A Genetic Approach for Distributing Semantic Databases of Crowd Simulations

M. Lozano, J. M. Orduña, V. Cavero

Universidad de Valencia. Departamento de Informática
Av. Vicent Andrés Estellés, s/n. 46100 Burjassot (Valencia), SPAIN
{Miguel.Lozano, Juan.Orduna, Vicente.Cavero}@uv.es

Abstract

Last years have witnessed how crowd simulations have become an essential tool for many virtual environment applications. These applications require both rendering visually plausible images and managing the behavior of autonomous agents, and therefore they need a scalable design that allow them to simultaneously tackle these two requirements. One of the main problems in the design of a scalable crowd simulations consists of efficiently distributing among different computers the semantic database containing the virtual world.

In this paper, we propose a genetic approach for distributing the semantic database of crowd simulations in such a way that the dependencies among the computers hosting the pieces of the database are minimized. The proposed approach avoids the saturation of these computers by ensuring that the size of the pieces assigned to each computer is properly balanced. The performance evaluation results show that the proposed approach significantly reduces the resulting overhead in regard to other local search methods, regardless of the movement pattern of the agents. Therefore, it allows an effective partition of the semantic database.

[1]

1. Introduction

Last years have witnessed how crowd simulations have become an essential tool for many virtual environment applications. Extensive use of virtual crowds has been made in many commercial movies. Also, high quality crowd simulations are crucial for many virtual environment applications in education, training, and entertainment [2, 9, 23].

[1]This research was funded by the Spanish MEC under grants CONSOLIDER-INGENIO CSD2006-00046 and TIN2006-15516-C04-04

Crowd simulations can be considered as virtual environment applications with were users do not control the avatars. Instead, avatars are autonomous agents that can have different missions in the virtual world. As a results, crowd simulations have two different goals. On the one hand, crowd simulations must focus on rendering visually plausible images of the environment, requiring a high computational cost. On the other hand, complex agents must have autonomous behaviors, greatly increasing the computational cost as well. Figure 1 shows an example of a crowd simulations. Concretely, it shows a detailed view of a urban environment. In this case, this environment is filled with 8000 autonomous agents that freely move an interact in the environment.

One of the main problems that arises in the design of a crowd simulation is the scalability. Some proposals tackle crowd simulations as a particle system with different levels of details (eg:*impostors*) in order to reduce the computational cost [3, 24]. Although these proposals can handle crowd dynamics and display populated interactive scenes (10000 virtual humans), they are not able to produce complex autonomous behaviors for their actors. On the contrary, several proposals have been made to provide efficient and autonomous behaviors to crowd simulations [21, 19, 4, 20, 15, 8]. However, they are based on a centralized system architecture, and they can only control a few hundreds of autonomous agents with different skills (pedestrians with navigation and/or social behaviors for urban/evacuation contexts). Taking into account that pedestrians represent the slowest human actors (in front of other kind of actors like drivers in cars, for example) these results show that scalability has still to be solved in crowd simulations.

Distributed schemes like networked-server architectures [10, 6, 18] or peer-to-peer architectures [16, 14, 17] have been proved to improve the scalability of virtual environments through the use of different interconnected computers. In order to exploit the potential of such computer architectures, the software architecture must also be distributed.

