we have obtained the best behavior of the ACS method for $\alpha = 1.0$, $\beta = 7.0$ and $Q = 1000$. Surprisingly, the values obtained for α and β are exactly the same values that the ones obtained in the implementation proposed for the TSP problem [17]. Nevertheless, the best value for the Q coefficient heavily depends on the way the quality function is computed. Thus, other implementations use 100 as the best value for the Q coefficient [5].

Additionally, there are other key parameters in ACS search method that must be properly tuned. In particular, we have tuned the values for the number of ants N, the pheromone evaporation rate and the number of iterations that ACS method must perform to obtain a near optimal partition. These parameters have been tuned for both small and large virtual worlds. Concretely, we performed this tuning in a small virtual world composed of 13 avatars and 3 servers, and also in a large world composed of 2000 avatars and 8 servers. However, for the sake of shortness we present here only the results obtained for large worlds (the purpose of solving the partitioning problem is to provide scalable DVE systems. Therefore, ACS method must efficiently work in large virtual worlds). Since the performance of the method may heavily depend on the location of the avatars, we have considered three different distributions of avatars: uniform, skewed, and clustered distribution. Figure 1 shows the assignment of avatars to servers that the density-based algorithm provides for each of the considered distributions. In this figure, avatars are coded as grey scale points. The location of the avatars in a 2D virtual world represents the spatial location of the avatar in the virtual world simulated by the DVE system. The server each avatar is assigned to by the density-based algorithm is coded with a certain grey level. The assignment shown in this Figure represents the initial partition of the ACS method for the considered distributions

Figure 2 shows the values of the quality function C_p (denoted as *system cost*) 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.

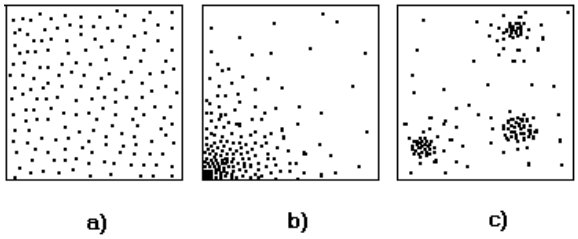

Fig. 1. Example of virtual world wherein avatars are located in a (a) uniform, (b) skewed, and c) clustered distribution

Fig. 2. Values of the quality function C_p for different evaporation rates of pheromone.

Figure 3 shows the values of C_p reached by the ACS method when different number of iterations are considered. It shows that C_p decreases when the number of iterations increases, until value of 25 iterations is reached. 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 selected for ACS method has been 25 iterations.

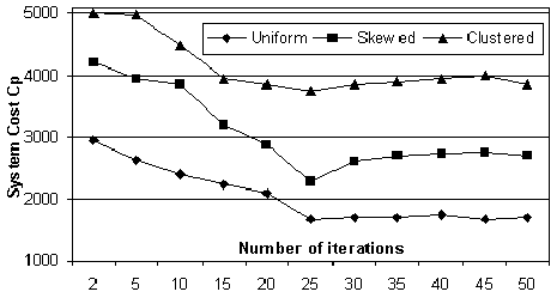

Fig. 3. Values of the quality function C_p for different number of iterations.

Finally, Figure 4 shows the values of C_p reached by

ACS method when different number of ants are considered. Again, for all the considered distributions C_p decreases when the number of ants increases, until a value of 100 ants is reached. From that point, system cost C_p remain constant. Again, the existing level of pheromone makes useless to add more ants to the population, since they are unable to find better paths.

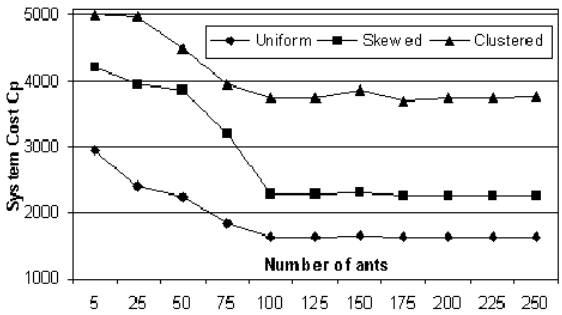

Fig. 4. Values of the quality function C_p for different number of ants.

Therefore, the proposed implementation of the ACS search method consists of the following steps (expressed as pseudo-code statements):

```
program ACS (Int Ants, N, 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 (DENSITY_BASED)
   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
```

IV. Performance Evaluation

In this section, we present the performance evaluation of the ACS method when it is used for solving the partitioning problem in DVE systems. We have empirically tested the ACS search method 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 performed the ACS search method as well as the *linear optimization technique* (LOT) [12]. This method currently provides the best results for the partitioning problem in DVE systems. Additionally, when considering small worlds we have also performed an exhaustive search (ES) through the solution space, obtaining the best partition as possible for the case of the small world. However, the computational cost required by an exhaustive search was too high for the case of the large world.

Table I shows the results obtained for the case of a small world. For each method, it shows the execution time T (in seconds) amd the value of C_p provided by that method. It can be seen that both LOT and ACS methods require only a small fraction of the execution time required for an exhaustive search. For a uniform distribution of avatars, the ACS method provides a partition with a higher value of C_p, although the execution time required by LOT method is longer than the execution time required for ACS method. For irregular distributions of avatars, the values of C_p provided by the ACS method are lower than the ones provided by the LOT method, while the required execution time is longer. Thus, for small worlds ACS search method does not provide a significant performance improvement.

