Algoritmos Genéticos para proporcionar QoS en Entornos Virtuales Distribuidos: SEGA*

S. Rueda, P. Morillo, J. M. Orduña
Dept. de Informática
Universidad de Valencia
Juan.Orduna@uv.es

J. Duato
DISCA
Universidad Politécnica de Valencia
jduato@gap.upv.es

Resumen

Las arquitecturas basadas en redes cliente-servidor se han convertido en el estándar de-facto para los sistemas de Entornos Virtuales Distribuidos (DVE). Estos sistemas permiten que una misma escena virtual 3D sea compartida por gran cantidad de usuarios remotos. Para que un sistema DVE pueda proporcionar calidad de servicio, los clientes del sistema deben asignarse a los servidores teniendo en cuenta tanto la productividad como la latencia del sistema. Este problema se conoce como el problema de la calidad de servicio.

En este artículo proponemos una nueva implementación de algoritmos genéticos para resolver el problema de la calidad de servicio en los Entornos Virtuales Distribuidos. Los resultados de evaluación muestran que debido a la capacidad de estas técnicas para encontrar buenos caminos de búsqueda escapando de mínimos locales, podemos obtener mejores soluciones que con otras heurísticas, a la vez que se reducen los tiempos de ejecución. Esta mejora puede incrementar significativamente la calidad de servicio proporcionada en los sistemas DVE.

1. Introducción

En los últimos años, los sistemas de Entornos Virtuales Distribuidos han experimentado un crecimiento espectacular. En estos sistemas múltiples usuarios interaccionan entre sí dentro de un mismo mundo virtual compartido a través de los computadores clientes interconectados mediante una o varias redes de interconexión. Para lograr la inmersión en estos mundos virtuales, el sistema se encarga de renderizar las imágenes del entorno tal y como las percibiría un usuario si se encontrara en ese punto del mundo. Cada usuario está representado dentro del mundo virtual por una entidad denominada *avatar* cuyo estado controla el usuario. Dado que en un DVE los avatares pueden verse mutuamente, cada cambio en un avatar debe transmitirse al resto de avatares dentro del mundo virtual.

Las arquitecturas basadas en redes cliente-servidor se han convertido en los últimos años en el estándar de-facto para el diseño de Entornos Virtuales Distribuidos [11, 5]. En estas arquitecturas el control de la simulación recae sobre un conjunto de servidores interconectados. Cada cliente se conecta a uno y sólo a uno de estos servidores. Cuando un usuario modifica un avatar, el servidor al que se encuentra conectado este avatar envía un mensaje de actualización al resto de usuarios avisando de los cambios que se han producido, de modo que todos los usuarios del sistema tengan una visión del mundo virtual consistente y actualizada. Cuando el número de clientes conectados comienza a crecer, debemos limitar el número de mensajes de actualización para que no se produzca una explosión de mensajes circulando por la red. Para ello, en la literatura de la materia se han introducido conceptos como Areas de Interés (AOI) [11] para intentar reducir el número de vecinos con los que

*Este trabajo ha sido financiado por el MCYT bajo el epígrafe TIC2003-08154-C06-04

un avatar debe comunicarse. Este tipo de conceptos definen un área de vecindad para un avatar, de tal manera que un avatar i sólo deberá comunicar sus movimientos (enviando un mensaje de actualización) a los avatares que se encuentren dentro del AOI de i. Los avatares que se encuentran dentro del AOI de un avatar i se denominan *avatares vecinos* de i. La Figura 1 muestra un ejemplo de un sistema DVE con una arquitectura basada en redes servidor. En este ejemplo, el AOI de un avatar se representa por un círculo.

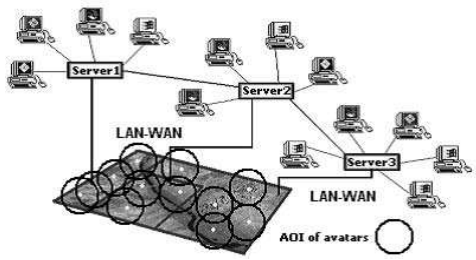

Figura 1: Examples of AOI in a DVE system whose architecture is based on networked servers.

