Distributed System RNA-AG for the Optimization of Design Patterns in Reinforced Concrete Beams

Justo Saico Saico, Br[1], Paul Ventura Acero, Br[1], Richard Tumailla Sanchez, Br[1], Hector Concha, Br[1] and Jose Sulla-Torres, Dr[1]
[1]Universidad Nacional de San Agustín, Perú, jsaico@unsa.edu.pe, jventura@unsa.edu.pe, rtumailla@unsa.edu.pe, hconcha@unsa.edu.pe, jsulla@unsa.edu.pe

Abstract– ***In this paper, a software system for designing reinforced concrete beams (constrained by its bending and shear force) is built, to show results quickly and accurately, using two Artificial Intelligence techniques: Artificial Neural Networks (ANN) and Genetic Algorithms (GA) and a Distributed Systems model: Master-Slave. The calculation of the optimal weights by using distributed parallelism is achieved by Java RMI, sockets and threads. The purpose of the System presented here is to obtain a NN that is capable of relate historic data used for the design of a beam (cantilever beam, reinforcing steel area, stirrup spacing) and that employs empiric design patterns on similar reinforced concrete beams from Escuela Profesional de Ingenieria Civil (EPIC) at Universidad Nacional de San Agustin (UNSA), Arequipa-Peru. The results are shown in diagrams comparing convergence times using our model and finally concluding its superior efectiveness of speed. For example for populations close to a million, the time is acceptable and the error rate is less than 1%.***

Keywords– ***Artificial Neural Networks, Genetic Algorithms, distributed Evolutionary Algorithm, Parallelism, Concurrence, Reinforced Concrete Beam Design.***

Digital Object Identifier (DOI):
http://dx.doi.org/10.18687/LACCEI2019.1.1.169
ISBN: 978-0-9993443-6-1 ISSN: 2414-6390

17th LACCEI International Multi-Conference for Engineering, Education, and Technology: "Industry, Innovation, And Infrastructure for Sustainable Cities and Communities", 24-26 July 2019, Jamaica.

Sistema Distribuido RNA-AG para la Optimización de Patrones de Diseño en Vigas de Concreto Armado

Justo Saico Saico, Br[1], Paul Ventura Acero, Br[1], Richard Tumailla Sanchez, Br[1], Hector Concha, Br[1] and Jose Sulla-Torres, Dr[1]

[1]Universidad Nacional de San Agustín, Perú, jsaico@unsa.edu.pe, jventura@unsa.edu.pe, rtumailla@unsa.edu.pe, hconcha@unsa.edu.pe, jsulla@unsa.edu.pe

Resumen*—En este artículo se construye un sistema capaz de realizar el diseño de una viga de concreto armado, sujeta a flexión normal y corte, el cual proporciona resultados con rapidez y seguridad, usando dos técnicas combinadas de Inteligencia Artificial (IA), Redes Neuronales Artificiales (RNA) y Algoritmos Genéticos (AG), y un modelo distribuido Maestro-Esclavo (MDME) para calcular los pesos óptimos de la RNA usando paralelismo distribuido, implementado en Java con Remote Method Invocation (RMI), Sockets y Threads. El propósito es obtener una RNA capaz de relacionar de forma coherente datos utilizados para el diseño de la viga (peralte de viga, área de acero de refuerzo y espaciamiento de estribos) con los patrones de diseño empírico, de vigas de concreto armado semejantes, de la Escuela Profesional de Ingeniería Civil (EPIC) de la Universidad Nacional de San Agustín (UNSA) de Arequipa. Como resultados se muestra una gráfica comparativa de los tiempos de convergencia de la RNA propuesta, concluyendo su eficacia superior en cuanto a velocidad. Por ejemplo, para poblaciones cercanas al millón, el tiempo es aceptable, y el error es menor a 1%.

***Palabras Clave*—Red Neuronal Artificial, Algoritmo Genético, Computación Evolutiva Distribuida, Paralelismo, Concurrencia, Diseño de Viga de Concreto Armado.**

Abstract*—In this paper, a software system for designing reinforced concrete beams (constrained by its bending and shear force) is built, to show results quickly and accurately, using two Artificial Intelligence techniques: Artificial Neural Networks (ANN) and Genetic Algorithms (GA) and a Distributed Systems model: Master-Slave. The calculation of the optimal weights by using distributed parallelism is achieved by Java RMI, sockets and threads. The purpose of the System presented here is to obtain a NN that is capable of relate historic data used for the design of a beam (cantilever beam, reinforcing steel area, stirrup spacing) and that employs empiric design patterns on similar reinforced concrete beams from Escuela Profesional de Ingenieria Civil (EPIC) at Universidad Nacional de San Agustin (UNSA), Arequipa-Peru. The results are shown in diagrams comparing convergence times using our model and finally concluding its superior efectiveness of speed. For example for populations close to a million, the time is acceptable and the error rate is less than 1%.

