SINTONIZACIÓN AUTOMÁTICA DE REDES NEURONALES SOM APLICANDO ALGORITMOS EVOLUTIVOS

Por:

LEONARDO ANDRES PIRELA MUÑOZ
YUDELMAN ELOY CARRILLO RODRÍGUEZ

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERIAS FISICOMECANICAS
ESCUELA DE INGENIERIAS ELECTRICA ELECTRONICA Y DE TELECOMUNICACIONES
BUCARAMANGA
2011

SINTONIZACIÓN AUTOMÁTICA DE REDES NEURONALES SOM APLICANDO ALGORITMOS EVOLUTIVOS

LEONARDO ANDRES PIRELA MUÑOZ
YUDELMAN ELOY CARRILLO RODRÍGUEZ

Trabajo de grado presentado como requisito para optar al título de ingeniero electrónico

Director:
PhD. RODOLFO VILLAMIZAR

Codirector:
MSc. JHONATAN CAMACHO

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERIAS FISICOMECANICAS
ESCUELA DE INGENIERIAS ELECTRICA ELECTRONICA Y DE TELECOMUNICACIONES
BUCARAMANGA
2011

CONTENIDO

Pág.

LISTA DE TABLAS

Pág.

LISTA DE FIGURAS

Pág.

ANEXOS

Pág.

RESUMEN

TITULO: SINTONIZACION AUTOMATICA DE REDES NEURONALES SOM APLICANDO ALGORITMOS EVOLUTIVOS[1]

AUTORES: Leonardo Andres Pirela Muñoz, Yudelman Eloy Carrillo Rodríguez[2]

PALABRAS CLAVES: Monitorización de Salud Estructural, Razonamiento Basado en Casos, Algoritmo diferencial evolutivo, Redes neuronales SOM

El presente artículo constituye un aporte significativo al problema de la Monitorización de Salud Estructural (MSE), abordado mediante razonamiento basado en casos (CBR) y usando una red neuronal tipo SOM (Self Organizing Map). Debido a la gran cantidad de parámetros que se deben definir durante el entrenamiento de la red, la eficiencia computacional del sistema experto que detecte, localice y cuantifique cambios porcentuales de rigidez en una estructura civil depende en gran medida de la selección adecuada de dichos parámetros. Por tanto este procedimiento, se plantea en este proyecto, mediante el uso de un algoritmo diferencial evolutivo que permita seleccionar de manera automática los parámetros adecuados requeridos para el entrenamiento de la red SOM.

Asimismo, se especificaron indicadores apropiados de desempeño de la red, con los que se evaluó la calidad del entrenamiento tendiente a aumentar la confiabilidad del diagnóstico realizado y facilitar a su vez la interpretación de los resultados obtenidos del sistema experto desarrollado.

Finalmente, se logró la implementación numérica del algoritmo computacional a una estructura en estudio, donde los patrones de daño usados para el proceso de entrenamiento, se obtuvieron de un modelo estructural tipo *Benchmark* reportado en la literatura. La implementación del algoritmo, permitió observar la disminución de los errores de identificación en contraste con los adquiridos al seleccionar los parámetros de entrenamiento de la red de manera sugerida en [6].

---

[1]Proyecto de grado desarrollado en la modalidad de investigación
[2]Facultad de ingenierías Fisico-Mécanicas, Escuela de Ingenierias Eléctrica Electrónica y telecomunicaciones Director: PhD. Rodolfo Villamizar. Co-director: MSc. Jhonatan Camacho

ABSTRACT

TITLE: AUTOMATIC TUNING OF NEURAL NETWORK SOM BY USING EVOLUTIONARY ALGORITHMS[1]

AUTHORS: Leonardo Andres Pirela Muñoz, Yudelman Eloy Carrillo Rodríguez[2]

KEYWORDS: Structural Health Monitoring, Case-Based-Reasoning, Differential Evolutionary Algorithm, neural network.

This project is a contribution to the Structural Health Monitoring (SSM) problem, solved by using case-based reasoning (CBR) with a neural network SOM (Self Organizing Map). Because large number of parameters neccesary to be defined for training purposes, computational efficiency for the expert system able to detect, locate and quantify stiffness percentage changes in a civil engineering structure, depends on a propriated selection. Therefore, a differential evolution algorithm to automatically select training parameters for the SOM network, was proposed to be used in this project.

Also, appropriated indicators to evaluate network training quality were proposed in order to increase diagnosis reliability and to interpret most adequately results from the developed expert system.

Finally, the proposed computational algorithm was numerically implemented for a study structure, where damage patterns used for training were obtained from a Benchmark structural model reported in literature. Algorithm implementation showed a decreasing of identification errors compared to those obtained by selecting manually network training parameters as suggested in [6]. Addition, the algorithm found the parameters that are most influential in a good training of the network (SOM) as the number of clusters and the function of neighborhood.

[1]Degree´s Project developed for investigation purposes
[2]Physical-Mechanical Faculty, Electric, Electronic and Telecomunications department, Director: PhD. Rodolfo Villamizar. Co-director: MSc. Jhonatan Camacho

UNIVERSIDAD INDUSTRIAL DE SANTANDER
ESCUELA DE INGENIERÍAS ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES

Sintonización automática de redes neuronales SOM aplicando algoritmos evolutivos

Leonardo Pirela
Universidad Industrial de Santander
Grupo CEMOS
Bucaramanga, Colombia
leanpimu@hotmail.com

Yudelman Carrillo
Universidad Industrial de Santander
Grupo CEMOS
Bucaramanga, Colombia
yudelman@hotmail.com

Rodolfo Villamizar
Universidad Industrial de Santander
Grupo CEMOS
Bucaramanga, Colombia
rovillam@uis.edu.co

Jhonatan Camacho
Universidad Industrial de Santander
Grupo CEMOS
Bucaramanga, Colombia
jhonatan_uis@hotmail.com

Resumen **– El presente artículo es un aporte al problema de la Monitorización de Salud Estructural abordado mediante razonamiento basado en casos (CBR), donde el almacenamiento de casos y el entrenamiento para inferir daños se realiza de manera evolutiva por medio de algoritmos diferenciales. Esto permite el fácil entrenamiento de un sistema experto programado para la detección, localización y cuantificación de cambios porcentuales de rigidez en una estructura. Se validó numéricamente el algoritmo, sintonizado de manera automática, con el que se obtuvo una mejora significativa en los resultados en contraste con los adquiridos al seleccionar los parámetros de entrenamiento de la red neuronal de forma manual.**

Índice de **Términos –Monitorización de Salud Estructural, Razonamiento Basado en Casos, Algoritmo Diferencial evolutivo.**

Abstract - This paper is a contribution to the Structural Health Monitoring problem by using case-based reasoning (CBR), where storage case and training in order to infer damage, the training is conducted in evolutionary way by using differential algorithms. Then a low cost programming expert system for detection, localization and quantification of stiffness changes in a structure can be obtained. A numerical validation of automatically tuned algorithm was achieved, where a meaning improvement, compared with the manually tuned neural network case, was obtained.