El *problema del particionado* [5] se ha presentado como uno de los aspectos clave en el diseño de sistemas DVE basados en arquitecturas red-servidor. Este problema consiste en asignar los avatares entre los distintos servidores del sistema, de tal manera que se logre una distribución eficiente de la carga del sistema. En este campo hemos desarrollado un método de particionado que proporciona un incremento significativo en el rendimiento de los sistemas DVE [10].

Para proporcionar calidad de servicio (QoS), en primer lugar el método de particionado deberá asegurarnos una productividad del sistema máxima, garantizando la no saturación del sistema [10]. Asegurado este extremo, el método de particionado puede dedicar los recursos sobrantes a reducir el tiempo de respuesta que se les ofrece a los avatares. Esta tarea supone buscar un compromiso entre productividad y latencia, y es la que se conoce como el problema de la Calidad de Servicio (QoS). Dado que este problema es NP-completo, en un trabajo previo presentamos una solución al problema de la QoS en sistemas DVE [9]. En dicho trabajo se propone la implementación y evaluación de dos métodos heurísticos, Simulated Annealing (SA) y Greedy Randomized Adaptive Search (GRASP), que permiten mejorar la calidad de servicio en los sistemas DVE.

En el presente trabajo se propone la implementación y evaluación un Algoritmo Genético (SEGA) para resolver el problema de la calidad de servicio en los mundos virtuales distribuidos. La evaluación de prestaciones nos muestra que esta técnica evolutiva es capaz de mejorar la calidad de servicio que el sistema DVE proporciona, al mismo tiempo que reduce los tiempos de ejecución en relación a las técnicas previamente evaluadas. Por lo tanto, podemos considerar esta técnica como un método heurístico apropiado para resolver el problema de la calidad de servicio en sistemas DVE.

El resto del artículo está organizado de la siguiente manera: la sección 2 describe los problemas que se deben resolver para dotar de calidad de servicio a los sistemas DVE. La sección 3 presenta el nuevo algoritmo genético que proponemos. En la sección 4 se expone la evaluación de prestaciones del algoritmo propuesto. Por último, la sección 5 presenta las conclusiones más importantes.

2. Estado del Arte

El problema de la calidad de servicio (problema QoS) en los sistemas DVE ya había sido tratado con anterioridad, y se han propuesto dos alternativas para solucionarlo [1, 3]. En una se emplea un método de compensación de latencia para paliar los efectos del jitter en la red [3]. En la otra se utiliza técnicas de renderización adaptativa [1] que modifica la resolución de los modelos 3D en función de la velocidad de conexión del cliente. Sin embargo, ninguna de ellas tiene en cuenta el comportamiento no lineal que presentan los sistemas DVE con respecto a la carga asignada a cada servidor, tal y como se describe en [8]. En consecuencia, ninguna de todas estas estrategias

es capaz de garantizar que las prestaciones que se proporcionan a los avatares no se degraden bajo un cierto umbral.

Estudios recientes sobre la percepción humana demuestran que en sistemas DVE el usuario percibe QoS si el tiempo medio de ida y vuelta para los mensajes enviados por cada avatar (ASR: *la respuesta media del sistema*) es menor a 250 ms. [3]. Dado que el ASR que se le ofrece a un avatar i depende de a qué servidores se encuentran asignados sus vecinos, para que un método de particionado proporcione QoS debe maximizar el número de avatares vecinos asignados al mismo servidor. Al mismo tiempo, el algoritmo deberá también balancear la carga que deben soportar los servidores (medida en número de avatares asignados) para conseguir mantener al sistema lejos del punto de saturación. Como último requisito el algoritmo deberá realizar todo esto sin cambiar de servidor a más de un 30 % de los avatares del sistema para garantizar su estabilidad [4]. Por lo tanto, el problema QoS consistirá en encontrar la mejor partición posible cumpliendo simultáneamente estos 3 requisitos.