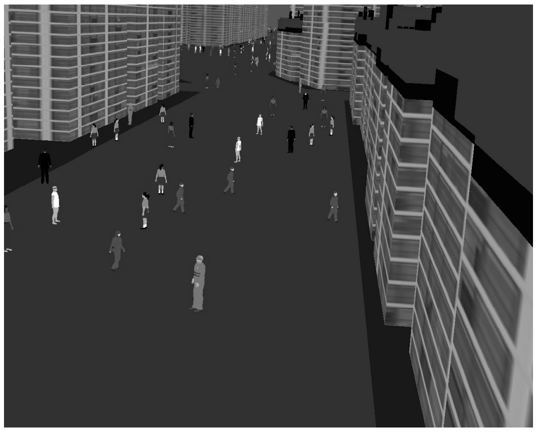

Figure 1. Detailed view of a urban environment with 8000 agents

An example of such software architecture could be the one shown in Figure 2. It is composed of two kind of elements: the *Action Server (AS)* and the *Client Processes (CP)*. The AS is unique, but the system can have as many client processes as necessary, in order to properly scale with the number of agents to be simulated. In his turn, each CP manages a group of autonomous agents. In order to take advantage from the underlying computer architecture, the most suitable distribution for this software architecture consists of allocating the AS in a single server, and uniformly distributing the CPs among the rest of the networked servers. Since each client process can manage a variable number of autonomous agents, the replicas usually host one CP, although it can hosts several ones. On the other hand, the action server is composed of two different modules, the Semantic Data Base (SDB) and the *Action Execution Module (AEM)*. The action server is hosted on another networked server.

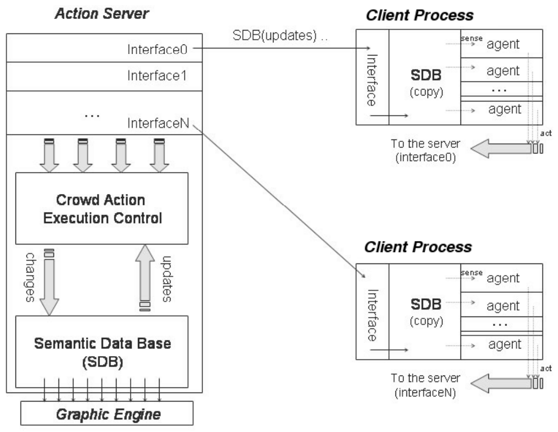

Figure 2. A distributed software architecture

Although a software architecture like the one shown in Figure 2 can improve the scalability of the crowd simulation with respect to centralized architectures, it is limited by the the centralized nature of the AS. This element becomes the system bottleneck when the number of agents reaches order of magnitude of tens of thousands. In this sense, a region-based partition of the semantic database and its distribution among different computers seems to be essential for actually exploiting the inherent scalability of distributed computer architectures. In such a scheme, each part of the distributed database would contain the information of a different region of the virtual world and it would reside in a different computer.

However, several problems arise when physically distributing the semantic database. First, in order to maintain the database consistency those agents near the borders of each region need to access to several regions. This requires the exchanging of locking requests among the computers hosting the partition of the database. Additionally, in order to avoid the saturation of those computers hosting the database the partition must be properly balanced. Taking into account these issues, in this paper, we propose a genetic approach for solving the problem of properly distributing the semantic database of crowd simulations among the computers in the system. Performance evaluation results show that the proposed algorithm improves the performance achieved by other local search methods, providing balanced partitions that actually minimize the number of locking requests.

The rest of the paper is organized as follows: Section 2 details the problems to be solved in order to properly distributing the semantic database and the evaluation function used for modeling the problem. Section 3 describes the implementation and tuning of the proposed genetic algorithm. Next, Section 4 presents the performance evaluation of the proposed method. Finally, Section 5 presents some concluding remarks.

2. The Semantic Database Problem

The problem of properly distributing the information stored in a centralized semantic database of a crowd simulation has not been still addressed. As discussed above, this problem can be split into two different problems: on the one hand, the overhead produced by those agents near the borders of each region. Effectively, when the semantic database is split into several pieces, each one containing a given region of the virtual world, the surroundings of an agent located near the border of a region are contained in more than one of the pieces. All of these pieces must wait (must be locked) until the action requested by that agent is checked in all the pieces, in order to ensure that other local requests do not change that area of the virtual world.

Regardless of the consistency protocol used, this constraint adds a significant overhead that must be minimized, since the requests from all the agents must be solved by the AS in a single cycle. On the other hand, the potential saturation of a given computer must be taken into account. In this sense, a proper balancing of the regions assigned to different computers must be made. Otherwise, one or more computers can reach saturation, greatly degrading the performance of the entire system [18].

