*Conciencia Tecnológica No. 37, Enero - Junio 2009*

# Evaluando Tres Metaheurísticas en una Arquitectura de Balanceo Dinámico de Carga Diseñada Bajo Patrones en CORBA

Investigación

Dr. Francisco Javier Luna Rosas, Jesús Chávez Esparza, Dr. Julio César Martínez Romo
Instituto Tecnológico de Aguascalientes, Av. Adolfo López Mateos 1801 Ote., Fracc. Bona Gens, C.P. 20256, Aguascalientes, Ags., México e-mail: fjluna@ita.mx, jucemaro@yahoo.com
Instituto Tecnológico de Toluca, Av. Instituto Tecnológico s/n, Ex-Rancho La Virgen, Metepec, Estado de México, México; e-mail: j_chavez2000@hotmail.com

**Resumen**

En los últimos años se ha modificado la concepción sobre la construcción de software, como una actividad de arquitectura, fijando la atención de los desarrolladores en un nuevo paradigma basado en el uso de patrones, transformando el diseño de artefactos de software, buscando aumentar su usabilidad y facilitar la tarea de construcción, además de conseguir con ello desarrollos más flexibles y eficientes en su desempeño. Este trabajo se centra en destacar el diseño mediante patrones de software de un servicio de balanceo de carga para aplicaciones distribuidas desarrolladas bajo el estándar de CORBA. Además, en el patrón "estrategias" se evalúa tres metaheurísticas como Algoritmos Genéticos, Simulado Recocido y Búsqueda Tabú. La aplicación de estas metodologías sugiere un mejor desempeño a la arquitectura de balanceo de carga y en nuestro resultado Búsqueda Tabú muestra mejor desempeño.

**Palabras clave:** CORBA, Balanceo de Carga, Metaheurísticas.

**Abstract**

In the last years have modified the conception on the software construction, as an architecture, fixing the attention of the developers in a new paradigm based on the use of patterns, transforming the design of software devices, looking for the increase from their usability and facilitate the construction of task, besides getting with it more flexible and more efficient in the developments in their performance. This work is centered in the design of patterns from software the use of a service of dynamic load balancing for distributed applications under the standard of CORBA. Also, the pattern "strategy" was evaluated with three metaheuristics like as Simulated Annealing, Genetic Algorithms, and Tabu Search. The application of these strategies suggests a better performance in the dynamic load balancing architecture and in our result Tabu Search shows better performance.

**Keywords:** CORBA, Load Balancing, Metaheuristics.

## Introducción

Las estrategias de balanceo de carga pueden ser divididas en dos grandes grupos: estrategias de balanceo estático y estrategias de balanceo dinámico. Las estrategias de balanceo estático, obtienen la localización de todos sus procesos antes de comenzar la ejecución. Las estrategias de balanceo dinámico intentan equilibrar la carga en tiempo de ejecución [1].

Cuando se hace balanceo de carga se aplica una técnica bien establecida para utilizar los recursos de computación disponibles más efectivamente, las aplicaciones distribuidas pueden mejorar su escalabilidad, tiempo de respuesta y uso de recursos empleando balanceo de carga en varias formas y en varios niveles del sistema, incluyendo la red, el sistema operativo y a nivel middleware [2].

1) Balanceo de Carga a Nivel de Red. Los Servidores de Nombres de Dominio (DNS) y los Ruteadores IP´s que sirven a una gran cantidad de máquinas host proveen este tipo de balanceo de carga [3].

2) Balanceo de Carga a Nivel del Sistema Operativo. Los Sistemas Operativos Distribuidos generalmente proveen este tipo de balanceo de carga a través del agrupamiento de computadoras, compartición de carga y mecanismos de migración de procesos [4].

3) Balanceo de Carga Basado en Middleware. Las interacciones globales son acopladas por arquitecturas llamadas middleware. La arquitectura más común de middleware para aplicaciones orientadas a objetos distribuidas es Common Object Request Broker Architecture (CORBA) [5]. Ha habido importantes enfoques para extender funcionalidades de CORBA que soporten balanceo de carga, por ejemplo, TAO [6], VisiBroker [7], MICO [8], etc. Para consolidar este enfoque y resolver el problema, la OMG diseño un Request For Proposal (RFP) [5], que es una propuesta para extender la funcionalidad de CORBA para balancear y monitorear carga en ambientes basados

en CORBA, para procesar distribuidamente y con alto desempeño las aplicaciones implementadas en el estándar.