El problema de la QoS en sistemas DVE es un problema NP-completo, y por lo tanto una búsqueda heurística puede proporcionar soluciones eficientes. Para poder llevar a cabo esta búsqueda necesitamos asignar a cada posible solución (partición) un valor de calidad que mida lo bien que cumple una determinada partición estos tres requisitos. En un artículo anterior ya definimos esta función de calidad [9], por lo que a continuación sólo la describiremos brevemente. Dicha función se muestra en la ecuación 1, y se compone de tres términos. El problema de la calidad de servicio consistirá en realizar una búsqueda heurística para encontrar la partición con el menor valor posible de F_{QoS}.

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

La función F_{QoS} es la suma de tres términos diferentes. El primero de ellos, calcula la suma de los valores de la función $h_{cpu}(i)$ en todos los servidores del sistema. La función $h_{cpu}(i)$ evalúa el porcentaje estimado de utilización de CPU en cada servidor i. La Figura 2 muestra el comportamiento de esta función. De esta forma, este término otorgará un valor pobre (alto) a la función de calidad de una partición que lleve a cualquiera de los servidores del sistema DVE a acercarse o sobrepasar el punto de saturación punto.

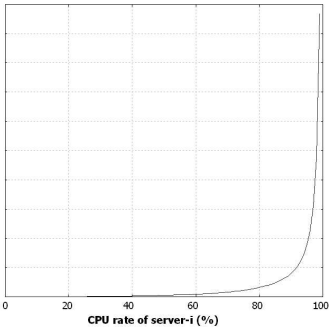

Figura 2: Comportamiento de $h_{cpu}(i)$

El segundo término de la función mide la latencia que la partición proporciona a cada uno de los avatares. El término $h_{asr}(i)$ mide la respuesta media (ASR) estimada del sistema a cada avatar. En la Figura 3 podemos ver cómo se comporta esta función compuesta por dos secciones. Dicha función trata de penalizar aquellas particiones en las que el ASR medio de los avatares es superior a 250 ms.

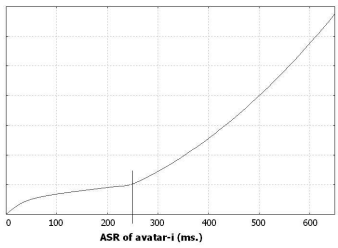

Figura 3: Comportamiento de $h_{asr}(i)$

Finalmente, el término n_m refleja el número de avatares que deben ser migrados de un

servidor a otro para obtener una partición determinada, a partir de la partición actual. Está función presenta dos secciones con diferente comportamiento, como muestra la Figura 4, y penaliza a aquellas soluciones que obliguen a trasladar de servidor a más del 30 % de los avatares.

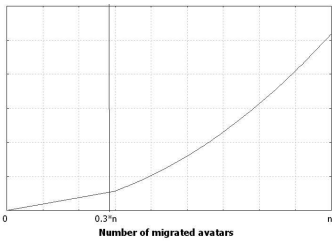

Figura 4: Comportamiento de n_m

3. Algoritmo Genético para Calidad de Servicio en sistemas DVE

En este apartado se describe la implementación de la variante de los Algoritmos Genéticos que hemos desarrollado para resolver el problema de la calidad de servicio en los sistemas DVE, que hemos denominado como *SEGA (Sexual Elitist Genetic Algorithm)*.

Los Algoritmos Genéticos (AG) son métodos heurísticos de búsqueda basados en el concepto darvinista de evolución por selección natural [2]. Se caracterizan por partir de una población inicial de posibles soluciones (cromosomas) a la que hacen evolucionar siguiendo ciertas reglas hasta alcanzar el estado final, tras cumplir una condición de convergencia, en la que la función de calidad alcanza un valor óptimo aceptable. Cada iteración del algoritmo consiste en generar una nueva población a partir de la existente empleando técnicas de recombinación o mutación sobre los cromosomas que forman dicha población. Un cromosoma está formado por dos elementos, por un lado el *genotipo* es la codificación del problema y por otro el *fenotipo* que contiene las características que el cromosoma representa, es información adicional sobre el problema a resolver y la aprovecharemos para ajustar el comportamiento del algoritmo. Además cada cromosoma tiene asociado un valor de *fitness*, que mide la bondad de la solución y que permitirá comparar unos cromosomas con otros.