Keywords**– Structural Health Monitoring, Case-Based-Reasoning, Differential Evolutionary Algorithm.**

I. INTRODUCCIÓN

Uno de los objetivos de la monitorización de Salud Estructural (Structural Health Monitoring o SHM), es el de evaluar en línea las condiciones dinámicas de las estructuras con el fin de determinar, localizar y cuantificar daños en la misma, además de predecir su vida útil [1]. Es por esto que se han venido desarrollando metodologías de supervisión experta [2], [3] que combinando técnicas para el procesamiento digital de señales y técnicas de inteligencia artificial con las que es posible identificar cambios porcentuales de rigidez en una estructura. Para este proyecto se utilizó la metodología presentada en [6], basada en la combinación de la transformada wavelet discreta (DWT), análisis de componentes principales (PCA), razonamiento basado en casos (CBR) y las redes neuronales SOM (Self Organizing Map). Sin embargo, debido a la gran cantidad de parámetros que se requieren definir durante el entrenamiento de la red neuronal SOM, el proceso para obtener un sistema eficiente requiere más alta experticia algo de empirismo y sus resultados al parecer no hay favorables, Por

tanto, en este artículo se aborda el problema mediante el uso de un algoritmo diferencial evolutivo que selecciona de manera automática los parámetros de entrenamiento asociados a la red SOM, lográndose una disminución significativa de los errores de identificación, obtenidos en trabajos previos [6]. Asimismo, para aumentar la confiabilidad del proceso de entrenamiento y validación de la red neuronal SOM, se definieron algunos indicadores numéricos que permitieron analizar la evolución del algoritmo.

II. SISTEMA DE MONITORIZACIÓN ESTRUCTURAL

En el presente artículo se aborda el problema de monitorización de Salud Estructural (MSE), nivel III, la cual comprende la detección, localización y cuantificación de daños. Donde, el daño es asociado a cambios de rigidez en los elementos de una estructura [6]. Los datos de la respuesta dinámica de la estructura se obtuvó de un problema *Benchmark* de la literatura, donde se encuentran disponibles modelos de elementos finitos y el software en MATLAB® para la generación de la respuesta de aceleración.
De acuerdo a las necesidades del programa para MSE, se desarrolla un modelo analítico de la estructura, cuyos parámetros generalmente son obtenidos mediante el método de elementos finitos. El modelo matemático corresponde a la ecuación diferencial de movimiento que representa el comportamiento dinámico de la estructura debido a excitaciones externas y condiciones internas tales como vientos, sismos, cargas, condiciones normales de operación, humedad, temperatura, esfuerzo y envejecimiento.

$$M\ddot{x} + C\dot{x} + Kx = F \qquad Ec.1$$

Donde **M** es la matriz diagonal de masas, **C** es la matriz de amortiguamientos, **K** es la matriz de rigidez, **F** es el vector de fuerzas aplicado en la estructura y es el vector con los desplazamientos de cada uno de los elementos de la estructura respecto a un marco de referencia inercial. Luego que el modelo analítico de la estructura se ha establecido, se realiza un ajuste del mismo utilizando información de los modos de vibración de la estructura proveniente de registros de velocidad, desplazamiento ó aceleración ambiental. Para lograr esta tarea, se instrumenta la estructura mediante sensores de aceleración localizados en elementos que aporten información significativa. [6].

A. Algoritmo de monitorización utilizado

Para la solución del problema de identificación de daños en la estructura, el algoritmo de supervisión utilizado es el desarrollado en [6] y presentado en la figura 1. Consiste en: i) Aplicar la Transformada *Wavelet Discreta (DWT)* a las señales provenientes de sensores de aceleración ubicados en elementos de la estructura, para obtener el vector de características originales. Este vector es conformado por los coeficientes de detalle y aproximación obtenidos en cada nivel de resolución del análisis *wavelet* realizado [6]. En esta etapa, se obtiene el patrón que representa las variaciones de rigidez en la estructura, sin embargo, el vector de características principales corresponde a vectores con alta dimensionalidad e información poco organizada siendo necesario reducir el tamaño del vector mediante técnicas que permitan conservar y organizar la información relevante. La reducción de dimensionalidad y selección de características se puede lograr mediante la implementación de métodos por proyección, como el análisis de componentes principales (*ACP*) [6] (ver figura 2). ii) Mediante *ACP* se obtienen descriptores a través de la combinación lineal de las variables originales. Estos nuevos factores describen las tendencias en el conjunto original de datos. Así, la aplicación de *ACP* busca encontrar las variables de mayor significancia estadística. iii). Luego mediante Razonamiento Basado en Casos (CBR), se obtiene un diagnóstico inicial por analogía. Para este último propósito, se entrena una red SOM como herramienta de clasificación, la cual permite organizar casos antiguos en memoria mejorando la velocidad del razonamiento [6].

B. Construcción de la base de casos

La base de casos es un arreglo en memoria, en donde se organiza el total de simulaciones que describen la respuesta dinámica de todos los patrones de daño bajo estudio [6]. Esta permite facilitar la búsqueda de los casos más similares a un nuevo patrón de daño, de tal forma que sea posible identificarlo.
En la metodología utilizada, la base de casos es un mapa auto-organizado, donde cada caso está definido por el daño en la estructura y la mínima representación de su respuesta dinámica. De esta

forma, la mínima representación es el conjunto de características principales que son extraídas mediante DWT y PCA aplicados a los registros temporales de aceleración. La Figura 1 describe el proceso de extracción de características y la construcción de la base de casos.

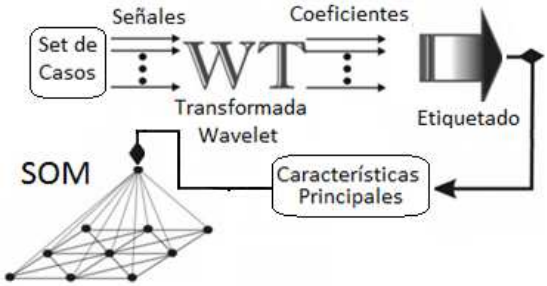

Figura 1. Construcción de la base de casos. Modificado [6]

Una vez obtenido el conjunto de características principales, se crea y entrena una red SOM. Esta facilita la búsqueda de patrones de daño y mediante un elemento intérprete se identifican nuevos casos [7].

C. Cálculo índices de error

Para la metodología de identificación de fallas se tuvieron en cuenta los siguientes indicadores de error:

- Error de dimensión: La dimensión indica la cantidad de elementos con daño en un caso en particular. El error de dimensión está definido por:

$$\%(error_{Dim}) = \left|\frac{Dim_{Estimada} - Dim_{Real}}{Dim_{máxima}}\right| \times 100 \qquad Ec\ (2)$$

Donde Dimensión máxima es igual a 4

- El error de severidad: La severidad indica el cambio porcentual de la rigidez en cada elemento respecto al estado nominal y se calcula como el promedio de los errores de cada elemento de la estructura. Este error puede ser evaluado por zonas de interés conforme la disposición geométrica de la estructura. Dicho error es calculado sobre los n elementos con mayor probabilidad de daño. El error de severidad está definido por:

$$\%(error_{Sev}) = \sum_{i=1}^{n}\left|\frac{Sev_{Esti}(i) - Sev_{Real}(i)}{Sev_{Real}(i)}\right|/n \times 100 \qquad Ec\ (3)$$

- Error de falsos positivos: Un falso positivo se presenta cuando el sistema experto detecta un daño, cuando realmente debería informar que la estructura no tiene daño en ningún elemento Ecuación (4).

$$\%(error_{FP}) = \frac{\#Falsos\ Positivos}{\#Casos\ sin\ daño} \times 100 \qquad Ec\ (4)$$

- Error de falsos negativos: Un falso negativo sucede cuando el algoritmo de supervisión detecta que no hay daño en algún elemento de la estructura, cuando realmente el nuevo caso relaciona la existencia de daño en algún elemento de la estructura. Ecuación (5)

$$\%(error_{FN}) = \frac{\#Falsos\ Negativos}{\#Casos\ con\ daño} \times 100 \qquad Ec\ (5)$$

- Error de tipo: Corresponde al número de elementos que fueron clasificados erróneamente en un tipo específico (columna, viga o brazo).

- Error de piso: Corresponde a la cantidad de elementos clasificados erróneamente en un piso específico (primero, segundo, tercero o cuarto)

- Error del elemento: Este error indica el elemento con daño en un caso en particular.