A continuación se analizan a detalle los conceptos claves de un servicio de balanceo de carga basado en CORBA, Figura 1 [6]:

Balanceador de carga. Es un componente que intenta distribuir la carga a través de grupos de servidores de una manera óptima. Un balanceador de carga debe consistir de un simple servidor centralizado o múltiples servidores descentralizados que colectivamente forman un balanceador lógico.

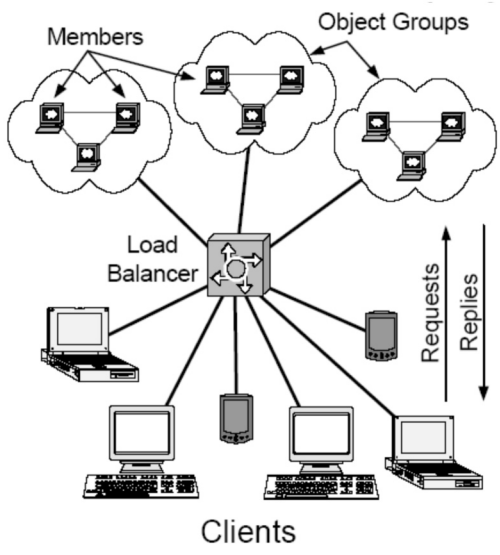


Figura 1. Conceptos Claves en un Servicio de Balanceo de Carga Middleware [2].

Réplica. Es un duplicado de un objeto particular sobre un servidor que es manejado por un balanceador de carga. Esta ejecuta las mismas tareas que el objeto original.

Grupo de objetos. Es un grupo de réplicas a través del cual la carga es balanceada. Las réplicas en tal grupo implementan las mismas operaciones remotas.

Sesión. En el contexto de balanceo de carga middleware, una sesión se define como el periodo de tiempo que un cliente invoca operaciones remotas (requerimientos) para acceder servicios proporcionados por objetos en un servidor particular (Figura 1).

En los últimos años se ha acuñado el término "metaheurístico, introducido por Fred Glover en 1986 y apoyado por diferentes eventos científicos de carácter internacional, como el congreso Metaheuristic International Conference, o la revista Journal of Heuristics. El término metaheurístico establece una diferencia conceptual entre el conjunto de reglas que permiten diseñar un procedimiento de resolución heurístico y el propio procedimiento de resolución. En este sentido el prefijo meta indica un mayor nivel de abstracción, en cuanto que las propias reglas, denominadas procedimiento metaheurístico, no están ligadas a ningún problema específico [9].

## Metodología

### A. Diseño del Servicio de Balanceo de Carga Bajo CORBA.

El proceso de desarrollo de software moderno incluye el diseño de patrones de software los cuales son una descripción formal de buenas soluciones a problemas ya planteados anteriormente, estos pueden ser usados por los desarrolladores de software como una colección de conocimiento experto acerca de un problema especifico. Un amplio rango de patrones de diseño puede ser obtenido, en [10], [11].

En esta sección se analiza la construcción de la arquitectura de balanceo de carga mediante patrones de software que fueron utilizados en la construcción y diseño de la misma, con ello se busca facilitar la futura reutilización del diseño y la arquitectura del sistema, logrando crear un lenguaje común de comunicación entre los desarrolladores además de promover el uso de buenas prácticas en el proceso de diseño y construcción.

#### A.1 Patrón Broker

Este patrón, presenta un conjunto de componentes desacoplados que interactúan a través de invocaciones a servicios remotos [10]. En el servicio de balanceo propuesto, representa el equilibrio que permite a los componentes acceder a los servicios que ofrecen otros componentes mediante invocaciones de servicio remotas y transparentes a la localización de los servidores, para cambiar, añadir o eliminar componentes en tiempo de ejecución (Figura 2).

En el balanceo estos requerimientos son pasados a través de proxies al broker. El broker busca para su localización al objeto servidor correspondiente y le pasa los requerimientos.

#### A.2 Patrón Interceptor

Permite agregar fácilmente funcionalidad al sistema para cambiar su comportamiento dinámicamente, sin necesidad de recompilarlo, es decir, hace el cambio de comportamiento en tiempo de ejecución [12], [13] ya sea interponiendo una nueva capa de procesamiento o cambiando el destino dinámicamente, ya que interpone

objetos los cuales pueden interceptar llamadas e insertar un procesamiento específico que puede estar basado en el análisis del contenido, además de redireccionar una llamada a un punto diferente, Figura 3[14].