An assignment of regions to computers consists of partitioning the virtual world into 2D or 3D regions and assigning each part of the semantic database (containing each of these regions) to a given computer. We will denote as *modules* each part of the database containing a given region. The computer k hosting a given module then will become the server for that region, meaning that all the agents located in that region of the virtual world (as shown in Figure 2, the autonomous agents are threads of a process being executed on another computer) will send requests to computer k to check their interactions with the virtual world. The semantic database problem will consists of finding a near optimal partition of the semantic database that minimizes the number of agents near the border of the regions and also that properly balances the number of agents in each of the regions. Additionally, the partition of the semantic database is a dynamic procedure. In each server cycle, the autonomous agents in a crowd simulation can move and interact with the virtual world existing in their surroundings, in such a way that when the simulation proceeds the current partition can become obsolete because many agents have moved from their original location. At this point, a new search for the best partition (taking into account the current position of the agents) must be performed.

Since this seems a complex problem, we propose the use of a bio-inspired heuristic technique that is capable to perform a heuristic search within the huge solution space of all possible partitions of the virtual world. The first step in the process of a heuristic search is the definition of a fitness function that properly models the problem to be solved. For the sake of clearness, in this paper we will consider 2-D worlds, although the technique can be easily extrapolated to 3-D worlds. We have considered rectangular (or square) regions delimited by four coordinates, *x_min, x_max, y_min* and *y_max*. The surroundings of each avatar are delimited by its Area of interest (AOI) [22]. The AOI defines the area of the virtual world that the agent can interact with. Usually, AOIs have spheric (3-D worlds) or circular shapes.

Since this particular problem is determined by the two main issues explained above, we have defined the fitness function to be minimized by the search method as the following one:

$$H(P) = \omega_1 \cdot \alpha(P) + \omega_2 \cdot \beta(P), \quad \omega_1 + \omega_2 = 1 \qquad (1)$$

The first term in this equation measures the number of agents in the resulting partition P whose surroundings crosses the region boundaries. Each interaction of such agents will require the locking of more than one of the modules, and this overhead must be minimized. Concretely, $\alpha(P)$ is computed as the sum of all the agents whose AOIs intersect two or more regions of the virtual world (that is, the number of avatars whose AOIs reside in more than one module). $\beta(P)$ is computed as the standard deviation of the average number of agents that each region contains. Therefore, $\beta(P)$ measures how balanced partition P is. Finally, ω_1 and ω_2 are weighting factors between 0 and 1 that can be tuned to change the behavior of the search as needed.

Thus, the proposed approach will consist of a heuristic search on order to find the partition that minimizes $H(P)$ as much as possible.

3. A Sexual Elitist Genetic Algorithm

Genetic Algorithms (GA) consists of a search method based on the concept of evolution by natural selection [13, 7]. GA starts from an initial population, made of R *chromosomes*, that evolves following certain rules, until reaching a convergence condition that maximizes a fitness function. Each iteration of the algorithm consists of generating a new population from the existing one by recombining or even mutating chromosomes. In this case, a chromosome consists of an integer array that contains k *Minimum Bound Rectangles (MBR)*, where k is the number of pieces which the semantic database must be split into. Each MBR is a quadruple (integer coordinates array) [*x_min, x_max, y_min, y_max*] that defines a rectangular region of the virtual world. Thus, a chromosome defines a partition of the virtual world in k regions.

Most of heuristic methods are based on the random generation of an initial population. However, if the initial population has been correctly defined, then the heuristic method easily obtains a good approximation to the global optimum. In this case the algorithm should maintain a certain level of structural diversity among all the chromosomes, in order to avoid the premature convergence of the search [13]. In this sense, it must be noticed that the method must be executed in each server cycle, and it must start from the result provided for the prior AS cycle. Thus, from this initial partition a population of R chromosomes is randomly generated. These chromosomes are sorted by the fitness function $H(P)$ associated with each chromosome in ascending order (the first chromosome is the one with the best partition). We have denoted this sorted list as *Best Solutions List (BSL)*. This list represents the initial population for the genetic algorithm, and it will contain the best R solutions found until that iteration by the GA. The value of R is a parameter that must be tuned. However, in order to prove that the proposed method can provide good results even in the worst case, we have not tuned this parameter, and we have used an arbitrary value (R=10).