***Index Terms*—Artificial Neural Networks, Genetic Algorithms, distributed Evolutionary Algorithm, Parallelism, Concurrence, Reinforced Concrete Beam Design.**

I. Introducción

Los avances en el campo de la Inteligencia Artificial han tenido una fuerte influencia en las diferentes áreas de la Ingeniería. Los nuevos métodos y algoritmos que han aparecido y están apareciendo permiten a los ingenieros usar estas nuevas técnicas de maneras diferentes y sobre problemas de diversa naturaleza [1], [2]. En este artículo se ha aplicado al área de estructuras para el diseño de vigas de concreto armado, aunque la metodología propuesta es aplicable a cualquier ámbito de la ingeniería. La forma en la que se obtienen modelos de funcionamiento es basándose en la experiencia y en la obtención de datos de ensayos. Los expertos intentan extraer el conocimiento de estos datos en forma de ecuaciones y expresiones matemáticas, que, en algunos casos, finalmente se traducen en normativas que regulan los futuros diseños, como en [3]. Este patrón de comportamiento es, en cierta medida, similar al proceso de aprendizaje de un ser humano, y como tal, se pueden aprovechar las técnicas de Inteligencia Artificial que emulan este comportamiento incluyendo técnicas híbridas [4], [5].

Por otro lado, en cualquier problema de ingeniería, la búsqueda por una solución adecuada y rápida es constante, pero esta búsqueda no siempre es una tarea simple e incurre en un costo que debe ser incorporado al costo total del proyecto [3]. Los profesionales de ingeniería estructural están siempre buscando agilizar los procesos de realización de sus proyectos [6]. La aplicación de redes neuronales y algoritmos genéticos a la ingeniería de estructuras es una tentativa de explorar un método alternativo que complemente a los ya existentes [7].

Se ha logrado utilizar ya exitosamente utilizar las RNA en el diseño de vigas de concreto, por lo cual aquí se pretende mejorar el trabajo ya realizado en las investigaciones precedentes. En términos generales, se parte siempre desde un modelamiento que pasa por el diseño de las variables, las restricciones y la función objetivo para finalmente ingresar los datos en el computador [6], [7]. En nuestro caso, implementamos AG en las redes neuronales para volver más efectivas las entradas al sistema en las diversas iteraciones (mejorar el aprendizaje) [8] (Fig. 2). Además, dado que los grandes volúmenes de datos pueden devenir en un costo muy alto en tiempo, se implementa también paralelización para mejorar este aspecto.

En este trabajo se construye un sistema capaz de realizar el diseño de una viga de concreto armado, sujeta a flexión normal y corte, el cual proporciona resultados con rapidez y seguridad, usando 2 técnicas combinadas, redes neuronales y

Digital Object Identifier (DOI):
http://dx.doi.org/10.18687/LACCEI2019.1.1.169
ISBN: 978-0-9993443-6-1 ISSN: 2414-6390

algoritmos genéticos, este último optimizado con un modelo de computación evolutiva distribuida (dEC) maestro-esclavo [9]. El propósito es obtener una RNA capaz de relacionar de forma coherente datos utilizados para el diseño de la viga (Momento, Fuerza de corte, el Esfuerzo máximo de compresión en el concreto, esfuerzo de fluencia para el acero de refuerzo, Ancho de viga) con los resultados del diseño, por lo tanto para entrenar y testear la red neuronal artificial es necesario conocer la solución del diseño (Peralte de Viga, Área de Acero de Refuerzo, Espaciamiento de Estribos) para algunas secciones de vigas de concreto armado (Fig. 1) semejantes.

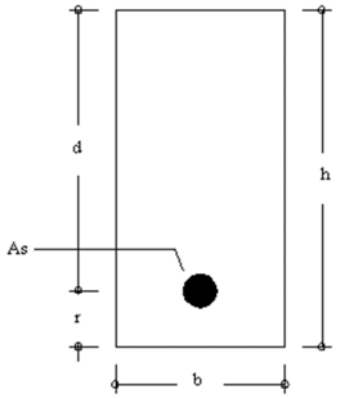

Fig. 1. Viga de Concreto Simplemente Armada

A continuación, el artículo se divide en secciones, como la II donde se mencionan algunos trabajos interesantes relacionados al tema de inteligencia artificial aplicado a construcción de vigas. En la sección III se desarrolla la parte de los materiales usados para la construcción del software, explicamos las diferentes técnicas y tecnologías usadas. En la sección IV se muestran los resultados obtenidos y sus respectivas gráficas de tiempo y validación. Luego la sección V menciona las conclusiones a las que se llegó luego de experimentar la propuesta de RNA-AG paralelizado. En la sección VI de trabajos futuros expresamos nuestro interés por otra técnica de optimización de AE.