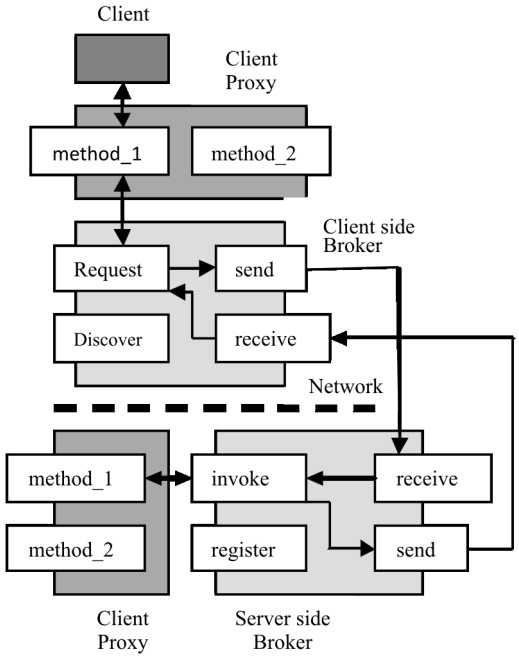


Figura 2. Patrón Broker

En el servicio de balanceo, este patrón representa el mecanismo para redirigir los requerimientos de los clientes a una réplica apropiada. Las aplicaciones basadas en CORBA contienen las construcciones necesarias para soportar transparentemente el envío de requerimientos de los clientes [13].

**A.3 Patrón Estrategia.**

En general, no es común que todas las aplicaciones distribuidas exhiban las mismas condiciones de carga, significa que algunas estrategias de balanceo son más aplicables a algunos tipos de aplicaciones que a otras. Un balanceador de carga debe ser lo suficientemente flexible para soportar diferentes tipos de estrategias de balanceo Figura 4, [10], [15].

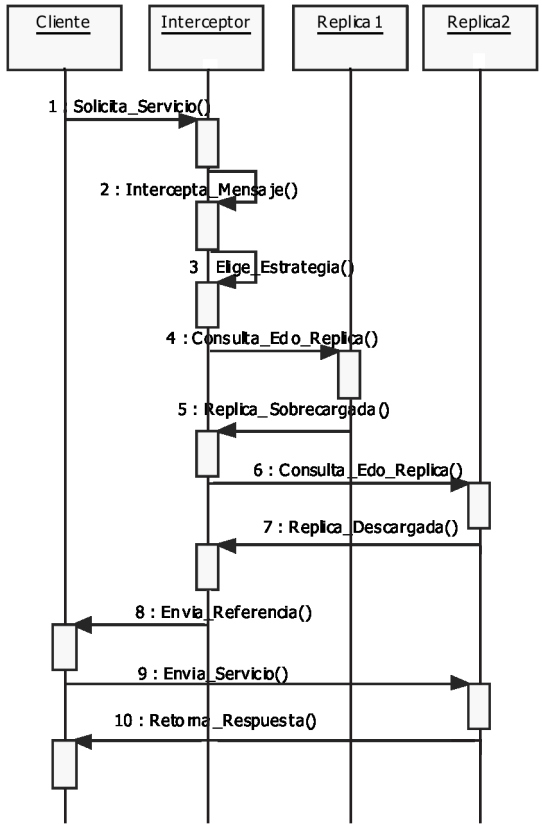


Figura 3. Diagrama de Secuencia del Patrón Interceptor.

Las estrategias encapsuladas en este patrón son las siguientes:

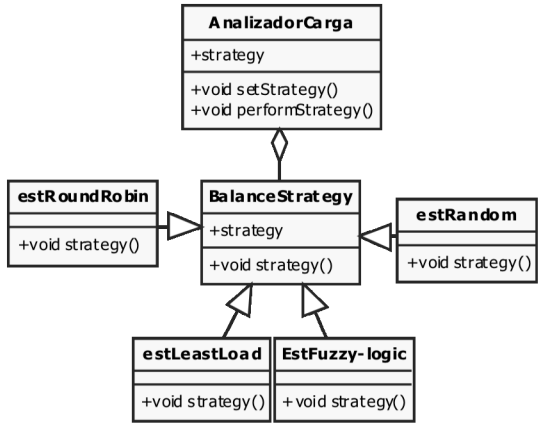


Figura 4. Diagrama de Clases del Patrón Estrategia.