D. Validación cruzada

Para estudiar los efectos del elemento intérprete cuando se evalúa sobre casos de entrenamiento y operación, se aplicó validación cruzada y búsqueda en malla. Esta técnica es comúnmente utilizada en la inteligencia artificial para validar los modelos generados a partir de un conjunto de datos. En el proceso de entrenamiento se debe considerar, tanto el error de aprendizaje, como el error de generalización para no incurrir en sobre aprendizajes o sobre ajustes (overfitting). Para minimizar la posibilidad de que exista una tendencia a memorizar los ejemplos de todo el conjunto de entrenamiento se emplea aproximadamente un 80% de los patrones para entrenar, reservándose un 20% como conjunto de test. El tipo de validación cruzada comúnmente usado emplea el procedimiento (*leave-k-out*) [14], que consiste en dividir aleatoriamente el conjunto de casos de entrenamiento en k grupos. Luego,

sobre k-1 grupos se aplica la metodología CBR y se evalúa el error de identificación para el grupo que no fue tenido en cuenta. En suma, la validación cruzada es una técnica que se utiliza para tratar problemas iterativos repartiendo la muestra en dos sistemas de datos, uno se utiliza para construir el modelo, y el otro para probarlo.

III. SINTONIZACION AUTOMÁTICA DE PARAMETROS DE LA RED SOM.

El presente artículo está enfocado en la búsqueda automática de los parámetros óptimos para el entrenamiento de una red SOM, mediante el uso de un algoritmo evolutivo diferencial (ED). El entrenamiento de esta red se realizó teniendo en cuenta los siguientes parámetros:

- Método de normalización: Especifica la operación de normalización de los datos de entrada. Corresponde a: Varianza, Rango lineal, logarítmico, logístico.

- Número de neuronas de salida: Representa el número de *clusters*, limitado entre $5\sqrt{n}$ y $\frac{9}{10}n$, donde *n* es la cantidad de casos de entrenamiento. El número de neuronas determina la suavidad de la proyección, lo que influye en el ajuste y capacidad de generalización de la SOM. [6], [11].

- Estructura de la grilla: Indica el tipo de topología local del mapa, puede ser Rectangular ó Hexagonal.

- Forma del mapa: Indica el tipo de topología global del mapa. Puede ser: Laminar, Cilíndrico ó Toroidal.

- Función de vecindad: Describe la interacción de los vectores de referencia m_j y m_c durante la adaptación y es a menudo una función del tiempo t, la función de vecindad y el número de neuronas determina el nivel de detalle de la cartografía resultante. Cuanto más grande es el área donde la función vecindad tiene elevados valores, más rígido es el mapa. Cuanto más grande sea el mapa, más flexible puede llegar a ser. Esta interacción determina la precisión y capacidad de generalización de la SOM. puede ser *Gaussiana*, *Gaussiana* recortada, Burbuja ó Exponencial

La evaluación de la calidad de la red construida se realizó mediante la obtención de los siguientes indicadores:

- Error topográfico: La medida de preservación de la topología describe la manera en la que la SOM preserva la topología del conjunto de dato, la que considera la estructura del mapa [14]. Se debe considerar que la vecindad se mantengae idealmente este error debe ser cercano a cero.

- Medida de distorsión: Calcula la medida de distorsión media para cada unidad del mapa. Supongase que x $\in R^n$ es el vector de entrada y el $m_j \in R^n$, j$\in$ (índices de las neuronas) son los vectores de referencia, sea d(x, m_j) define una función distancia generalizada de x, y m_j la medida distorsión ε se define en la ecuación (6):

$$\varepsilon = \sum_{i \in L} h_{ci} d(x, m_j) \qquad Ec(6)$$

Donde la función vecindad h_{ci} describe la interacción de los vectores de referencia, L denota el conjunto de los índices de todas las unidades de la red, esta ecuación se define como una suma de funciones de distancia ponderada por h_{ci}, por lo que *c* es el índice del vector más cercano al vector referencia o *codebook*. La distorsión media se puede obtener integrando la ecuación (6) para obtener la ecuación (7).

$$E = \int \sum_{i \in L} h_{ci} d(x, m_j) \qquad Ec(7)$$

Una manera de definir la SOM es definirla como el conjunto de las m_j que globalmente minimiza E. [11]

- Uniformidad del histograma: Mide la distribución de casos en cada clúster. Para la metodología propuesta, idealmente cada clúster debería tener casos de daño en un solo elemento (ó de una zona geométrica de la estructura), no deberían existir *clusters* vacíos y todos los *clusters* deberían tener igual cantidad de casos.

A. Algoritmo diferencial evolutivo

El algoritmo de Evolución Diferencial (ED) fue propuesto por Storm y Price [8], [9] en 1998 y consiste en una técnica no determinista basada en la evolución de una población de vectores (individuos) de valores reales que representan las soluciones en el espacio de búsqueda. La diferencia entre la ED y otros Algoritmos Evolutivos (AEs) es que emplea combinaciones lineales de individuos en la población actual, para generar nuevas soluciones que heredan características deseables para la solución del problema. Además, es eficiente en el momento de dar solución a problemas tanto reales como artificiales [16], [15]. La figura 2 describe por medio de un diagrama de flujo, la metodología general del algoritmo utilizado.

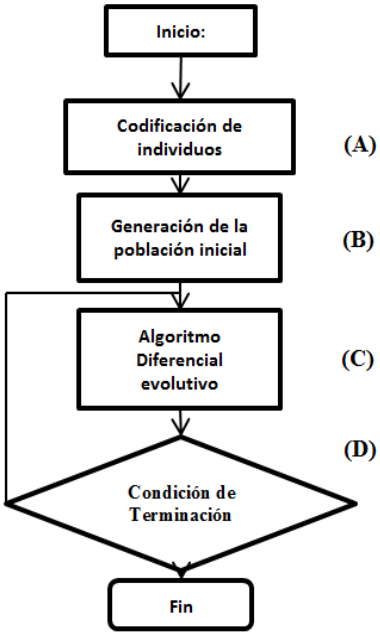

Figura 2. Diagrama de flujo de la estructura del algoritmo genético. [11]

B. Codificación de individuos

La codificación de cada individuo de la población se realiza mediante un cromosoma o vector en el que están contenidos *D* componentes o genes. Cada gen corresponde a un parámetro de entrenamiento de la red SOM respectivamente y es único en dicho cromosoma. La figura 3 muestra la representación del cromosoma para un individuo.

X_{ij} = | g_1 | g_2 | ……., | ……., | g_D |

Figura 3. Representación gráfica del cromosoma para un individuo

La tabla 1 contiene los parámetros de entrenamiento de la red SOM utilizados, diferentes opciones y sus respectivas codificaciones. Teniendo en cuenta los parámetros definidos en la sección II, D=5.

Parámetro	Opciones	Código
Método de normalizar (g_1)	Varianza	1
	Rango lineal	2
	Logarítmico	3
	Logístico	4
Forma del mapa (g_2)	Laminar	1
	Cilíndrico	2
	Toroidal	3
Estructura de la grilla (g_3)	Hexagonal	1
	Rectangular	2
Neuronas de salida(g_4)	Valor aleatorio entre (9/10)*n y (5√n)	X
Función de vecindad (g_5)	Gaussiana	1
	Gaussiana recortada	2
	Burbuja o exponencial	3

Tabla 1.Codificación de los parámetros de entrenamiento de la red SOM.

La figura 4 presenta un ejemplo de un cromosoma teniendo en cuenta la codificación de genes mostrada en la tabla 1.