II. Trabajos Relacionados

En esta sección se agregan trabajos que influyeron en la realización del presente artículo.

A. Distributed evolutionary algorithms and their models: A survey of the state of the art

En [9] hace una recolección de información de los artículos más relevantes en los temas de Computación Evolutiva y sus diferentes técnicas que pueden ser paralelizadas y distribuidas según los modelos que se clasifican en un *framework* para su desarrollo. También se ven temas de ultima importancia en la investigación como el uso de computación en la nube, CUDA, MapReduce, Hadoop, Big Data, etc. Así como aplicaciones reales y la dirección que siguen estas técnicas en el futuro [8].

B. Design Optimization of Reinforced Concrete Frames

En [7] realiza la optimización del diseño de la estructura de estructura plana de hormigón armado con el fin de minimizar el coste del hormigón y del acero para vigas y columnas adoptando el modelo computacional de Red Neural Artificial (RNA) a través del programa NeuroShell-2 . El procedimiento de diseño se ajusta al Código ACI-318-08. Las variables utilizadas para la optimización del diseño son la anchura, la profundidad y el área de acero reforzado, incluyendo refuerzo longitudinal y refuerzo de corte. Un marco de Concreto Reforzado (RC por sus siglas en inglés) de tres pisos de dos pisos se modela con la selección de diferentes longitudes de tramos y diferentes casos de carga. Los resultados aceptables del diseño se obtienen de más de 50 ejemplos que están sujetos a todas las restricciones del Código ACI, utilizando diferentes secciones transversales y estos resultados se utilizan para entrenar el programa NeuroShell-2. Los resultados obtenidos demuestran la eficiencia del procedimiento de RNA para el diseño de marco RC de múltiples pisos.

C. Optimum detailed design of reinforced concrete continuous beams using Genetic Algorithms

En [3] tenemos un caso representativo de la aplicación de algoritmos evolutivos (en este caso AG) en la solución de un problema de optimización para diseño de vigas de concreto continuas. Cabe resaltar que este problema de optimización se formula como un problema de diseño en lugar de un problema de análisis, de manera que se llega a imitar el procedimiento de diseño tradicional. Otro punto por resaltar es la flexibilidad del sistema propuesto que (al igual que el que se propone en este artículo) permite al diseñador especificar varios tipos de requerimientos de diseño y además también proporciona soluciones realistas y racionales que pueden ser utilizadas directamente en la construcción. Los AG se utilizan para aplicar operadores genéticos con el fin de modificar las poblaciones siempre que los criterios de convergencia no sean logrados. El sistema propuesto por Govindaraj y Ramasami (2005) resalta también la simplicidad matemática del modelo ya que al utilizar metaheurísticas (AG) se libra de las implicaciones formales de un posible modelamiento matemático. La complejidad de su modelo es el detalle técnico respecto al diseño de vigas de concreto armado. Desde el punto de vista computacional, resulta simple (solo utiliza AG) aunque sería difícil definirlo como tal en un análisis general del sistema teniendo en cuenta el número de variables que maneja el sistema y las operaciones que se realizan sobre ellas.

D. Diseño Automatizado De Vigas Rectangulares, Basado en Criterios de Economía y Durabilidad frente al ataque por Cloruros.

En [10] se ha dividido en dos partes claramente diferenciadas. Por un lado, la búsqueda de referencias científicas en las que aparezcan coeficientes de difusión de cloruros para diferentes tipos de cemento, con el objetivo de introducirlos en una red neuronal artificial para conseguir una predicción de ellos y elaborar así una propuesta de tabla. Por otro lado, se desarrolló un programa de optimización estructural de una viga biapoyada, mediante la aplicación de técnicas heurísticas, que la comprueba estructuralmente respecto a la normativa EHE-08 y la optimiza para conseguir un menor coste de realización. A este programa se le incluye los cálculos relativos al estado límite de durabilidad frente al ataque de cloruros, incorporando la nueva tabla de difusión de cloruros propuesta, con el objetivo de analizar los resultados obtenidos y ver cómo se puede alargar la vida útil de estructuras a un coste bajo. Concluyendo respecto al uso de técnicas heurísticas, hay que resaltar la gran aplicación y uso de algoritmos genéticos (AG), la búsqueda por gradiente (DLS) y de la cristalización simulada (SA) en los diferentes trabajos realizados hasta la actualidad. También hay que destacar la gran aplicación de los algoritmos usados recientemente para la optimización de estructuras como la colonia de hormigas (*Ant Colony*), "*Harmony search*" y enjambre de luciérnagas (*Glowworm Swarm*) híbrido. El uso de las redes neuronales en el ámbito de la ingeniería estructural ha sido principalmente para optimizar estructuras como vigas biapoyadas, elementos lineales fabricados con hormigón de alta resistencia y secciones de vigas y pilares. Actualmente se han desarrollado estudios para predecir la longitud de transferencia del pretensado de cordones de acero en piezas de hormigón.