Each iteration consists of generating a *descendant* generation of R chromosomes, starting from an *ancestor* generation. The way that the algorithm provides the next generation determines the behavior of the GA. We have chosen a sexual reproduction technique [13], in such a way that each descendant is generated starting from two ancestors. In each iteration the first ancestor for the $i-th$ chromosome of the population is the $i-th$ chromosome of the population in the previous iteration. The second ancestor is randomly selected among the 50% of the previous population with the best fitness function.

From each two ancestors, an offspring is obtained by computing a randomly skewed average of the corresponding coordinates in each of the ancestors. This skewed average is computed for all the coordinates in an MBR and for all the MBRs in a chromosome. As an example, Figure 3 shows the MBRs corresponding to two ancestors and an example of the resulting offspring. In this Figure, we can see an ancestor MBR a whose coordinates are defined by the quadruple [*xa_min, xa_max, ya_min, ya_max*] and a second ancestor MBR whose coordinates are defined by the quadruple [*xb_min, xb_max, yb_min, yb_max*]. From these two MBRs, the MBR with dashed lines is computed.

It must be noticed that this reproduction method can produce non-valid offsprings, because they define MBRs with different shapes. If this situation occurs, then the nonvalid offspring is discarded and another offspring is computed. For example, the resulting MBR in Figure 3 is narrower than ancestor a. If the rest of MBRs in the resulting chromosome do not include the area of a not covered by the resulting offspring, then the agents located at that area will not be assigned to the semantic database. Moreover, the initial partition, provided by the previous execution of the GA, can be corrupted due to the movement of agents during the AS cycle. In this case, the initial population starts from a modified partition where some regions are expanded as necessary to cover all the agents at the current locations.

Since invalid offsprings can be generated, the number of chromosomes in the offspring population can be lower than R. When an invalid offspring is generated it is simply discarded, and another offspring is generated from different ancestors in the population. When all the ancestor population has been used for producing offsprings and the number of valid offsprings is lower than R, then the process starts again until R valid offsprings are obtained. At this point, the R offsprings and the chromosomes in the BSL are sorted and merged to obtain the new BSL for that iteration. Mutation is not used, since exchanging two or more coordinates between different MBRs could lead to invalid chromosomes.

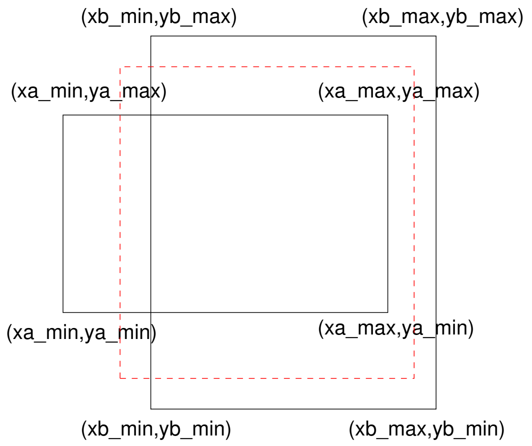

Figure 3. Offspring generation

The whole process performed in each iteration i can be expressed as the following pseudo-code statements :

```
Iteration i

CONST
   R  /* Num. of chromosomes in the population */

TYPE
  MBR : array[4] of integer;
  chromosome : array [k] of MBR;


VAR
   int j, l
   Anc1, Anc2 : chromosome    /* Ancestors */
   offs : chromosome          /* Offspring */

  BSL : array [R] of chromosome;
  offs_array : array [R] of chromosome;

begin
    z :=1;
    while z < R do
       For j:=1 to R do
          Anc1 := BSL(j);
          Anc2 := First_Half_Select (BSL);
          offs := skew_average (Anc1,Anc2);
          if (valid (offspring) )
                insert(offs_array, offs);
                z++;
          end_if
       end_for
     end_while
     Evaluate_And_Sort (offs_array, BSL);
end
```

Since one of the main constraints in the semantic database problem is the execution time of the search (it must be shorter than a fraction of the AS cycle, denoted as T, in order to provide an effective partition) we have established the execution time as one of the finishing conditions of the algorithm. Concretely, we have set T to half of the AS cycle period, that is, 125 msec.. Additionally, in order to ensure that the proposed method provides the best solution as possible, we have added the decrease of $H(P)$ as a convergence condition. That is, if the $H(P)$ value of the first chromosome in the BSL is not decreased in two successive iterations, then the algorithm finishes. Therefore, the first chromosome in the BSL is chosen as the result of the search either when the convergence condition is reached or when the algorithm has been executed during T milliseconds.