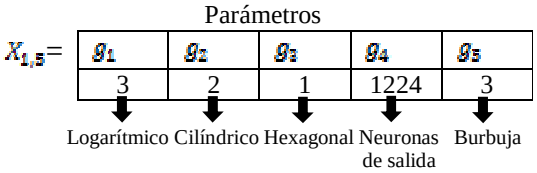

Figura 4. Ejemplo de codificación para un individuo.

C. Generación de la población inicial

La población inicial (pop) corresponde a una matriz [NPxD], donde las filas (NP) conciernen a la cantidad de individuos que tendrá la población y D la longitud de cada individuo o cromosoma como se muestra en la figura 5.

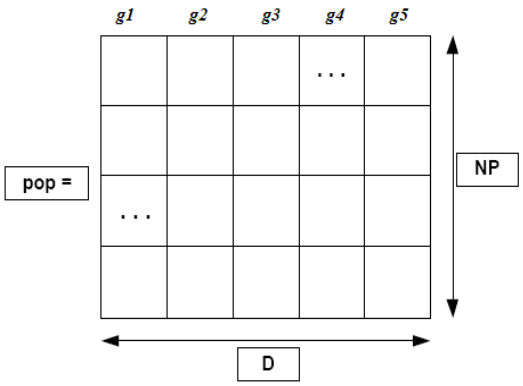

Figura 5. Población Inicial

Para generar cada gen de la población se selecciona de manera aleatoria un valor teniendo en cuenta la codificación expuesta en la tabla 1.

D. Operación del algoritmo diferencial evolutivo

La figura 6 describe el procedimiento que realiza el algoritmo implementado con el fin de realizar la búsqueda del mejor individuo de una población dada.

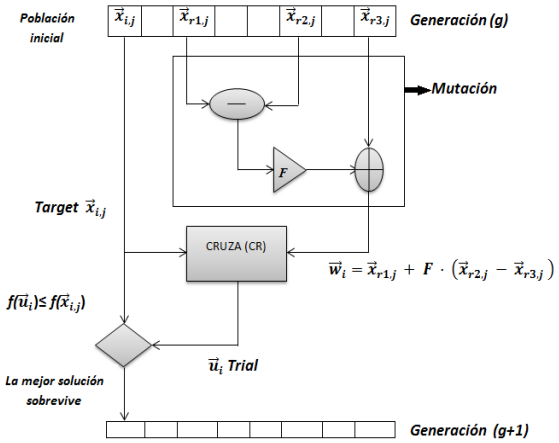

Figura 6. Funcionamiento del algoritmo evolutivo diferencial implementado.

La ED emplea una población de individuos (pop) que representan a las soluciones candidatas del problema original. Después de la inicialización, la población es sujeta a un proceso iterativo de mutación, recombinación (o cruce) y selección durante g generaciones o iteraciones. Para ello, se selecciona el vector *target* $\vec{x}_{i,j}$(primer vector de la población) y 3 vectores aleatoriamente que conforman la primera generación (g). Estos 3 individuos ($\vec{x}_{r1,i}$, $\vec{x}_{r2,i}$ y $\vec{x}_{r3,i}$) son utilizados para aplicar los operadores de mutación y cruce con el fin de obtener un vector denominado *trial*$(\vec{u}_i)$ (ver figura 6). A los vectores trial y target es corresponde un valor de la función de optimización o fitness, $f(\vec{u}_i)$ y $f(\vec{x}_{i,j})$ respectivamente.Dicho valor, permite decidir el integrante de la siguiente generación $(g+1)$.
El proceso debe ser repetido para cada uno de los individuos $\vec{x}_{i,j}$ de la población g, hasta completar la nueva generación $(g+1)$. El algoritmo desarrollara nuevas generaciones $(g+n)$ hasta que cumpla con el objetivo o condición de parada.
A continuación son detallados cada uno de los componentes del algoritmo.

Mutación

Para realizar la mutación diferencial en el ED, se seleccionan aleatoriamente tres individuos de la población inicial. Uno de ellos $\vec{x}_{r1,j}$ será el vector base y los restantes $\vec{x}_{r2,j}$ y $\vec{x}_{r3,j}$ los vectores diferenciales [10]. En esta operación se añade la diferencia proporcional de $\vec{x}_{r2,j}$ y $\vec{x}_{r3,j}$a $\vec{x}_{r1,j}$, para obtener un nuevo individuo mutado denominado $\vec{w}_i$ que se genera mediante la ecuación (12) (ver fig. 6).

$$\vec{w}_i = \vec{x}_{r1,j} + F \cdot (\vec{x}_{r2,j} - \vec{x}_{r3,j}) \qquad Ec\ (8)$$

r1, r2, r3 ∈ {1, 2,. . . . , NP}

Donde F corresponde a la constante de mutación y su función es establecer un rango de diferenciación entre los individuos x_{r2} y x_{r3} , lo que evita el estancamiento en el proceso de búsqueda. La elección habitual para este parámetro de control es un número entre 0,4 y 1, establecido de manera aleatoria para cada iteración del algoritmo. Esto permite variaciones estocásticas en la amplificación de la diferencia de los vectores y ayuda a mantener la diversidad de la población.

Recombinación o cruce entre padres

Tras la mutación, se realiza la operación de recombinación sobre cada individuo $\vec{x}_{i,j}$ (target) para generar un individuo intermedio $\vec{u}_i$ (trial). El individuo intermedio $\vec{u}_i$ es construido mezclando lascomponentes de $\vec{w}_i$ y $\vec{x}_i$, ecuación 11, bajo una probabilidad predefinida Cr ∈ [0, 1], como indica la ecuación 9.

$$\vec{u}_i = \begin{cases} \vec{w}_{i,(j)} \; si \; r \leq Cr_i \\ \vec{x}_{i,(j)} \; en \; otro \; caso \end{cases} \qquad Ec(9)$$

En la ecuación (9), *r* representa un número aleatorio de 0 a 1. Si *r* es menor que el coeficiente de cruce (CR), el elemento de la posición *j* del individuo mutado $\vec{w}_{i,(j)}$ se intercambia con el elemento *j* del individuo trial $\vec{u}_{i,(j)}$. En otro caso, se selecciona el elemento *j* del individuo target $\vec{x}_{i,(j)}$ para ser ubicado en el elemento *j* del individuo trial $ui(j)$ como se muestra en la figura 7 [10].

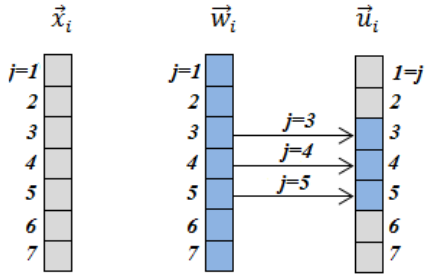

Figura 7. Operación de recombinación o cruza

El coeficiente de cruce es seleccionado de manera aleatoria para cada iteración del algoritmo, teniendo en cuenta el número de iteraciones así como el valor máximo y mínimo de CR [1,0]. [15], [17].

Selección de sobrevivientes

El operador de selección es calculado sobre los individuos *target* $(\vec{x}_{i,g})$y *trial*$(\vec{u}_{i,g})$para decidir el individuo que deberá integrar la nueva generación $(\vec{x}_{i,g+1})$. De esta forma, es posible aceptar mejoras sobre individuos anteriores evaluando la función de optimización $f(\vec{x}_i)$ como indica la Ecuación (10).

$$x_{i,g+1} = \begin{cases} u_{i,g} f(\vec{u}_i) \leq f(\vec{x}_i), \\ \vec{x}_{i,g} \; en \; otro \; caso \end{cases} \quad Ec \; (10)$$

La ecuación 11 expresa la función de optimización que permite evaluar la aptitud de cada una de los individuos que conforman la población objetivo del algoritmo evolutivo. De esta manera, se plantea minimizar una función que considere ponderaciones de los indicadores de error promedio definidos en la sección II apartado C. Es decir, se minimiza el error de identificación [6].

$$f(\vec{x}_{i,j}) = \sum_{i=1}^{n} w_i * e_i + DS \qquad Ec(11)$$

En donde:

w_i: Valor del peso.
e_i: Indicadores de error promedio.
DS: Desviación estándar.