En nuestra implementación, el genotipo es un vector con tantos elementos como avatares existan en el sistema, cada uno de los cuales contiene un número indicando a qué servidor está asociado ese avatar. Si tenemos N avatares en el sistema, este vector tendrá N elementos y si tenemos S servidores cada elemento del vector será un número entre 0 y $S-1$. El fenotipo contiene información a cerca de la carga estimada (medida en términos de utilización de la CPU) que un avatar añade al servidor al que es asignado. Además incluye información indicando si un avatar es un *boarder avatar* o no. Un avatar asignado a un servidor s se denomina *boarder avatar* si alguno de sus vecinos (los avatares que se encuentran dentro de su AOI) está asignado a un servidor distinto de s. Como función de calidad utilizaremos el valor de F_{QoS}, por lo tanto el algoritmo genético deberá encontrar el cromosomas con el menor valor de fitness posible.

La mayoría de métodos heurísticos se basan en la generación aleatoria de una población inicial. Sin embargo, si esta población inicial se define correctamente, con conocimiento expreso del problema y no de forma aleatoria, entonces la búsqueda puede obtener una buena aproximación al óptimo global con mucha mayor rapidez. El problema que surge en estos casos es que hay que evitar que el algoritmo converja demasiado pronto y caiga en un mínimo local, para ello hay mantener un mínimo de diversidad estructural entre los diferentes cromosomas que componen la población [6]. Siguiendo esta línea, la población inicial generada en el algoritmo propuesto se obtiene de variaciones a la solución del problema del particionado que proporciona el algoritmo ALB [7]. Este algoritmo de balanceo de carga proporciona particiones del sistema DVE correctamente balanceadas, lo que nos asegura bajos valores de fitness pues tendremos valores bajos para el primer término ($\sum_{i=0}^{s} h_{cpu}(i)$) de la función de calidad F_{QoS}.

Cada iteración del algoritmo consiste en crear una generación de cromosomas *descendientes* a partir de una generación de cromosomas *ascendientes*. La forma en la que pasar de generación en generación determina el comportamiento del algoritmo genético. Para SEGA decidimos emplear reproducción sexual [6], por lo que para generar un descendiente necesitamos dos progenitores. Sin embargo, con objeto de tener mayor diversidad también se ha introducido una cierta tasa de reproducción asexual (sólo hace falta un antecesor para generar un descendiente). Además, y de ahí el nombre del método, hemos decidido emplear *elitismo*, lo que significa que los mejores individuos de una generación pasarán directamente a la siguiente sin sufrir variación alguna [6]. El empleo de una solución elitista acelera el proceso de convergencia del algoritmo y por lo tanto permite encontrar antes una solución óptima aceptable. De este modo, en cada iteración de SEGA, se genera una población intermedia con $P+N_{elitist}$ cromosomas, donde los $N_{elitist}$ cromosomas son aquellos que tienen en la generación anterior tenían los mejores (los menores) valores de fitness. Al final de la iteración, la nueva generación estará formada por los P mejores cromosomas (los de menor fitness) en la población intermedia.

Para la reproducción sexual, se elige como primer progenitor para generar el i-ésimo cromosoma al cromosoma i-ésimo de la generación anterior. El segundo progenitor se elige de forma aleatoria entre el 50 % de la generación anterior con mejores valores de fitness. Una vez seleccionados los cromosomas progenitores, el descendiente i-ésimo se obtiene del cruce (crossover) de ambos. El crossover puede ser de tres tipos, también aleatoriamente seleccionado, crossover de un punto, crossover multipunto o crossover uniforme [6]. Para el caso de la reproducción asexual, el descendiente es el mismo i-ésimo cromosoma progenitor.