III. Materiales y Métodos

A continuación, se presentan las técnicas y tecnologías utilizadas para la implementación.

A. Algoritmos Genéticos

Los Algoritmos Genéticos (AG) son heurísticas usadas para encontrar un vector x* (una cadena) de parámetros libres con valores en una región admisible para el cual un criterio de calidad arbitraria es optimizado [11].

Para nuestro trabajo, la configuración de la red esta definida en términos del número de entradas y salidas para el numero de neuronas en la capa de entrada y salida respectivamente. No hay un método directo para elegir el número de nodos en capas ocultas para el cual se debe adoptar un método de prueba y error. Los pesos (genes) que representan una solución potencial al problema, son unidos para formar una cadena llamado cromosoma. Cada gen (peso) es un número real. Se utilizará un Algoritmo Genético para ajustar los pesos de las conexiones de la RNA para conseguir que sus salidas sea capaz de converger a las que debería producir. Esta técnica se basa en la utilización de una población de individuos y de tres operaciones genéticas para la obtención de mejores individuos que son capaces de producir resultados mejores que los individuos precedentes. Partiendo de una cierta población, con unos pesos de conexiones se evalúan dichos individuos calculando el Error Cuadrático Medio (ECM) que producen sus salidas comparadas con las que deberían producir. Una vez evaluados, se procede a ordenarlos de menor a mayor ECM (de mejor a peor). Una vez finalizado este proceso de inicialización de la población, se procede a realizar iterativamente la selección, cruce y mutación (operaciones genéticas) sobre dicha población.

Los algoritmos genéticos utilizan tres tipos de operadores: selección, cruce y mutación.

1) *Selección o reproducción*: Este operador escoge cromosomas entre la población para efectuar la reproducción. Cuanto más capaz sea el cromosoma, más veces será seleccionado para reproducirse.

2) *Cruce*: Se trata de un operador cuya labor es elegir un lugar, y cambiar las secuencias antes y después de esa posición entre dos cromosomas, para crear nueva descendencia (por ejemplo, las cadenas 10010011 y 11111010 pueden cruzarse después del tercer lugar para producir la descendencia 10011010 y 11110011). Imita la recombinación biológica entre dos organismos haploides.

3) *Mutación*: Este operador produce variaciones de modo aleatorio en un cromosoma (por ejemplo, la cadena 00011100 puede mutar su segunda posición para dar lugar a la cadena 01011100). La mutación puede darse en cada posición de un bit en una cadena, con una probabilidad, normalmente muy pequeña (por ejemplo 0.001).

Como se ve, los Algoritmos Genéticos difieren de los métodos tradicionales de búsqueda y optimización, en cuatro cuestiones esenciales:

1) Trabajan con un código del conjunto de parámetros, no con el conjunto mismo (necesitan que el conjunto de parámetros del problema de optimización esté codificado en cadenas finitas sobre un determinado alfabeto). Por trabajar a nivel de código, y no con las funciones y sus variables de control, como los otros métodos, son más difíciles de "engañar".

2) Buscan una población de puntos, no un único punto. Manteniendo una población de puntos muestrales bien adaptados, se reduce la probabilidad de caer en una cima falsa.

3) Emplean la función objetivo, no necesitan derivadas ni otra información complementaria, tan difícil a veces de conseguir. De este modo ganan en eficiencia y en generalidad.

4) Se valen de reglas de transición estocásticas, no deterministas. Los Algoritmos Genéticos se valen de operadores aleatorios para guiar la búsqueda de los mejores puntos; puede parecer extraño, pero la Naturaleza está llena de precedentes al respecto.

B. Redes Neuronales Artificiales

Las redes neuronales artificiales (RNA) son una familia de modelos inspirados en redes neuronales biológicas y se utilizan para aproximar funciones que pueden depender de un gran número de insumos y generalmente son desconocidas. Las RNA se presentan como sistemas de "neuronas" interconectadas que intercambian mensajes entre sí. Las conexiones tienen pesos que pueden sintonizarse, haciendo redes neurales adaptables a los insumos y capaces de aprender. La capacidad de las RNA para aproximar con precisión las funciones desconocidas. de ahí su gran utilidad [12].