4. Performance Evaluation

In this section, we present the performance evaluation of the heuristic method described in the previous section. In order to show the genetic approach can improve better performance than other approaches, we have also tested another local search method, *K-means* [11]. K-means is a non-supervised clustering technique used in data mining and other AI fields. In this problem, K (the number of pieces which the semantic database must be split into) is fixed a priori. K-means splits the data set into k-partitions, considering that the distance from each agent to its centroid (the point whose x coordinate is the average x coordinate of all the agents and whose y coordinate is the average y coordinate of all the agents) should be minimized. It is an iterative method, and it continually processes the data set until the coordinates of all the centroids do not change in two iterations.

We have evaluated both the K-means method and the proposed method in a crowd simulation composed by 8000 autonomous agents and with K=5. Concretely, the crowd simulations consist of the evacuation of a structured 2-D world where there are several emergency exits. The autonomous agents must try to escape from the world as soon as possible. We have performed different experiments with different populations and different virtual worlds. In all of them we have considered three well-known movement patterns: Changing Circular Pattern (CCP) [1], HP-All (HPA) [5] and HP-Near (HPN) [12]. CCP considers that all avatars in the virtual world move randomly around the virtual scene following circular trajectories. HPA considers that there exists certain “hot points” where all avatars approach sooner or later. This movement pattern is typical of multiuser games, and this application is an example of this pattern, since all agents must find one of the existing exits. Finally, HPN also considers these hot-points, but only avatars located within a given radius of the hot-points approach these locations. In order to achieve these movement patterns, we have considered the following 2-D world configurations: full, where there are a lot of emergency exits uniformly distributed along the whole 2-D world (CCP pattern); perimeter, where all the emergency exits are uniformly distributed along the four borders of the virtual world (HP-Near); up, where there are only a few exits and they are located at the upper border of the world (HP-All); and down, where there is only one exit located at the lower border of the world (HP-All with a single hot-point). In order to illustrate these configurations, Figure 4 shows four snapshots of the virtual world with these configurations at half of the simulation time. In this figure, the 2-D world is viewed from above, and agents are represented as grey dots.

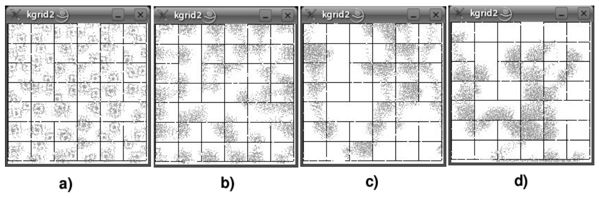

Figure 4. Movement patterns: a) full b) perimeter c) up d) down

A common feature for all these movement patterns is that as simulation proceeds and some agents accomplish their mission (escaping from the virtual world), the number of agents in the database decreases. Another common feature is that the obstacles in the virtual world (there are a lot of walls and only some doors allow to access the correct path for escaping) make agents to crowd in some regions of the virtual world.

The first step is to study the improvement achieved by the proposed approach (in terms of both the fitness function and execution times) with respect to the reference method (K-means). Table 1 shows an example of the performance evaluation results for both methods under the four configurations described above. The results obtained in all the experiments performed with different populations and in different virtual worlds were very similar. Each value in the tables of this section are computed as the average of at least 10 different simulations of the same configuration. Each row in this table shows the execution time in milliseconds required to compute each algorithm for a population of 8000 agents and the resulting $H(P)$ values achieved by each method. We have labeled the proposed method as *genMBR*, for genetic MBR. For both methods ω_1 and ω_2 values were set to 0.6 and 0.4, respectively.