**B. Las estrategias Metaheurísticas**

La mejor forma de resolver un problema de balanceo de carga es centrar el problema como un problema de optimización [4]. En la última década, los Algoritmos Genéticos (GAs), Recocido Simulado (SA), Búsqueda

Tabú (TS) han sido extensamente usados como herramientas de búsqueda y optimización en varios dominios de problemas, incluyendo las ciencias, el comercio y la ingeniería. La razón primaria de su éxito es la amplia aplicabilidad, facilidad de uso y perspectiva global [16] [17] [18] [19].

**B.1 Algoritmos Genéticos.**

John Koza define un Algoritmo Genético partiendo de los elementos básicos de que está compuesto [20]:

"El Algoritmo Genético, es un algoritmo matemático altamente paralelo que transforma un conjunto (población) de objetos matemáticos individuales (típicamente cadenas de caracteres de longitud fija que se ajustan al modelo de las cadenas de cromosomas), cada uno de los cuales se asocia con una aptitud, en una población nueva (es decir, la siguiente generación) usando operaciones modeladas de acuerdo al principio Darwiniano de reproducción y sobrevivencia del más apto y tras haberse presentado de forma natural una serie de operaciones genéticas (notablemente la recombinación sexual)".

En seguida se muestra los pasos del algoritmo genético:

1. [Iniciar] Generar una población aleatoria de n cromosomas (mejores soluciones del problema).
2. [Fitness - Elitismo] Evaluar el fitness f(x) (función objetivo) de cada cromosoma x en la población y el mejor es agregado a la nueva población.
3. [Nueva población] Crear una nueva población repitiendo los siguientes pasos hasta que la población este completa.
   3.1 [Selección] Selecciona dos cromosomas padres de la población de acuerdo a su fitness (el mejor fitness, tiene más oportunidad de ser seleccionado).
   3.2 [Cruzamiento] Con una probabilidad de cruzamiento los padres forman unos hijos. Si el cruzamiento no es llevado a cabo, los hijos son iguales a los padres.
   3.3 [Mutación] Con una probabilidad de mutación mutar los hijos (en una posición del nuevo cromosoma).
   3.4 [Aceptación] Colocar los hijos en la nueva población.
4. [Remplazar] Usar la nueva población para una corrida futura del algoritmo.
5. [Probar] Si el fin de la condición es satisfecha, detener, y, regresa la mejor solución en la actual población.
6. [Fin de ciclo] Ir al paso 2 [22].

La codificación de cromosomas es la primera pregunta a responder cuando se inicia a resolver un problema con Algoritmos Genéticos. La codificación depende de lo difícil del problema. La codificación binaria para representar la cadena de cromosomas, es decir, una posible solución al problema.

La población inicial consta de 50 cromosomas y representan una primera generación de soluciones posibles; en base a la función objetivo o fitness se calcula lo adecuado de cada solución; en nuestro caso, el mejor acomodo es que tenga una desviación estándar menor, que significa que los requerimientos fueron asignados de manera uniforme y la carga fue bien distribuida. Al existir Elitismo primeramente se seleccionan los mejores dos elementos de la población a que formen parte de la siguiente generación. Por medio de la técnica de selección de ruleta. Al seleccionarse los padres se realiza la operación de cruzamiento en un punto, así una operación de mutación con una probabilidad de baja de 0.05 por cada bit del cromosoma se lleva a cabo [17] [21].

El número de generaciones son la condición de paro del algoritmo par nuestro caso estudiamos 50 y 100 iteraciones.

**B.2. Simulado Recocido.**

El Simulated Annealing es una técnica de optimización combinatorial que se usa para afrontar problemas de gran complejidad matemática, de modo que se obtengan soluciones cercanas a la óptima.

La idea original que dio lugar al SA es llamada algoritmo de Metrópolis, el que a su vez está basado en el método equilibrio en el análisis del comportamiento microscópico de los cuerpos.