Una vez obtenido el cromosoma descendiente, este puede sufrir dos procesos más que alterarán su contenido. El primero de ellos es un proceso de recombinación que sólo se aplica en las primeras iteraciones del algoritmo para garantizar la diversidad estructural de los cromosomas, evitando así la convergencia prematura. Este mecanismo consiste seleccionar aleatoriamente dos avatares e intercambiar los servidores a los que se encuentran asignados. Para explorar sólo soluciones altamente probables este mecanismo sólo se aplica sobre avatares boarder. Por último los cromosomas pueden sufrir un proceso de mutación consistente en alterar el servidor al que un avatar, seleccionado aleatoriamente, está asignado. En el siguiente esquema pseudo-código podemos ver el proceso completo que se produce en cada iteración t+1 que parte de la generación previa G_t.

```
Iteración t+1 (Generación G_t) CONST
   N_elitist    /* N° cromosomas elitistas */
   P            /* N° cromosomas en Gt */
   N            /* N° avatares en el DVE */
   T_sex        /* Tasa reproducción sexual */
   T_mut        /* Tasa mutación */
TYPE
   cromosoma : int[N]
VAR
   Anc1, Anc2 : cromosoma  /* Padres */
   Desc : cromosoma        /* Hijo */
   tc : int /* Tipo de crossover */
inicio
   Copiar_N_elitist_de_G_t_a_G_inter()
   desde i:=N_elitist hasta P+N_elitist hacer
      Anc1 := G_t[i]
      a := Select_Reproduccion(T_sex)
      si a = Sexual entonces
         Anc2 := Select_Progenitor(G_t)
         tc := Select_Crossover()
         Desc := Crossover(Anc1,Anc2,tc)
      sino /* Asexual */
         Desc := Anc1
      fin_si
      si (!converg_cond) entonces
         Recombination(Desc)
      fin_si
      if (random() <T_mut) entonces
         mutation(Desc)
      fin_si
      Eval_Fitness(Desc)
      G_inter[i] := Desc
   fin_for
   G_t+1:=Sort(G_inter,P)
```

`fin`

Algoritmo 1: Algoritmo SEGA

Para establecer la condición de convergencia (condición de finalización del algoritmo), hemos considerado tres parámetros distintos. El primero de ellos se trata de la desviación estándar del valor de la función fitness (F_{QoS}). Cuando todos los individuos en la población son muy parecidos (la desviación estándar del fitness es baja) difícilmente podremos obtener una solución mejor que la actual. En consecuencia cuando la desviación estándar sea menor que un cierto umbral detendremos el algoritmo. El segundo parámetro que hemos considerado ha sido el número de generaciones consecutivas sin mejorar el valor del fitness. Si tras un determinado número de iteraciones no conseguimos encontrar un cromosoma mejor (una solución mejor) daremos por buena la actual. Por último, hemos limitado también el número total de iteraciones que el algoritmo puede realizar. Cuando se cumpla cualquiera de estas tres condiciones el algoritmo se detendrá.

El algoritmo que acabamos de describir consta de diversos parámetros susceptibles de ser ajustados con el fin de obtener el máximo rendimiento posible de SEGA. Esos parámetros son el número de cromosomas en la población (P), la tasa de reproducción sexual (probabilidad de seleccionar reproducción sexual frente a asexual), la tasa de mutación, el número de cromosomas elitistas ($Nelitist$), y los umbrales para la desviación estándar, el número de repeticiones consecutivas sin mejorar el fitness y el número máximo de iteraciones. Hemos ajustado de forma empírica todos estos parámetros, para ello hemos realizado pruebas sobre dos configuraciones de sistemas DVE distintas, denominadas MEDIUM1 y MEDIUM2 [9]. El primero es un ejemplo de DVE de tamaño medio, tendremos 250 avatares y 3 servidores. El segundo es un ejemplo de DVE mayor con 700 avatares y 10 servidores. Debido a limitaciones de espacio, presentamos aquí únicamente los valores obtenidos como valores óptimos para cada uno de estos parámetros.

Además, para evaluar las prestaciones de los métodos de particionado hemos trabajado con tres distribuciones de avatares distintas dentro del mundo virtual. Hemos elegido estas distribuciones por ser las más ampliamente utilizadas en la literatura de la materia, estas son: uniforme, sesgada y agrupada [5]. La razón de usar tres distribuciones distintas es que la carga que introducen en el sistema es diferente para cada una de ellas. En la Figura 5 podemos ver un ejemplo de estas distribuciones en un mundo virtual 2D en la que el mundo es el cuadrado y los avatares son los puntos negros. Hemos ajustado el algoritmo propuesto sobre estas tres distribuciones de avatares en el mundo virtual.