En nuestro trabajo, la Selección de la Entrada y Salida de la RNA está dado por el momento ultimo de Diseño, Fuerza cortante en la parte critica de la viga, ancho de sección, grado de fluencia acero y resistencia a la compresión del concreto, el modelo será capaz de predecir los valores de peralte, área de refuerzo y espaciamiento de estribos. Por lo tanto, la entrada a la red es: El vector de entrada para este modelo esta dado por: $I = M, V, Fc, Fy, bw$

El diseñador desea conocer:

- Peralte efectivo de la viga(h)
- Área de la sección transversal del acero refuerzo (As)
- Espaciamiento del acero transversal (s)

El vector de salida para el modelo es:

$O = h, As, S$

En la Fig. 2 se muestra la interacción de la RNA con el AG en un modelo híbrido.

C. Taxonomía de los modelos de Computación Evolutiva Distribuida: Modelo Maestro-Esclavo

Dentro de la taxonomía de la computación evolutiva encontramos que se divide en 2 tipos (Fig. 3), según divide las tareas de cálculo [9].

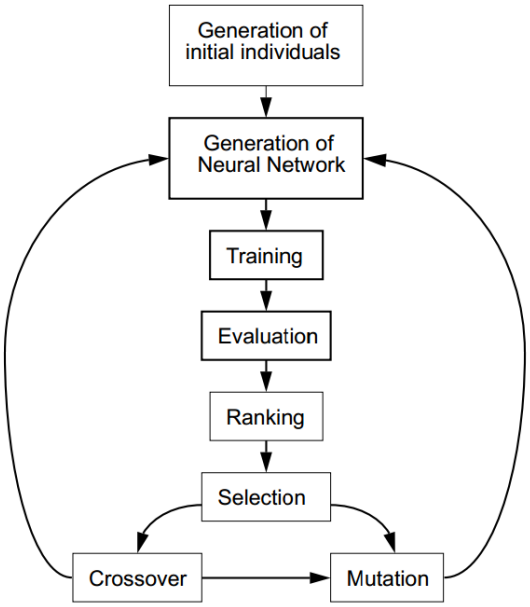

Fig. 2. Modelo hibrido: Red Neuronal Artificial-Algoritmo Genético [8]

1) Distribuido por la población: Distribuye a los individuos de la población (o subpoblaciones) a múltiples procesadores o nodos de computación.

2) Distribuido por las dimensiones: Distribuye particiones de las dimensiones del problema (o subespacios)

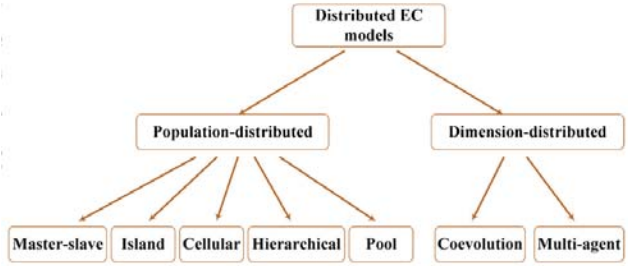

Fig. 3. Taxonomía de modelos de Computación Evolutiva Distribuida [9]

El modelo maestro-esclavo resume un enfoque distribuido a las operaciones de EA y evaluaciones de dominio como se ilustra en la Fig. 4. El maestro realiza las operaciones de cruce, mutación y selección, pero envía individuos a los esclavos para evaluaciones de la estabilidad, ya que éstos constituyen la mayoría de la carga de cálculo. Como las evaluaciones de los individuos son mutuamente independientes, no existe un requisito de comunicación entre los esclavos. El modelo maestro-esclavo es por lo tanto simple, en el cual las comunicaciones sólo ocurren cuando el maestro único envía individuos a los esclavos y los esclavos devuelven los valores de la propiedad correspondiente al maestro en cada generación [9].

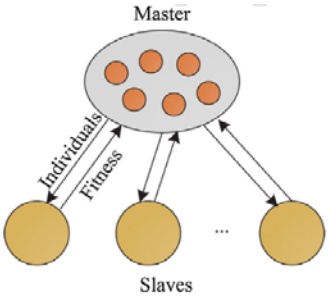

Fig. 4. Modelo Maestro-Esclavo [9]

D. Modelo Distribuido

Para el diseño de nuestro proyecto usamos de referencia el *framework* que encontramos en [9] cuyo resumen se puede observar en la Fig. 5, del elegimos la siguiente configuración:

1) *Algoritmos Evolutivos*: Algoritmos Genéticos.
2) *Modelo Distribuido*: modelo Maestro-Esclavo.
3) *Entorno de Programación*: Java threads y RMI.
4) *Plataforma Física*: Agrupamiento en Red Local.