Se fundamenta en el proceso físico de calentamiento de un sólido, seguido por un enfriamiento hasta lograr un estado cristalino con una estructura perfecta. Durante este proceso, la energía libre del sólido es minimizada. A una temperatura T, en el estado con nivel de energía Eì, se genera el estado siguiente j a través de de una pequeña perturbación. Si la diferencia de energía Ej – Ei es menor o igual a cero, el estado j es aceptado. Si la diferencia de energía es mayor que cero, el estado j es aceptado con una probabilidad dada por (1), donde KB es la constante de Boltzmann. Si la disminución de la temperatura es hecha de manera paulatina, el sólido puede alcanzar el estado de equilibrio en cada nivel de temperatura.

$$e^{\frac{E_i - E_j}{k_B T}} \quad (1)$$

El SA aplica la simulación de la secuencia de estados en el proceso de enfriamiento para la solución

de problemas de minimización. El costo de la configuración se asocia a la energía y T corresponde al parámetro de control. A partir del estado i con costo f(i), se genera el estado j con costo f(j). El criterio de aceptación determina si el nuevo estado es aceptado; para esto se calcula la probabilidad:

$$Prob.Acep.\ j = \begin{cases} 1 & si\ f(j) \le f(i) \\ e^{\frac{f(i)-f(j)}{T}} & si\ f(j) > f(i) \end{cases}$$

Con esta estrategia se evita el quedar atrapado en mínimos locales. Inicialmente cuando T es grande, se permiten nuevas configuraciones que deterioren la función objetivo. A medida que disminuye la temperatura, la probabilidad de aceptar soluciones peores que la que se tiene es cada vez menor.

Es clave la determinación de la temperatura inicial To, así como la velocidad de enfriamiento y la longitud Nk de la cadena de tentativas para cada nivel de temperatura Tk.

Estos parámetros se calibran para adaptarse al tipo y tamaño del problema, de tal manera que se consigan soluciones satisfactorias con el SA [25] [17].

### B.3. Búsqueda Tabú

La Búsqueda Tabú (Tabu Search - TS) es un procedimiento metaheurístico cuya característica distintiva es el uso de memoria adaptativa y de estrategias especiales de resolución de problemas. Su filosofía se basa en la explotación de diversas estrategias inteligentes para la resolución de problemas, basadas en procedimientos de aprendizaje, principalmente en mantener una lista de limitada de estados que está prohibido volver a visitar, es decir son estados Tabú. El marco de memoria adaptativa de TS explota la historia del proceso de resolución del problema haciendo referencia a cuatro dimensiones principales, consistentes en la propiedad de ser reciente, en frecuencia, en calidad, y en influencia. [23] [24]

En la figura 6 se muestra los pasos de la búsqueda tabú. Donde de la solución actual que se presume es la mejor solución se exploran n movimientos y se encuentran nuevas soluciones, de ellas se selecciona la mejor, se revisa que no este marcada como una solución prohibida o tabú, si no es así, es la nueva solución y es marcada como tabú, en caso contrario se revisa el criterio de aspiración que consiste en invalidar la clasificación tabú de este movimiento y seleccionarlo como mejor iteración siempre y cuando el movimiento produce una solución con un valor de función objetivo que es superior a cualquier valor de aislamiento previo. Cuando no se cumpla el criterio de aspiración se vuelven a regenerar movimientos.

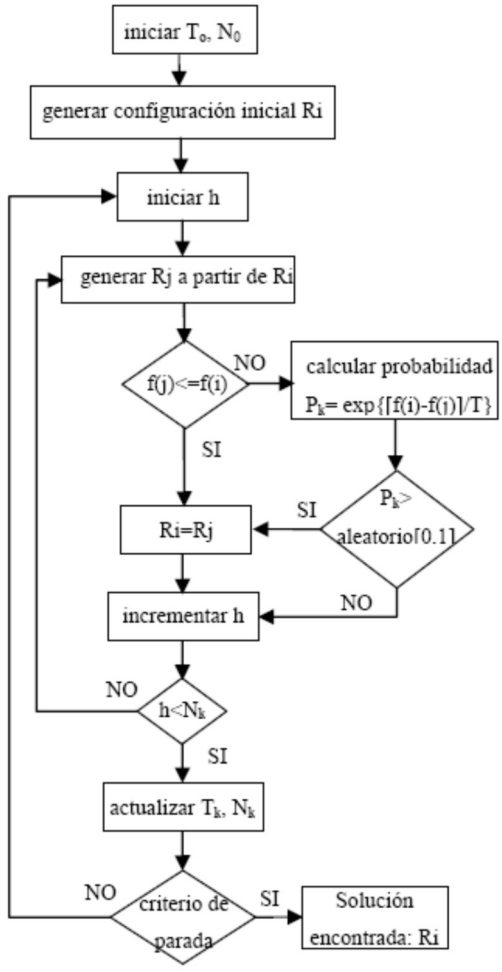


Figura 5. Diagrama del algoritmo Recocido simulado

Búsqueda Tabú es una Meta heurística que usa búsqueda agresiva del óptimo del problema. Lo de Búsqueda Agresiva se refiere a evitar que la búsqueda quede "encajonada" en un óptimo local, en vez de alcanzar al óptimo. En pocas palabras, evitar óptimos locales.