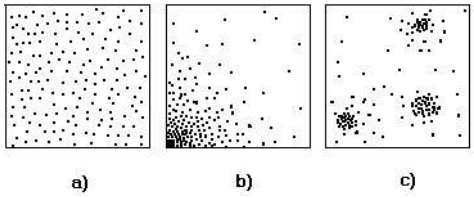

Figura 5: Distributions of avatars in a 2-D virtual world: a) Uniform b) Skewed c) Clustered

La tabla 1 muestra la totalidad de parámetros ajustados para el algoritmo SEGA.

Parámetro	Valor
Tamaño Población (P)	100
Porcentaje Elitistas ($N_{elitist}$)	50
Tasa Mutación (T_{mut})	0.05
Tasa Reproducción Sexual (T_{sex})	0.75
Iteraciones max	300
Repeticiones max	50
Umbral diversidad	0.005

Cuadro 1: Parámetros SEGA

4. Evaluación de prestaciones

Para poder comparar la heurística descrita en el apartado anterior, hemos enfrentado los resultados obtenidos con esta técnica con los resultados obtenidos mediante otras dos heurísticas, SA y GRASP, que ya se habían adap-

tado al problema de la calidad de servicio de sistemas DVE [9]. Los resultados ha sido obtenidos de forma empírica sobre las configuraciones que se describen en la sección previa, tanto MEDIUM1 como MEDIUM2. No obstante, debido a limitaciones de espacio sólo presentamos aquí los resultados sobre MEDIUM2, dado que los resultados en MEDIUM1 fueron muy similares.

La tabla 2 muestra los resultados obtenidos cuando los avatares se distribuyen en el mundo virtual siguiendo los tres tipos de distribución presentados en la sección anterior. En dichas tablas, cada columna representa el resultado obtenido por cada método heurístico: SA, GRASP y SEGA. Además, la primera columna presenta los valores obtenidos al aplicar un algoritmo de particionado no orientado a calidad de servicio, el algoritmo ALB[7]. Este algoritmo simplemente proporciona una partición inicial con carga bien balanceada. Cada fila de la tabla introduce una medida de prestaciones distinta: el porcentaje máximo estimado de utilización de CPU entre todos los servidores del sistema (no debe superar el 99 %); el coste del sistema en términos de la función de calidad F_{QoS} asociada la partición resultante proporcionada por cada método; el número estimado de avatares a los que se les va a proporcionar calidad de servicio[9]; el número de avatares que deberán ser migrados para llegar a la partición resultado partiendo de la partición actual (inicial); y por último, el tiempo de ejecución en segundos.

Como podemos observar en los cuadros, para todas las distribuciones de organización que siguen los avatares en el mundo virtual, todos los métodos cumplen los requisitos de no superar el punto de saturación del sistema, CPU por debajo del 99 % de utilización, y de no migrar más del 30 % de los avatares.

Cuando los avatares del mundo virtual se organizan siguiendo una distribución uniforme, la carga del sistema es baja, no llega al 20 %, y los tres métodos estudiados se comportan eficientemente y de manera similar.

Cuando estudiamos el comportamiento en distribuciones sesgadas, la carga que introducen en el sistema el mismo número de avatares es mayor, con lo que aumenta la utilización de la CPU. En estos casos, SEGA mejora considerablemente la calidad de servicio de las particiones propuestas frente a las que proporcionan SA y GRASP, incrementando en un 24 % el número de avatares con calidad de servicio si lo comparamos con GRASP y en un 48 % si lo comparamos con SA. Todo ello además, con el menor tiempo de ejecución necesario para obtener la partición final.