El algoritmo RNA-AG será alojado en un servidor master para poder ser accedido por múltiples clientes a la vez (múltiples ejecuciones). Esto se realizará mediante una arquitectura *thread-per-request* (un hilo por cada petición), lo cual permitirá incrementar (teóricamente) hasta en 5 veces la velocidad de respuesta del sistema, siendo de todas formas limitado por el tiempo de respuesta del sistema I/O [13]. Esto será muy útil para realizar análisis de sensibilidad (simulación) donde se tenga que realizar múltiples cálculos variando exhaustivamente los parámetros de entrada.

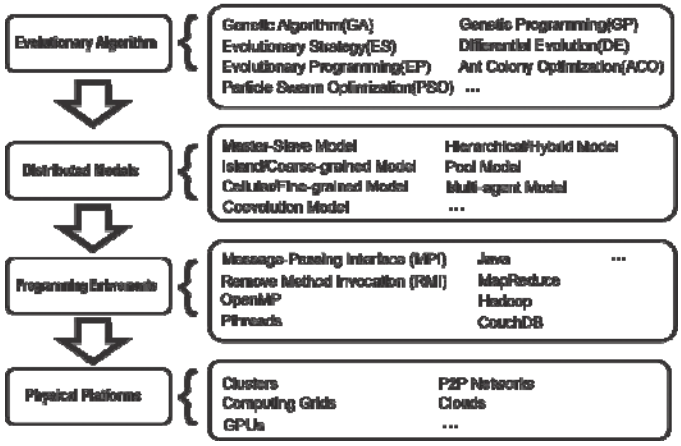

Fig. 5. Framework de Computación Evolutiva Distribuida [9]

La implementación de este modelo se puede realizar en Java principalmente de dos formas:

1) Mediante sus funcionalidades prestablecidas para *threads* y *sockets* (Fig. 6)
2) Mediante invocación a métodos remotos (RMI) (Fig. 7). A continuación, se explica cómo es que se aplica estas herramientas para distribuir (paralelizar) las múltiples ejecuciones de nuestro algoritmo.

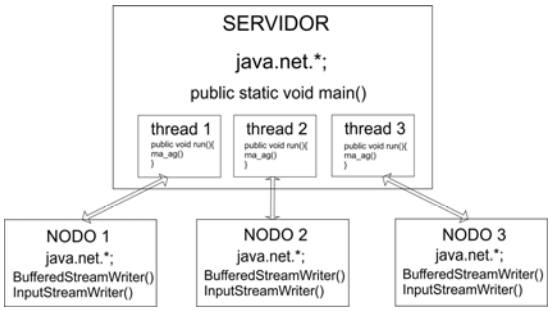

Fig. 6. Arquitectura thread-per-request utilizando Java threads y sockets (Fuente propia)

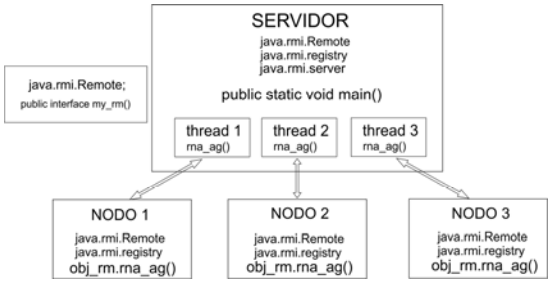

Fig. 7. Arquitectura Thread-per-request utilizando RMI (Fuente propia)

E. Java RMI

En RMI para crear un servicio remoto es necesario definir una interfaz que derive de la interfaz *Remote*. Esta interfaz deberá contener los métodos requeridos por ese servicio, especificando en cada uno de ellos que pueden activar la excepción *RemoteException*. La ventaja de utilizar RMI es que maneja el *multithreading* automáticamente sin necesidad de que el usuario sea el que defina los parámetros de conexión y demás (RMI *Registry*). Simplemente accedemos a nuestro método principal como si fuera local. La única particularidad de este modelo es crear una interfaz donde se listen los métodos que podrán ser invocados remotamente, en nuestro caso, el algoritmo RNA-AG [13].

F. Java Sockets

El paquete java.net provee dos clases (*Socket y Server-Socket*) que implementan el lado del cliente y del servidor de la conexión, respectivamente. Cuando creamos los sockets estamos definiendo una conexión de uno a uno (TCP/IP), de modo que para trabajar con varios nodos tenemos que paralelizar nuestro trabajo, debiendo así utilizar también múltiples hilos (uno para cada conexión). Además de ello también tenemos que instanciar objetos de las clases *BufferedInputStream*, *InputStreamReader*, *BufferedOutputStream* y *OutputStreamWriter* para el envío de

mensajes. Para enviar/recibir objetos necesitamos también instanciar *ObjectInputStream* y *ObjectOutputStream* [13].