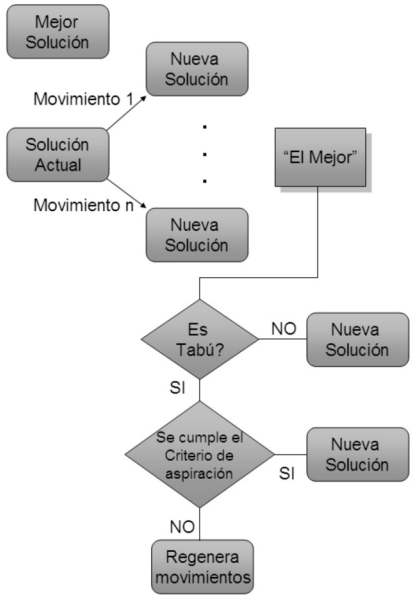


Figura 6. Diagrama del algoritmo de Búsqueda Tabú [17]

**Resultados y discusiones**

Para la evaluación de las tres metaheurísticas se simularon las pruebas para controlar las mismas condiciones sin considerar los tiempos de comunicación, es decir, se realizaron en un solo equipo con un procesador Pentium 4 a una velocidad de 2.93 Ghz con 1 Gb de memoria Ram; así no se usó un ambiente de red e implica que no se llevaron a cabo los procesos, más sin embargo se calcula el tiempo que tarda en planear el acomodo de asignación a los servidores y a su vez la forma del acomodo.

Así se llevaron a cabo con el criterio de paro de 50, 100, 150, 200 y 250 iteraciones con 2000, 4000, 6000 8000 y 10000 número de requerimientos obteniéndose un tiempo promedio que no implicó que el algoritmo Búsqueda Tabú tuviera un tiempo menor. A pesar de ello no representa una diferencia significativa entre las metaheurísticas, como se muestra en la gráfica de la figura 7.


Figura 7. Tiempo promedio en segundos de las metaheurísticas

En cuestión de eficiencia se evalúa mediante la desviación estándar del acomodo de los requerimientos. Mostrándose el siguiente comportamiento en la figura 8, Búsqueda tabú a medida que se incrementan los requerimientos disminuya la desviación estándar promedio, sucede lo contrario con las restantes metaheurísticas.

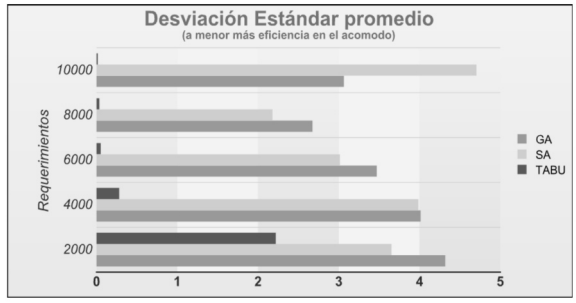


Figura 8. Desviación Estándar Promedio de las Metaheurísticas

**Conclusiones**

Con la visualización de los resultados podemos concluir que la metaheurística que conjuga eficiencia y efectividad por el mejor tiempo promedio y la menor variación es la Búsqueda Tabú. Lo que confirma la literatura de ser una de las mejores metaheurística con su característica fundamental de que proporciona una solución factible muy buena en un tiempo mucho menor que el requerido por un método exacto.

Además el diseño de la arquitectura bajo patrones hace que se involucre un orden en su composición que se apega y puede sustentar diseños más complejos y escalabilidad refiriéndome a aceptar más estrategias de otro tipo.

Aunque quedan trabajos pendientes para generar las condiciones de una red que involucre el tiempo de comunicación y las hetereogenidades de los equipos.

**Referencias**

[1] B. A. Shirazi, A. R. Hurson, and M. K. Kavi. (1995). *Sheduling and Load balancing in Parallel and Distributed Systems*. IEEE Computers Society Press, Los Alamitos, California USA.