Table 1 shows that for all the configurations the fitness function values provided by the proposed approach are lower than the ones provided by the K-means method. On the contrary, the execution times required by the proposed approach is one order of magnitude higher than the time required for executing the K-means algorithm, as it could be expected. Nevertheless, the only time constraint imposed by the application is that the partition must be provided before half of the AS period, that is, 125 milliseconds. Since none of the execution times reaches that threshold, these results show that the genetic approach can efficiently solve

Full		
Method	Ex. Time	H(P)
K-means	3	411
genMBR	98	296
Perimeter		
Method	Ex. Time	H(P)
K-means	5	487
genMBR	91	342
Up		
Method	Ex. Time	H(P)
K-means	7	923
genMBR	124	677
Down		
Method	Ex. Time	H(P)
K-means	7	866
genMBR	115	750

Table 1. Comparison study for different configurations.

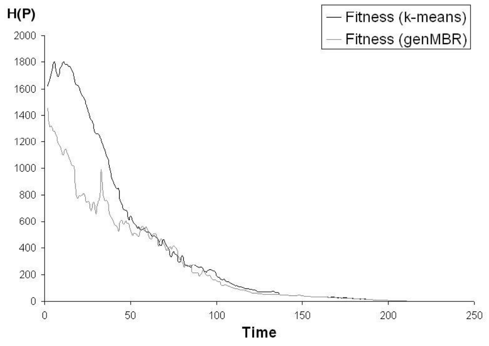

Figure 5. H(P) values provided during a crowd simulation (Full configuration)

Method	Locks	Std. Deviation
K-means	343	513
genMBR	154	510

Table 2. Performance evaluation for the Full configuration.

the semantic database problem.

Nevertheless, the results shown in table 1 do not show the actual improvement that the proposed approach provides to real systems. In order to achieve this goal, we have executed crowd simulations with the four movement patterns shown above and we have used both methods (with ω_1=0.6) for distributing the semantic database. Figure 5 shows the $H(P)$ values provided by both methods during the crowd simulation of the Full configuration. In this figure (and the following ones), the x-axis show the simulation time in seconds, while the y-axis shows the $H(P)$ values provided by both methods for all the AS cycles. It can be clearly seen that during the first 50 seconds the $H(P)$ values provided by the proposed approach are clearly lower than the ones provided by the K-means method. After that period the differences decrease due to the population decrease (in this configuration there are a lot of emergency exits and the population rapidly decreases).

In order to measure the effects that the proposed approach has on the simulations shown in Figure 5, Table 2 shows the average number of locking requests produced in each AS period among the computers hosting the database modules. Each value shown in this table is the average value for all the AS cycles of the simulation time. Also, this table shows the average standard deviation for the average number of agents in each module (that is, how balanced the provided partitions are).

Table 2 shows that the proposed method provides significantly better results than the reference method in terms of the number of locking requests. Concretely, the proposed method reduces the average number of locking requests to less than half of the ones provided by the K-means method. Also, table 2 shows the average standard deviation of the number of agents hosted in each module (that is, how balanced the partitions are during the simulations). This column of the table shows that the proposed method provides a similar balancing to the one provided by the K-means method.

Figure 6 shows the $H(P)$ values provided by both methods during a simulation with the Perimeter configuration. In this case the genMBR plot show much lower values than the K-means plot for the first 110 seconds of the simulation. After that period, the difference between the two plots is reduced and it remains more or less constant until the end of the simulation. However, the genMBR plot is always below the K-means plot, and thus this simulation longs a shorter time than the simulation with the K-means method. Since less locking requests are sent, the hosting computers can answer the agents faster, and therefore agents can accomplish their mission (evacuation) in a shorter time.

Table 3 shows the average number of locking requests produced in each AS period and the average standard deviation for the simulation performed using a perimeter configuration of the virtual world.

Table 3 shows that the proposed method is able to provide a lower number of locks than the K-means method, while the average standard deviation is still lower than the one provided by the K-means method.