	Uniform		**Skewed**		**Clustered**	
	T (s.)	C_p	*T (s.)*	C_p	*T (s.)*	C_p
ES	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
ACS	0.0007	6.59	0.003	7.61	0.0024	8.76

TABLA I
Results for a small DVE system

Nevertheless, the main purpose of the proposed method is to improve the scalability of DVE systems. Therefore, it must provide a significant performance improvement when it is used in large DVE systems. Table II shows the results obtained for the case of a large world. In this case the value of C_p provided by the ACS method is slightly higher than the one provided by the LOT method when avatars are uniformly distributed, while the required execution time is approximately one sixth of the time required for the LOT method. When avatars are located following a skewed distribution then both the required execution time and the value of C_p provided by the ACS method are much lower than the ones provided by the LOT method. Thus, the LOT method provides a value of C_p a 58% higher than the one provided by the ACS method, and requires an execution time approximately a 100% longer than the one required for the ACS method. When avatars are located following a clustered distribution, then the value of C_p provided by the ACS method is only a 36.7% of the value provided by the LOT method, while the execution time required by the ACS method is a 46.4%

of the one required by the LOT method.

	Uniform		Skewed		Clustered	
	T (s.)	C_p	T (s.)	C_p	T (s.)	C_p
LOT	30.939	1637	32.176	3461	43.314	5904
ACS	5.484	1674	14.05	2286	23.213	3737

TABLA II
RESULTS FOR A LARGE DVE SYSTEM

The reason for this behavior is that the inherent flexibility of ACS method allows it to dynamically find good paths in changing environments. In DVE systems avatars can constantly join or leave the system, and they can also move at any time. This changing environment is where ACS method can fully exploit its flexibility.

V. CONCLUSIONS

In this paper, we have proposed a new implementation of the Ant Colony System search method 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 proposed implementation 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, the proposed implementation of the ACS method is able to find solutions slightly better than the ones provided by the LOT method for non-uniform distributions of avatars. However, ACS method requires longer execution times for these distributions of avatars. Inversely, ACS method provides a worse solution (a higher value of the quality function C_p associated to the provided partition) for a uniform distribution of avatars, but requires a shorter execution time. These results validates ACS method as an alternative to the LOT method for solving the partitioning problem.

However, in order to design a scalable DVE system the partitioning method must provide a good performance when the number of avatars increases. Therefore, the main purpose of the proposed implementation of the ACS method is to provide an efficient partitioning method for large virtual worlds. The results obtained for large virtual worlds show that if the distribution of avatars is non-uniform then the proposed implementation is able to find much better solutions for the partitioning problem than the LOT method, while requiring much shorter execution time. If the distribution of avatars is uniform, then the solutions provided by both methods are similar. However, the execution time required by the ACS method is much shorter than the one required by the LOT method. These results show that the use of ACS method can significantly improve the performance of the partitioning method for large virtual worlds

The reason for the performance improvement achieved with ACS method is the inherent flexibility of ACS method. This flexibility allows it to dynamically find good paths in changing environments. Since avatars can join or leave a DVE system at any time, and they can also freely move throughout the virtual world, DVE systems are extremely changing environments. Therefore, they can fully exploit the flexibility of ACS method for solving the partitioning problem.

REFERENCIAS

[1] M. Abrash, "Quakeś game engine: The big picture", in *Dr. Dobb's Journal*, Spring 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. Barham, T.Paul, "Exploiting Reality with Multicast Groups", in *IEEE Computer Graphics & Applications*, pp.38-45, September 1995.

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

[5] M. Dorigo and L. Gambardella, "Ant colony system: A Cooperative Learning Approach to the Traveling Salesman Problem", in *IEEE Transactions on Evolutionary Computation*, 1997.

[6] 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.

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

[8] T.A. Funkhouser, "Network Topologies for Scalable Multi-User Virtual Environments", *Technical Report Bell Laboratories*, 1996.

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

[10] 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", in *Proceedings of the 2nd. IEEE Global Internet Conference*, November.1997.

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

[12] Jonh C.S. Lui, M.F. Chan, "An Efficient Partitioning Algorithm for Distributed Virtual Environment Systems", in *IEEE Transactions on Parallel and Distributed Systems*, Vol. 13, No. 3, March 2002

[13] Michael R. Macedonia, "A Taxonomy for Networked Virtual Environments", in *IEEE Multimedia*, Vol. 4 No. 1, pp 48-56. January-March, 1997.

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

[15] Tohei Nitta, Kazuhiro Fujita, Sachio Cono, "An Application Of Distributed Virtual Environment To Foreign Language", in *IEEE Education Society*, October 2000.

[16] M. Randall, A. Lewis "A Parallel Implementation of Ant Colony Optimization", in *Journal of Parallel and Distributed Computing* No. 62, pp. 1421-1432, 2002.

[17] M. Riff, C. Grandon, R. Torres, "Evaluación de algoritmos h´íbridos basados en colonias de hormigas para TSP", Universidad de Colombia, 2002.

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

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

[20] 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.