G. Java Threads

Con Java creamos un bucle *while* para recibir todas las peticiones y correr un hilo para cada una de ellas. Las rutinas a ejecutarse se encuentran en el método *run*() de nuestra clase servidor. El paralelismo logrado puede tener múltiples beneficios. Permite acceder al servicio a varios usuarios desde cualquier punto en que pueda tener acceso a la red. Se puede también explotar la capacidad de procesamiento del servidor mediante múltiples instancias del problema que por lo general tomaría incluso horas de acuerdo con la dimensión de nuestros datos de entrada [13].

H. Datos Usados: Patrones de Diseño Entrada y Salida

En las siguientes tablas se muestra el conjunto de datos empíricos en forma de patrones entrada (Tabla I) y salida (Tabla II) de diseño de una viga, proporcionados por la Escuela Profesional de Ingeniería Civil (EPIC) de la UNSA, que sirven para entrenar nuestra RNA. Cabe señalar que se muestran los primeros 10 datos de un total de 300.

IV. Resultados

En esta sección se mostrará la utilidad de nuestra implementación, comenzando con los recuerdos de hardware que se dispone y la medición del tiempo de convergencia paralelizado y no paralelizado, además se incluye una gráfica para comparar que tan cerca los valores de la RNA se aproximan a los valores experimentales del experto.

TABLA I
Patrones De Entrada Para El Diseño De Vigas (EPIC-UNSA)

n	Momento (kN - M)	Cortante kN	$f'c$ Concreto (N/mm^2)	Grado de acero (N/mm^2)	Ancho de viga (mm)
1	30	70	20	250	250
2	40	80	25	415	300
3	50	80	20	415	350
4	60	100	20	250	300
5	100	120	25	415	300
6	70	90	20	250	300
7	75	55	25	500	300
8	90	60	20	500	300
9	80	35	25	250	320
10	45	50	25	250	300

TABLA II
Patrones De Salida Para El Diseño De Vigas (EPIC-UNSA)

n	Peralte (mm)	Área Sección Transversal Acero (mm^2)	Espaciamiento de Estribos (mm)
1	220	878.8	75
2	215	698.47	95
3	250	768.92	150
4	280	1370.92	75
5	330	1110.39	110
6	300	1476.38	100
7	300	792.54	210
8	360	769.896	255
9	280	1827.9	95
10	220	1218.19	100

A. Descripción del Hardware

Para nuestro trabajo, las pruebas se realizaron utilizando una PC que sirvió como servidor local con las siguientes especificaciones:

- Procesador Intel(R) Core(TM) i5-2400 CPU @3.10GHz
- Memoria RAM 4.00 GB
- Sistema operativo Windows 10 x64

B. Gráficas comparativas

En la Fig. 8, Tiempo en función de las Época de población se puede observar como hay una ventaja considerable, al utilizar la técnica de paralelización en algoritmos genéticos, en la Red Neuronal Artificial ya que vemos que, para una cantidad de población muy grande, cercana al medio millón sucede que el tiempo de convergencia es considerablemente mejor (más rapidez) que si lo comparamos con la técnica original sin paralelizar (considerablemente lento).

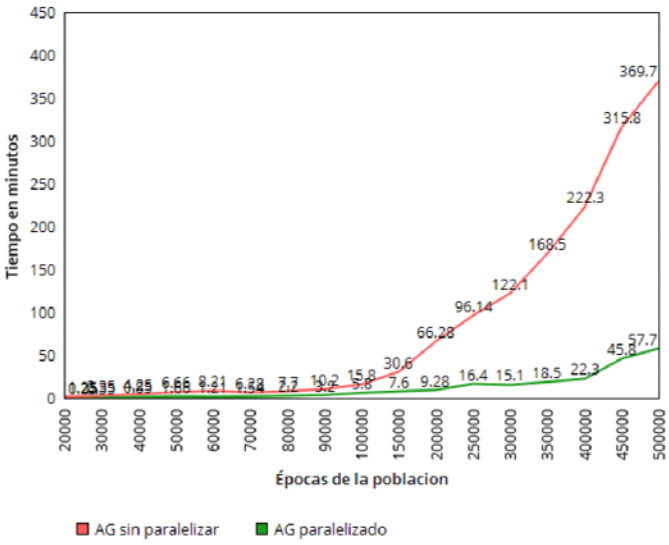

Fig. 8. Tiempo en función de las épocas de la población para el RNA-AG paralelizado

En la Fig. 9 Validación para el acero, representa los valores estimados por nuestro modelo distribuido y los valores

Digital Object Identifier: (to be inserted by LACCEI).
ISSN, ISBN: (to be inserted by LACCEI).

obtenidos en el laboratorio durante la experimentación por el experto, aquí se muestra claramente que nuestro sistema distribuido se asemeja bastante, su acercamiento dependerá de establecer el erro de convergencia.