E. Condición de terminación

Los criterios de terminación consideran los siguientes aspectos: i) que el valor de aptitud o *fitness* alcance un valor promedio, ii) una cantidad límite de iteraciones o iii) que el algoritmo no encuentre mejores individuos. Para el presente trabajo se determinó mediante pruebas pilotos ejecuciones del algoritmo no superiores a 20 iteraciones y que el valor *fitness* alcance un valor de 6.5%. Las pruebas pilotos mostraron que los individuos no presentan variaciones significativas alrededor del valor *fitness* establecido, y que más de 20 iteraciones requieren tiempos de ejecución superior a 48 horas.

IV. VALIDACIÓN NUMÉRICA

A. Caso numérico

La estructura utilizada para la validación numérica es el *Benchmark UBC*, consistente en un *frame* metálico de 4 pisos, con un modelo a escala 1:3 ubicado en el laboratorio de Ingeniería Sísmica de la Universidad de *British Columbia*, Canadá. La ASCE desarrolló un modelo numérico basado en las propiedades de la estructura *Benchmark*, este consiste en un modelo de 120 grados de libertad. El modelo se construyó teniendo en cuenta el método de elementos finitos (ver figura 8), considerando 9 elementos columna (verticales) en cada piso, 12 elementos viga (horizontales) en cada piso y 8 elementos de refuerzo (diagonales) en cada piso, para un total de 116 elementos. Cada piso, contiene 29 elementos entre columnas, vigas y refuerzos (‘*Braces*’). Además, las columnas se agrupan en 9 secciones, para las que se simularon diferentes escenarios de daño y condiciones de operación, utilizando alteraciones del modelo analítico (Ecuación 1).
El *Benchmark* proporciona los valores nominales de **M**, **K** y **C** de la ecuación 1. Teniendo en cuenta lo anterior, se alteró la matriz de rigidez **K** para producir variaciones porcentuales del módulo de

Young en cada elemento de la estructura, como resultado, se registraron 16 aceleraciones que representan cada variación. Las variaciones realizadas en cada elemento comprenden el rango [0,50%] incluyendo un caso de daño completo (100% de variación). Se consideraron, hasta 4 elementos con variación simultanea para un total de 6500 casos.

Es decir, cada caso corresponde a la variación porcentual de rigidez en el rango de [0,50%], para 1, 2, 3 o 4 elementos simultáneamente y está representado por 16 registros de aceleración. El modelo de la Ecuación 1 fue excitado con ruido blanco para simular operación ante vientos horizontales.

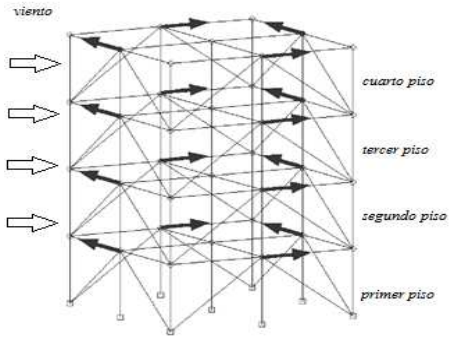

Figura 8. Estructura Benchmark utilizada. Modificado [6]

B. Resultados numéricos

La tabla 2 contiene la población inicial para el mejor individuo de la población inicial obtenido para una ejecución, con el fin de dar a conocer cómo varían los parámetros de entrenamiento de la red SOM en cada ejecución del algoritmo ED.

	Parámetros de la red SOM					
Individuos	g1	g2	g3	g4	g5	fitness
	4	1	2	1220	1	0.1024
	2	2	2	1745	1	0.0824
	4	3	1	1485	1	0.0881
	2	3	2	1736	2	0.0838
	1	3	2	1294	3	0.0901

Tabla 2.Población inicial.

Se emplearon 3 grupos en la validación cruzada; así, se aplica la metodología CBR sobre 2 grupos y se evalúan los errores presentados en la sección II apartado D sobre los datos del grupo que no se tuvo en cuenta. El proceso se repite de tal forma que a cada grupo se le calcule un error de identificación. De esta forma se obtienen 3 errores de identificación para los cuales se calcula el valor promedio y la desviación estándar (DS). La tabla 3 y 5 permiten ver los resultados obtenidos luego de realizado el proceso tanto para el mejor individuo de la población inicial como para el mejor individuo de la población final.

Grupos de validación cruzada						
Error	**Grupo 1**		**Grupo 2**		**Grupo 3**	
	FP	**FN**	**FP**	**FN**	**FP**	**FN**
	53	6	48	11	47	9
	errTy	errFl	errTy	errFl	errTy	errFl
	0.021	0.091	0.018	0.087	0.019	0.090
Indices	**Entrenamiento con el conjunto de datos completo**					
	Nl		**nh**		**Nv**	
	0.3807		0.6193		0.3807	
	0.3847		0.6153		0.3847	
	0.3854		0.6146		0.3854	

Tabla 3. Indicadores y errores sobre los grupos de entrenamiento en validación cruzada para el mejor individuo de la población inicial.

Para interpretar mejor las tablas 3 y 5, *errFl* corresponde a la cantidad de elementos clasificados erróneamente en un piso específico (primero, segundo, tercero o cuarto), *errTy* corresponde al número de elementos que fueron clasificados erróneamente en un tipo específico (columna, viga o brazo), FN el número de falsos negativos y FP es el número de falsos positivos.

Luego de realizadas 20 iteraciones se encontró la población final que se muestra en la tabla 4, en la que se puede apreciar una disminución en el valor de la función *fitness* para cada individuo de dicha población con respecto a la mostrada en la tabla 2.

	Parámetros de la red SOM					
Individu	g1	g2	g3	g4	g5	*Fitness*
	4	1	2	1213	1	0.0779
	2	1	2	1800	3	0.0757
	4	1	2	1800	3	0.0687
	4	1	1	1456	1	0.0725

os	4	1	1	1532	1	0.0778

Tabla 4. Población final.

Para obtener los valores de *fitness* mostrados en las tablas 2 y 4, se realizó el promedio de dichos valores para los tres grupos de validación cruzada, además de adicionar la desviación estándar, esto con el fin de obtener el mejor individuo, debido a que el cálculo de la desviación estándar nos asegura un resultado más exacto sobre el valor de la función de optimización, ya que este se ve afectado en gran parte al ser un promedio.

La figura 9 muestra el comportamiento de la función de optimización realizadas 20 iteraciones, en donde los valores mostrados corresponden a los mejores vectores de parámetros encontrados por el algoritmo para cada iteración, además se puede apreciar que el mejor individuo mostrado en la tabla 4, posee un valor de *fitness* de 0.0687.

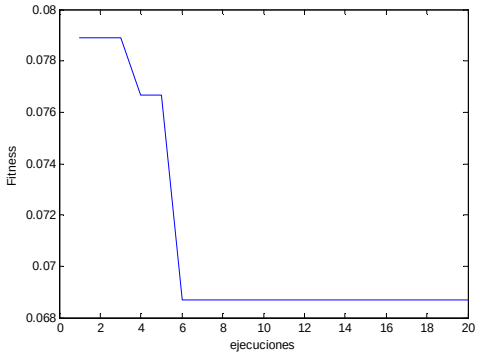

Figura 9. Comportamiento de la función fitness después de 20 iteraciones

Se calculó cada uno de los correspondientes errores e indicadores mencionados anteriormente para cada uno de los grupos de validación cruzada del mejor individuo, estos valores se pueden ver en la tabla 5.