Dist. Uniforme				
	ALB	SA	GRASP	SEGA
CPU_{max}	17	17	19	16
F_{QoS}	73851	69633	62048	61558
QoS	496	526	619	621
$\Gamma(P_0)$		54	101	101
T_{exe} *(s.)*		7,68	7,31	9,07
Dist. Sesgada				
	ALB	SA	GRASP	SEGA
CPU_{max}	58	53	58	54
F_{QoS}	99408	88426	82857	75326
QoS	211	290	347	431
$\Gamma(P_0)$		194	81	169
T_{exe} *(s.)*		84,08	20,07	18,43
Dist. Agrupada				
	ALB	SA	GRASP	SEGA
CPU_{max}	79	69	72	70
F_{QoS}	99690	92677	88960	84774
QoS	314	349	415	436
$\Gamma(P_0)$		95	82	120
T_{exe} *(s.)*		48,89	43,66	23,40

Cuadro 2: Prestaciones para configuración MEDIUM2

Finalmente, cuando los avatares presentan una distribución agrupada obtenemos los valores máximos de utilización de CPU, o lo que es lo mismo, que esta es la distribución en la que los avatares introducen mayor carga en el sistema. Aquí SEGA es capaz de proporcionar particiones con un valor de F_{QoS} menor en un 5 % que las que se obtienen con GRASP y un 8.5 % menor que los que se obtiene con SA. Aunque la mejora más importante se obtiene en cuanto a tiempos de ejecución se refiere, reduciendo a la mitad el tiempo de ejecución de GRASP.

Podemos decir, por tanto, que las presta-

ciones de SEGA aumentan conforme la distribución de los avatares en el mundo virtual genera mayor carga al sistema.

5. Conclusions

E este trabajo hemos propuesto una variante sexual y elitista de un algoritmo genético para resolver el problema de la calidad de servicio para mundos virtuales distribuidos. Hemos comparado las prestaciones de dicho método con las obtenidas por otros métodos heurísticos aplicados al mismo problema.

Los resultados nos muestran que el método propuesto no introduce mejoras en el rendimiento cuando la carga del sistema es baja, pero sin embargo se ajusta muy bien a la carga que introducen los avatares, mejorando las prestaciones conforme el sistema se va cargando. Dado que el propósito del método es proporcionar calidad de servicio al mayor número de avatares posible, sin importar la distribución en el mundo virtual que estos adapten, estos resultados validan al método SEGA propuesto como método heurístico apropiado para proporcionar calidad de servicio en sistemas DVE.

Referencias

[1] Z. Choukair, D. Retailleau, and M. Hellstrom. Environment for Performing Collaborative Distributed Virtual Environments with QoS. In *Proceedings of the International Conference on Parallel and Distributed Systems (ICPADS'00)*, pages 111–118. IEEE Computer Society, 2000.

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

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

[4] K. Lee and D. Lee. A Scalable Dynamic Load Distribution Scheme for Multi-Server Distributed Virtual Environment Systems With Highly-Skewed User Distribution. In *Proceedings of the 10th ACM Symposium on Virtual Reality Software and Technology (VRST 2003)*, pages 160–168. ACM, 2003.

[5] John C.S. Lui and M.F. Chan. An Efficient Partitioning Algorithm for Distributed Virtual Environment Systems. *IEEE Trans. Parallel and Distributed Systems*, 13, 2002.

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

[7] P. Morillo, J. M. Orduña, M. Fernández, and J. Duato. An Adaptive Load Balancing Technique for Distributed Virtual Environment Systems. In *Proc. of Intl. Conf. on PDCS'03*, pages 256–261. IASTED, ACTA Press, 2003.

[8] P. Morillo, J. M. Orduña, M. Fernández, and J. Duato. On the Characterization of Distributed Virtual Environment Systems. In *Euro-Par' 2003 - Lecture Notes in Computer Science 2790*, pages 1190–1198. ACM, Springer-Verlag, 2003.

[9] P. Morillo, J. M. Orduña, M. Fernández, and J. Duato. A Comparison Study of Metaheuristic Techniques for Providing QoS to Avatars in DVE Systems. In *ICCSA' 2004 - Lecture Notes in Computer Science 3044*, pages 661–670. Springer-Verlag, 2004.

[10] P. Morillo, J. M. Orduña, M. Fernández, and J. Duato. A Fine-Grain Method for Solving the Partitioning Problem in Distributed Virtual Environment Systems. In *Proc. of Intl. Conf. on PDCS'04*, pages 292–297. IASTED, ACTA Press, 2004.

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