Figure 7 shows the $H(P)$ values provided by both methods during a simulation with the Up configuration. In

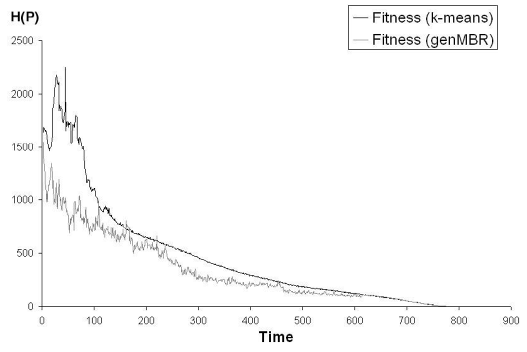

Figure 6. H(P) values provided during a crowd simulation (Perimeter configuration)

Method	Locks	Std. Deviation
K-means	189	936
genMBR	131	661

Table 3. Performance evaluation for the Perimeter configuration.

this case, the values of $H(P)$ provided by the proposed method are around half of the ones provided by the K-means method during the 300 first seconds of the simulation. After that period both plots shows more or less the same behavior. Again, these results show that the proposed method provides better results in terms of the fitness function.

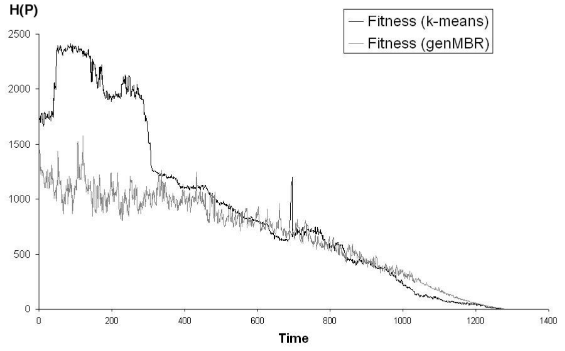

Figure 7. H(P) values provided during a crowd simulation (Up configuration)

Table 4 shows the average number of locking requests produced in each AS period and the average standard deviation for the simulation performed using an Up configuration of the virtual world.

Method	Locks	Std. Deviation
K-means	475	1597
genMBR	232	1346

Table 4. Performance evaluation for the Up configuration.

Table 4 shows that again the proposed method provides around half of the locking requests provided by the K-means method, while the partitions are slightly better balanced.

Finally, Figure 8 and table 5 show the performance results for the Down configuration. In this case the proposed method does not provide clearly lower values of $H(P)$ than the ones provided by the K-means method. In terms of actual performance improvement, the average number of locks shown in table 5 for the proposed method is less than half of the ones provided by the K-means method. The partitions provided by the proposed method show a similar standtard deviation than the ones provided by the K-means method. In order to ensure that the provided partitions do not lead to saturation, we have measured the maximum CPU utilization reached by all the computers in the simulation, and in all of them this value was lower than 85% (in order to reach saturation, a CPU utilization around 99% or more must be reached [18]). Therefore, the proposed method provide balanced partitions that generate less locking requests, actually improving the system throughput (the number of agents supported by the system).

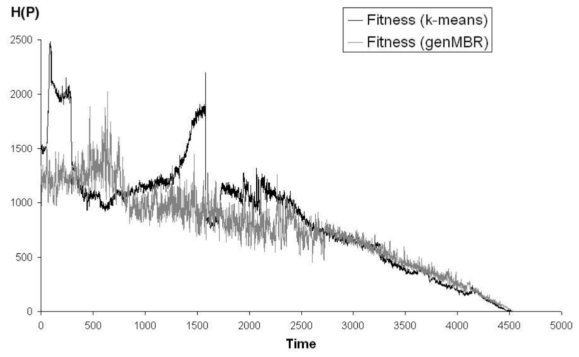

Figure 8. H(P) values provided during a crowd simulation (Down configuration)

When comparing table 5 with the rest of the tables we can see that the numbers in table 5 are the highest ones. This indicates that this is the configuration generating the highest workload, since there is only one emergency exit in

Method	Locks	Std. Deviation
K-means	684	1142
genMBR	299	1180

Table 5. Performance evaluation for the Down configuration.

the virtual world. Even in this case the proposed approach is able to generate a lower overhead due to the locking requests, while still balancing the workload associated to each module of the database.