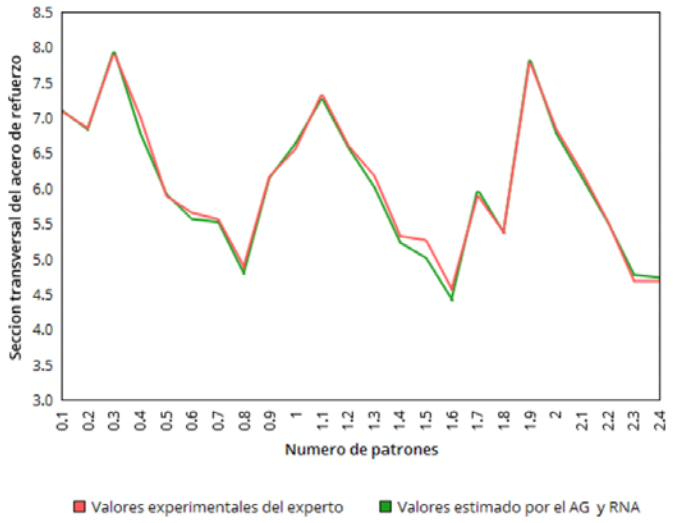

Fig. 9. Validación para el acero del refuerzo de las vigas

V. Conclusiones

En este artículo se implementó una herramienta capaz de resolver la tediosa tarea de calcular el diseño de viga óptimo para determinados parámetros de entrada. Con el RNA-AG paralelizado y distribuido de tareas, se mejora la convergencia de la red neuronal gracias a los algoritmos evolutivos, reduciendo el error entre lo esperado y lo calculado a menos de 1% de manera que el modelo final resulta en una alternativa eficaz en tiempo para el diseño de vigas de concreto armado.

VI. TRABAJOS FUTUROS

Como trabajo futuro se planea implementar la RNA con la técnica de Optimización por Enjambre de Partículas ya que esta técnica muestra gran desempeño en diversas áreas, y también al ser aplicado a vigas doblemente reforzadas. Se puede extender las funcionalidades para hacer simulaciones en tiempo real del diseño de viga, para su uso en análisis de sensibilidad.

Referencias

[1] A. Blanco, M. Delgado, and M. C. Pegalajar, "A genetic algorithm to obtain the optimal recurrent neural network," *Int. J. Approx. Reason.*, 2000.

[2] P. Tahmasebi and A. Hezarkhani, "A hybrid neural networks-fuzzy logic-genetic algorithm for grade estimation," *Comput. Geosci.*, 2012.

[3] V. Govindaraj and J. V. Ramasamy, "Optimum detailed design of reinforced concrete continuous beams using Genetic Algorithms," *Comput. Struct.*, 2005.

[4] J. Yang, K. T. Chan, T. Dai, F. W. Yu, and L. Chen, "Hybrid Artificial Neural Network-Genetic Algorithm Technique for Condensing Temperature Control of Air-Cooled Chillers," in *Procedia Engineering*, 2015.

[5] A. Saleema and T. Amarunnishad, "A New Steganography Algorithm Using Hybrid Fuzzy Neural Networks," *Procedia Technol.*, 2016.

[6] A. A. A. Aga and F. M. Adam, "Design Optimization of Reinforced Concrete Frames," *Open J. Civ. Eng.*, 2015.

[7] F. M. Adam, A. E. Mohamed, S. A. Babiker, and others, "Design Optimization of reinforced concrete beams using artificial neural network," 2012.

[8] P. Koehn, "Combining Genetic Algorithms and Neural Networks : The Encoding Problem," *Thesis*, 1994.

[9] Y. J. Gong *et al.*, "Distributed evolutionary algorithms and their models: A survey of the state-of-the-art," *Applied Soft Computing Journal*. 2015.

[10] E. C. García, "Diseño automatizado de vigas rectangulares, basado en criterios de economía y durabilidad frente al ataque por cloruros," 2016.

[11] E. Alba and J. M. Troya, "A survey of parallel distributed genetic algorithms," *Complexity*, 1999.

[12] H. Demuth and M. Beale, *Neural Network Toolbox*. 2013.

[13] G. F. Coulouris, J. Dollimore, and T. Kindberg, *Distributed systems: concepts and design*. pearson education, 2005.