	Grupos de validación cruzada					
Error	**Grupo 1**		**Grupo 2**		**Grupo 3**	
	FP	**FN**	**FP**	**FN**	**FP**	**FN**
	35	0	44	4	38	0
	errTy	errFl	errTy	errFl	errTy	errFl
	0.016	0.074	0.016	0.069	0.018	0.070
Ind	**Entrenamiento con el conjunto de datos completo**					
	nl		**nh**		**nv**	
	0.4124		0.5876		0.4124	
ices	0.4177		0.5823		0.4177	
	0.4186		0.5814		0.4186	

Tabla 5. Indicadores y errores sobre los grupos de entrenamiento en validación cruzada para el mejor individuo de la población final.

Se observó que existe una disminución en contraste con los obtenidos en [6], esto permitió obtener mejores resultados en la localización, identificando de una mejor manera el piso donde se ubica el daño y el tipo de elemento con daño, además también permite ver que existe una mejor distribución de los casos en cada *cluster*, debido a que se tienen valores de *nh* y de *nl* alrededor del 50% cada uno. Además de esto, se determinaron otros indicadores de error relacionados con el error medio y la desviación estándar, en la estimación de la ubicación del elemento y el porcentaje de daño (*mean_El*, *std_El*, *mean_Se*, *std_Se* respectivamente). Asimismo, se calculó el error en la estimación de la dimensión del daño.

En comparación con la tabla 3 se disminuyeron los errores e indicadores aunque en algunos casos no la diferencia no es amplia debido a que al ser seleccionada la población de forma aleatoria se puede obtener un individuo muy cercano al mejor obtenido en la población final.

Las figuras 10, 11, 12 ,13 y 14 permiten observar como disminuyeron los indicadores de error en contraste con los obtenidos en [6], además se puede apreciar de igual manera que el grado de distanciamiento con respecto a su valor medio también presenta una disminución significativa, permitiendo así realizar un aporte de gran importancia a la metodología de estimación de la severidad del daño.

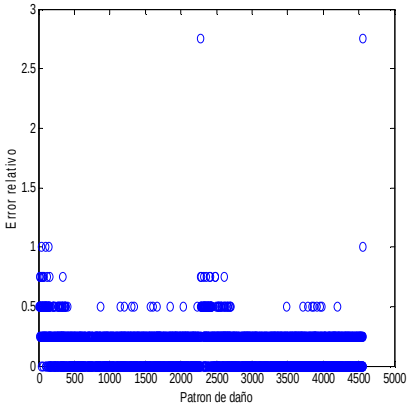

Figura 10. Error relativo de dimensión.

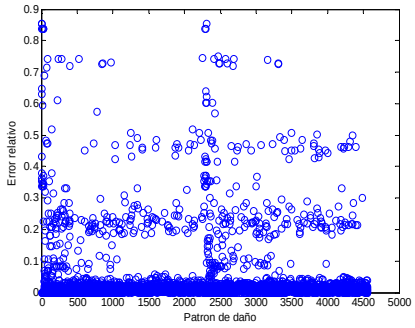

Figura 11.Error medio del número del elemento.

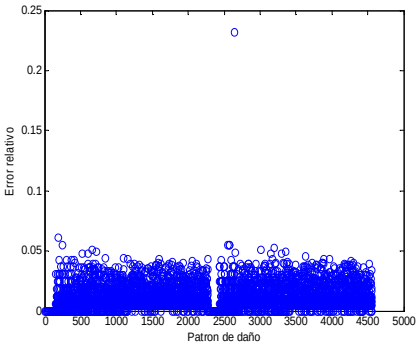

Figura 12. Desviación estándar del número del elemento

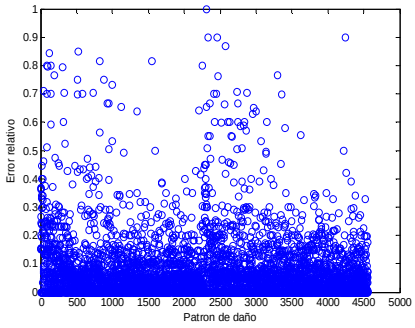

Figura 13. Error relativo para la estimación de severidad. Error medio.

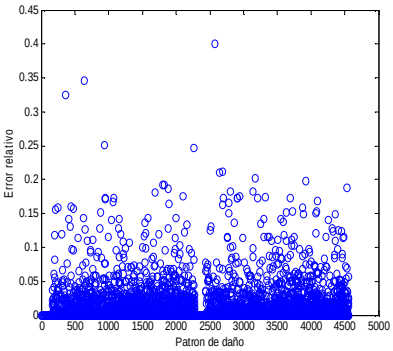

Figura 14. Error relativo para la estimación de severidad. Desviación estándar.

La figura 15 permite observar el comportamiento del error topográfico transcurridas 20 iteraciones para una ejecución, en donde se aprecia que a medida que el valor de la función *fitness* disminuye, la medición de la proyección del mapa se aproxima a cero indicando de esta forma que existe una mejora en la calidad de la red SOM.

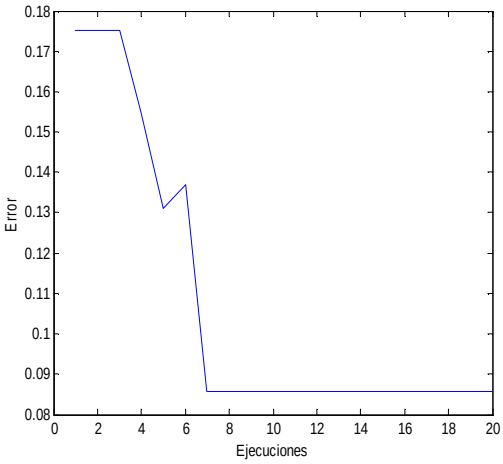

Figura 15. Error topográfico

La figura 16 permite ver las medidas de distorsion del mapa, donde se observa que las variaciones de cada uno de las unidades del mapa tienen valores cercanos a cero, manteniendo asi una buena topologia ya que sus medidas de distorsion son bajas. La distorsion media del mapa es: $\bar{E}$ =0.7664

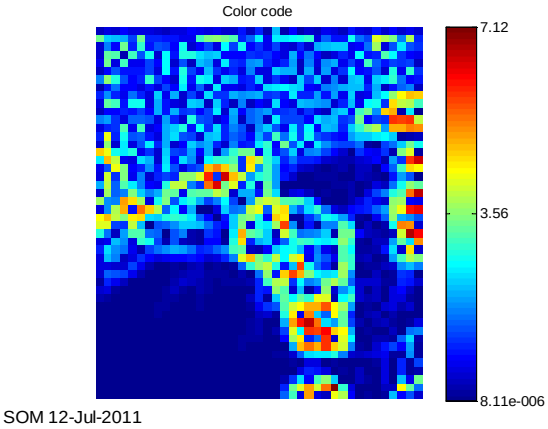

Figura 16. Medida de Distorsión para cada unidad del mapa

También es importante analizar la tendencia que se presenta una vez se inicializala metodologia propuesta para diferentes poblaciones iniciales aleatorias. La figura 17 permite analizar la tendencia, con el que para diferentes valores finales de *fitness* obtenidos luego de realizar 6 ejecuciones diferentes con un valor minimo de 20 iteraciones, se puede observar que los respectivos valores de fitness se encuentran entre un 7% y un 6% aproximadamente.