[2] O. Ossama, O´R C., S. (April 2001). Designing an Adaptive CORBA Load Balancing Service Using TAO. *IEEE Distributed Systems on Line,* Vol. 2.

[3] CISCO SYSTEMS Inc., High Availability Services, www.cisco.com/warp/public/cc/so/neso/ibso/s390/ mnibm_wp.htm, 2000.

4] A. Goscinski, (1992), *Distributed Operating Systems, The Logical Design,* Addison Wesley.

[5] Common Object Request Broker Architecture (CORBA/IIOP) January 2008 Version 3.1 - Editorial update formal/2008-01-04 (última vez visitado 10/07/2008) http://www.omg.org/technology/documents/corba_spec_catalog.htm

[6] O. Ossama, O´R C., S. (March 2001), Strategies for CORBA Middleware-Based Load Balancing, Vol. 2, Number 3, IEEE *Distributed Systems on Line.*

[7] M. Lindermeier, Load Management for Distributed Object-Oriented Enviroments. In *2nd International Symposium on Distributed Objects and Applications* (DOA 2000), Antwerp, Belguim, Sept. OMG, 2000

[8] A. Puder, K. Roemer, MICO, (Mar 2002), *An Open Source CORBA Implementation, Morgan* Kaufmann, 3rd Edition.

[9] E. Crespo, R. Martí, J. Pacheco; (2007), Procedimientos metaheurísticos en economía y empresa; Rect@ nº 3 *Revista Electrónica de Comunicaciones y Trabajos* ASEPUMA; Madrid.

[10] F. Buschman, et al.; (1996), *A system of patterns, Pattern-Oriented, Software-Architecture.* John Wiley and Sons Ltd.

[11] E. Gamma, H. R., et al. (1995), *Design Patterns: Elements of Reusable Object-Oriented Software.* Addison-Wesley.

[12] E. Curry, CH. D., et al. (2004), Extending Message-Oriented Middleware Using Interception, Proc. *3rd Int'l Workshop on Distributed Event-Based Systems* (DEBS 04), pp. 32–37, Institute of Eng. and Tech.

[13] S. Douglass, S, M., et al. (2000), *Pattern-Oriented Software Architecture, Patterns for Concurrent and Networked Objects.* Volume 2, Chichester: Wiley.

[14] F. K Buschman, H., et al., (2007), Pattern Oriented Software Architecture "A pattern Language for Distributed Computing". Volume 4, Viley.

[15] S. Stelting, y M. O., (2003), *Patrones de diseño aplicados a JAVA.* Primera edición en español, Pearson Educación, S. A, Prentice Hall Sun Microsystems.

[16] D. E. Goldberg, (1989), *Genetic Algorithms in Search, Optimization and Machine Learning, Addison Wesley.*

[17] Sadiq, M. Sait and Habib Youssef; (1999), *Interative Computer Algorithms with Applications in Engineering; Solving Combinatorial Optimization Problems;* IEEE Computer Society.

[18] F. Glover, (1990), "Artificial intelligence, heuristic frameworks an tabu search". *Managerial And Decision Economics,* 11:365-375.

[19] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, M. Protasi; (Springer 2003), *Complexity and Approximation; Combinatorial Optimizaion Problems and Their Approximability Properties.*

[20] J. R. Koza, *Genetic Programming: On the programming of computer by means of natural selection. Cambridge,* MA: MIT Press.

[21] Haupt Randy L. and Haupt Sue E. Practical Genetic Algorithms. Wiley Inter-Science 1998.

[22] Obitko Marek; Genetics Algorithms; Outline of the Basic Genetic Algorithm http://www.obitko.com/tutorials/genetic-algorithms/ga-basic-description.php

[23] Glover, Fred; Melián, Belén; (2003), Búsqueda Tabú; Inteligencia Artificial; *Revista Iberoamericana de Inteligencia Artificial;* No. 19 pp. 29-48; ISSN: 1137-3601; http://www.aepia.org/revista

[24] Russell, Stuart; Norvig, Peter; (1998), *Inteligencia Artificial, un enfoque moderno;* Prentice Hall.

[25] Franco B, John F.; Tabares, Pompilio; (Diciembre de 2005), Aplicación Del Simulated Annealing al Problema de las N Reinas; *Scientia et Technica* Año XI, No 29, UTP. ISSN 0122-1701 publicación: 19 de junio de 2008.

**Artículo recibido:** 9 de febrero de 2009

**Aceptado para publicación:** 22 de mayo de 2009