5. Conclusions

In this paper, we have proposed a genetic algorithm for solving the semantic database problem that arises in crowd simulations when this database must be split in order to be distributed among several computers of a distributed system.

Performance evaluation results show that the proposed approach provides partitions that are properly balanced, avoiding the saturation of the servers as much as possible. Also, the proposed approach significantly reduces the overhead due to the partition consistency (number of locks) in regard to other local search methods, regardless of the movement pattern the agents can follow. Therefore, it allows an effective partition of the AS, eliminating the bottleneck that a centralized database represents.

References

[1] N. Beatrice, S. Antonio, L. Rynson, and L. Frederick. A multiserver architecture for distributed virtual walkthrough. In *Proceedings of ACM VRST'02*, pages 163–170, 2002.

[2] D. Diller, W. Ferguson, W. Leung, A. Benyo, and D. Foley. Behavior modelling in comercial games. In *BRIMS '04: Proceedings of the 2004 Behavior Representation in Modelling and Simulation Conference*, 2004.

[3] S. Dobbyn, J. Hamill, K. O'Conor, and C. O'Sullivan. Geopostors: a real-time geometry/impostor crowd rendering system. *ACM Trans. Graph.*, 24(3):933–933, 2005.

[4] S. Donikian. Informed virtual environments. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*. ACM Press, 2004.

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

[6] C. Greenhalgh, A. Bullock, E. Frecon, D. Llyod, and A. Steed. Making networked virtual environments work. *Presence: Teleoperators and Virtual Environments*, 10(2):142–159, 2001.

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

[8] A. Iglesias and F. Luengo. New goal selection scheme for behavioral animation of intelligent virtual agents. *IEICE Transactions on Information and Systems, Special Issue on 'CyberWorlds'*, E88-D(5):865–871, 2005.

[9] P. A. Kruszewski. A game-based cots system for simulating intelligent 3d agents. In *BRIMS '05: Proceedings of the 2005 Behavior Representation in Modelling and Simulation Conference*, 2005.

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

[11] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the 5th Berkeley symposium on mathematical statistics and probability*, pages 281–297, 1967.

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

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

[14] D. Milojicic, V. Kalogeraki, R. Lukose, K. Nagaraja, J. Pruyne, B. Richard, S. Rollins, and Z. Xu. Peer-to-peer computing. Technical report, Technical Report HPL-2002-57, HP Laboratories, Palo Alto, 2002.

[15] J. S. Monzani, A. Caicedo, and D. Thalmann. Integrating behavioral animation techniques. In *Proceedings of EUROGRAPHICS 2001*, pages 309–318. Computer Graphics Forum, Vol. 20, Issue 3, 2001.

[16] S. Mooney and B. Games. *Battlezone: Official Strategy Guide*. BradyGame Publisher, 1998.

[17] P. Morillo, W. Moncho, J. M. Ordua, and J. Duato. Providing full awareness to dves based on peer-to-peer architectures. *Lecture Notes on Computer Science*, 4035:336–347, 2006.

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

[19] H. Nakanishi and T. Ishida. Freewalk/q: social interaction platform in virtual space. In *VRST '04: Proceedings of the ACM symposium on Virtual reality software and technology*, pages 97–104, New York, NY, USA, 2004. ACM Press.

[20] S. Raupp and D. Thalmann. Hierarchical model for real time simulation of virtual human crowds. *IEEE Transactions oon Visualization and Computer Graphics*, 7(2):152–164, 2001.

[21] W. Shao and D. Terzopoulos. Autonomous pedestrians. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 19–28, New York, NY, USA, 2005. ACM Press.

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

[23] M. Sung, M. Gleicher, and S. Chenney. Scalable behaviors for crowd simulations. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 519–528. ACM Press, 2004.

[24] F. Tecchia, C. Loscos, and Y. Chrysathou. Visualizing crowds in real time. *Computer Graphics Forum*, 21, 2002.