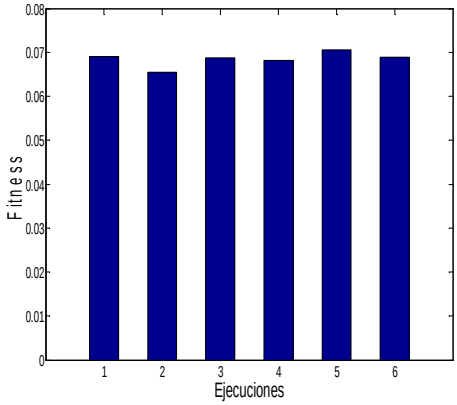

Figura 17.Comportamiento de la función fitness para 6 ejecuciones.

La figura 18 permite observar que el valor final del error topográfico tiende alrededor del 8 %, lo que indica que al ser estos valores bajos, permite concluir que se obtuvo una buena calidad del mapa.

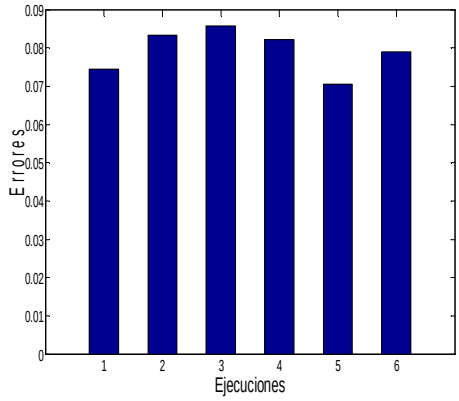

Figura 18. Comportamiento del error topográfico para 6 ejecuciones.

La tabla 6 contiene los parámetros finales obtenidos con sus respectivos valores de fitness para 6 ejecuciones realizadas. Lo que permite apreciar como la red SOM es sensible a los parámetros seleccionados, debido a que para obtener un menor valor de fitness es importante que el parametro número de clusters (g4) se encuentre en un valor cercano o igual a 1800, de igual forma se analizó que el parametro forma del mapa (g2), también afecta los resultados como se aprecia en la tabla 6, donde el mejor individuo de la ejecución 3, a pesar que el parámetro g4 toma un valor de 1800, este presenta un mejor valor de fitness que el mejor individuo de la ejecución 1con una opción cilindrica (2) en dicho parámetro.

		Parámetros de la red SOM					
		g1	g2	g3	g4	g5	*fitness*
Ejecuciones	1	4	1	1	1800	3	0.0690
	2	1	2	2	1800	1	0.0655
	3	4	1	2	1800	3	0.0687
	4	2	3	1	1800	3	0.0682
	5	1	1	2	1800	1	0.0705
	6	1	2	2	1733	3	0.0689

Tabla 6.Mejores individuos para 6 ejecuciones.

Luego de realizadas diferentes ejecuciones, se encontró que el parámetro función de vecindad (g5), debe encontrarse entre la opción 1 (gaussiana) y la opción 3 (burbuja o exponencial) con el fin de obtener la mejor *fitness* (ver tabla 6), pero de igual forma el mejor individuo obtenido en repetidas ocasiones para diferentes ejecuciones fue el que tiene un valor de *fitness* de 0.0655 como se muestra en la tabla 6, es preferible que dicho parámetro (g5) se encuentre en la opción 1 (gaussiana).

V. CONCLUSIONES

Con los resultados obtenidos en la sección IV se demuestra que la automatización de los parámetros permite obtener un mejor desempeño de la red neuronal SOM, en donde se disminuyó cada uno de los errores y los indicadores mencionados con respecto a trabajos previos [6], sin embargo, el costo computacional para entrenamiento y operación de la red SOM aumenta, debido a que para obtener los mejores valores de la función de optimización el número

de *clusters* debe estar alrededor de 1800 quedando alrededor del 40% de *clusters* vacíos.

Teniendo en cuenta que el valor del peso fue el mismo para cada uno de los errores de la función *fitness*, se observó que todos disminuyen indicando que ninguno de ellos predomina sobre los demás. Sin embargo, se propone para trabajos futuros modificar este peso de manera que algunos de ellos sean mayores, preferiblemente el error de piso y el error de tipo con el fin de mejorar la distribución geométrica y dinámica de la estructura.

El error topográfico presentó una disminución con respecto al obtenido en [6], lo que permite concluir que existe una mejor preservación de la topología, sin embargo, se pudo apreciar que la tendencia que presenta luego de realizadas varias ejecuciones tiene un valor alrededor del 8%, por lo que es recomendable para trabajos futuros realizar un cambio en la medida de similaridad utilizada, con el fin de encontrar si es posible un mejor valor del mismo.

Luego de observar los resultados obtenidos sobre el comportamiento del algoritmo, transcurridas varias ejecuciones, se concluye que de igual manera cómo influye el número de *clusters* también es importante que la función de vecindad se mantenga entre las opciones gaussiana y burbuja para obtener el mejor valor de la función de optimización posible. Además se pudo apreciar que la forma del mapa y la estructura de la grilla no repercuten de manera significativa en la búsqueda del mejor valor.

VI. BIBLIOGRAFÍA.

[1]Andrés Felipe Quintero Parra y Rodolfo Villamizar Mejía, Estado del arte en monitorización de salud estructural: Un enfoque basado en agentes inteligentes. Junio 2010Ciencia e Ingeniería Neogranadina, Vol. 20-1, pp. 117-132. Bogotá, Junio de 2010.

[2] Giraldo, Diego. "A Structural Health Monitoring Framework for Civil Structures".Doctoral Dissertation thesis.Washington University. 2006.

[3] Caicedo, J.M. and Johnson, E.A., "Structural Health Monitoring of Flexible Civil Structures," Doctoral Dissertation thesis. Washington University. 2003.

[4] Structural Control and Earthquak Engineering [en linea] Laboratory.Washington University. Disponible en internet : http://mase.wustl.edu/wusceel/quake/

[5] Marulanda, J., Thomson, P., Caicedo, J. M., Dyke, Shirley J. Monitoreo de Salud Estructural de Puentes Metálicos.Proceedings of the First Colombian Workshop on Steel Structures, Cartagena, Colombia, Sep. 26-28, 2001.

[6]Jhonatan Camacho Navarro. 2010. Sistema Experto para la Monitorización de Salud Estructural mediante el Reconocimiento de Patrones: Adaptación y Validación Numérica. Tesis de grado obtenido Universidad Industrial de Santander, Bucaramanga, Colombia.

[7] DSMus, Razonamiento basado en casos CBR. http://www.imaisoftware.com/openlab/da-ta/proyectos/DSMus/presentación.pdf [6 de febrero de 2010].

[8] Storn, R., Price, K.: Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. J. of Global Optimization 11(4) (1997) 341–359

[9] Price, K.V., Storn, R.M., Lampinen, J.A.: Differential Evolution A Practical Approach to Global Optimization Natural Computing Series. Springer-Verlag, Berlin, Germany (2005)

[10] Kenneth V. Price, Rainer M. Storn, and Jouni A. Lampinen. DifferentialEvolution A Practical Approach to Global Optimization. Natural ComputingSeries. Springer-Verlag, Berlin, Germany, 2005.

[11]T. Kohonen, Self-Organizing Maps, 3rd ed. Springer, 2001, vol. 30.

[12]Juan Miguel Marín Diazaraque (Ph.D). Los mapas auto-organizados deKohonen (SOM) Universidad Carlos III de Madrid

[13] Laboratory of Computer and Information Science (CIS) Adaptive Informatics Research Centre [en linea] Department of Computer Science and Engineering at the Helsinki University of Technology [Citado Diciembre de

2007 Disponible en internet:]http://www.cis.hut.fi/somtoolbox/

[14] Marín Juan. Introducción a las Redes NeuronalesAplicadasCURSO DE EXPERTOS DE U.C.M. (2011) http://halweb.uc3m.es/esp/Personal/personas/jmmarin/esp/Expertos/CursoNN11.pdf [6 de febrero de 2010]

[15] Luis Vicente Santana Quintero.Un Algoritmo Basado en Evolución Diferencialpara Resolver Problemas Multiobjetivo. Tesis de maestría obtenido Centro de Investigación y de Estudios Avanzadosdel Instituto Politécnico Nacional Departamento de Ingeniería Eléctrica Sección de Computación, Noviembre de 2004. México DF.

[16]Carlos Augusto Monterrosa López. Analizando la Relación entre elFactor de Escalamiento y losTamaños de Población en elAlgoritmo de EvoluciónDiferencial al Resolver Problemasde Optimización con Restricciones.Dirección General de Educación Superior Tecnológica Instituto Tecnológico de Tuxtla Gutiérrezdepartamento de Ingeniería en Sistemas Computacionales. México, Enero de 2010.

[17] Swagatam Das, Ajith Abraham, AmitKonar. Automatic Clustering Using an Improved Differential Evolution Algorithm. [en linea] [6 de febrero de 2010]. Disponible en internet: http://www.softcomputing.net/smca-paper1.pdf

[18] JuhaVesanto, Johan Himberg, Esa Alhoniemi and Juha Parhankas, Organizing Map in Matlab: the SOM Toolbox. Laboratory of Computer and Information Science Helnsinki University of Technology. Finland. February 18, 2000.

ANEXOS

A. REDES NEURONALES SOM

Un modelo SOM está compuesto por dos capas de neuronas. La capa de entrada (formada por N neuronas, una por cada variable de entrada) se encarga de recibir y transmitir a la capa de salida la información procedente del exterior. La capa de salida (formada por M neuronas) es la encargada de procesar la información y formar el mapa de rasgos. Cada neurona de entrada i está conectada con cada una de las neuronas de salida j mediante un peso wji. De esta forma, las neuronas de salida tienen asociado un vector de pesos Wj llamado vector de referencia (o *codebook*), debido a que constituye el vector prototipo (o promedio) de la categoría representada por la neurona de salida j. Así, el SOM define una proyección desde un espacio de datos en alta dimensión a un mapa bidimensional de neuronas. (Ver fig. A.1)

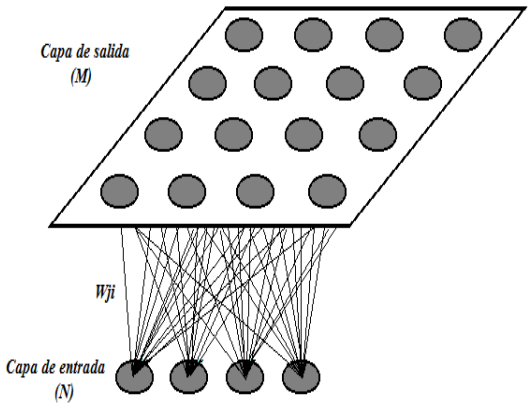

Figura A.1. Mapa bidimensional de la Red SOM modificado [12]

Entre las neuronas de la capa de salida, puede decirse que existen conexiones laterales de excitación e inhibición implícitas, pues aunque no estén conectadas, cada una de estas neuronas va a tener cierta influencia sobre sus vecinas. Esto se consigue a través de un proceso de competición entre las neuronas y de la aplicación de una función denominada de vecindad, que produce la topología o estructura del mapa. Las topologías más frecuentes son la rectangular y la hexagonal. Las neuronas adyacentes pertenecen a una vecindad Nj de la neurona j. La topología y el número de neuronas permanecen fijos desde el principio. Los vectores de referencia del *codebook* se acercan a las áreas donde la densidad de datos es alta [11], [12].
El proceso de aprendizaje de la SOM consiste en que un vector x es seleccionado al azar del conjunto de datos y se calcula la distancia (similitud) a los vectores del *codebook*, usando la distancia euclídiana Ecuación (1):

$$\| x - mc \| = mín\{\| x - mj \|\} \qquad Ec(1)$$

Una vez que se ha encontrado el vector más próximo o BMU (*best matching unit*) el resto de vectores del *codebook* es actualizado. Durante la fase de entrenamiento, el SOM forma una red elástica que se pliega dentro de la nube de datos originales. El *BMU* y sus vecinos (en sentido topológico) se mueven cerca del vector x en el espacio de datos como se puede apreciar en la figura A.2. Las líneas continuas y discontinuas corresponden a la situación antes y después de la actualización, respectivamente.

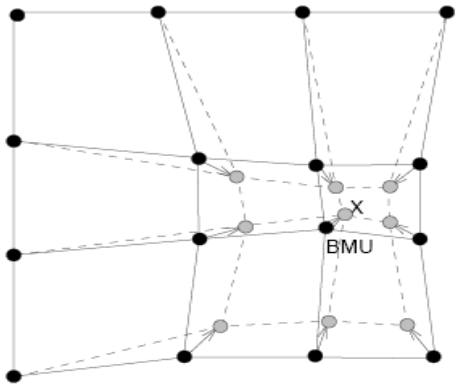

Figura A.2. Actualizando el vector más próximo (BMU) y sus vecinos a la muestra de entrada marcada con x. Fuente [18].

La magnitud de dicha atracción está regida por la tasa de aprendizaje. Mientras se va produciendo el proceso de actualización y nuevos vectores se asignan al mapa, la tasa de aprendizaje decrece gradualmente hacia cero. Junto con ella también decrece el radio de vecindad. La regla de actualización para el vector de referencia dado i es el de la ecuación (2):

$$m_j(t+1) = \begin{cases} m_j(t) + \alpha(t)\left(x(t) - m_j(t)\right) & j \epsilon N_c(t) \\ m_j(t) \; j \epsilon N_c(t) & \end{cases} \quad Ec(2)$$

Donde $\alpha(t)$es la tasa de aprendizaje$(0 < \alpha(t) < 1)$. El proceso de aprendizaje se va repitiendo hasta que el entrenamiento termina. El número de pasos de entrenamiento se debe fijar a priori, para calcular la tasa de convergencia de la función de vecindad y de la tasa de aprendizaje. Una vez terminado el entrenamiento, el mapa ha de ordenarse en sentido topológico: n vectores topológicamente próximos se aplican en n neuronas adyacentes o incluso en la misma neurona. (Figura 2). [11], [12].

La figura A.3 permite observar como la red SOM genera *clusters* (grupos) donde se organizan los casos con características similares. De esta manera, esta red está conformada por *L* neuronas (una por cada característica) en la capa de entrada y *r* neuronas o *clusters* en la capa de salida. Como resultado, se obtiene un modelo por cada *clúster* de casos [6], [7].

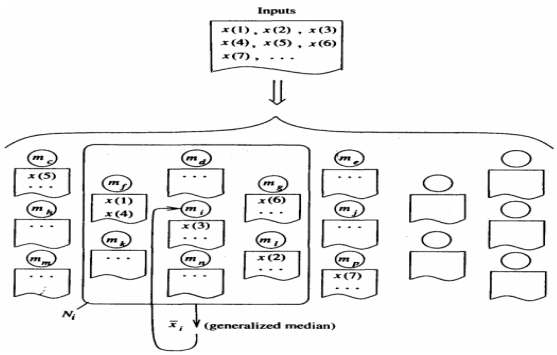

Figura A.3. Entrenamiento de la Red SOM. Fuente [11]

En este sentido, en la metodología de identificación presentada, luego de haberse entrenado el mapa auto-organizado, se generan nuevos casos con el fin de validar la metodología. Seguidamente, se obtiene el vector de características principales para el nuevo patrón de daño. Con estas características, la red SOM recupera un conjunto de casos similares y almacenados en un *clúster* basándose en una medida de similaridad fundamentada en la menor distancia euclidiana entre el vector de características principales y el modelo del *clúster*.