Estrategias evolutivas y toma de decisiones

Research in Computing Science

Series Editorial Board

Research in Computing Science es una publicación trimestral, de circulación internacional, editada por el Centro de Investigación en Computación del IPN, para dar a conocer los avances de investigación científica y desarrollo tecnológico de la comunidad científica internacional. **Volumen 94,** mayo 2015. Tiraje: 500 ejemplares. *Certificado de Reserva de Derechos al Uso Exclusivo del Título* No. : 04-2005-121611550100-102, expedido por el Instituto Nacional de Derecho de Autor. *Certificado de Licitud de Título* No. 12897, *Certificado de licitud de Contenido* No. 10470, expedidos por la Comisión Calificadora de Publicaciones y Revistas Ilustradas. El contenido de los artículos es responsabilidad exclusiva de sus respectivos autores. Impreso en la Ciudad de México, en los Talleres Gráficos del IPN – Dirección de Publicaciones, Tres Guerras 27, Centro Histórico, México, D.F. Distribuida por el Centro de Investigación en Computación, Av. Juan de Dios Bátiz S/N, Esq. Av. Miguel Othón de Mendizábal, Col. Nueva Industrial Vallejo, C.P. 07738, México, D.F. Tel. 57 29 60 00, ext. 56571.

Editor responsable: *Grigori Sidorov*, *RFC SIGR651028L69*

Research in Computing Science is published by the Center for Computing Research of IPN. **Volume 94**, May 2015. Printing 500. The authors are responsible for the contents of their articles. Printed in Mexico City, in the IPN Graphic Workshop – Publication Office.

Volume 94

Estrategias evolutivas y toma de decisiones

María de Lourdes Martínez Villaseñor (ed.)

Instituto Politécnico Nacional, Centro de Investigación en Computación
México 2015

ISSN: 1870-4069

Instituto Politécnico Nacional (IPN)
Centro de Investigación en Computación (CIC)
Av. Juan de Dios Bátiz s/n esq. M. Othón de Mendizábal
Unidad Profesional "Adolfo López Mateos", Zacatenco
07738, México D.F., México

http://www.rcs.cic.ipn.mx
http://www.ipn.mx
http://www.cic.ipn.mx

Indexed in LATINDEX and Periodica / Indexada en LATINDEX y Periódica

Printing: 500 / Tiraje: 500

Printed in Mexico / Impreso en México

Editorial

Las estrategias inspiradas en la evolución y los sistemas inteligentes de toma de decisiones, ramas de la inteligencia artificial, contribuyen hoy en día aportando soluciones a problemas complejos y se han vuelto clave para el desarrollo de muchas aplicaciones novedosas que tienen impacto importante en la vida diaria. El propósito de este volumen es reflejar las nuevas direcciones de investigación y aplicaciones de los métodos de la inteligencia artificial en estas áreas.

Esta publicación contiene 15 de los artículos relacionados con varios aspectos del desarrollo de los métodos de estrategias evolutivas y toma de decisiones. Hay ejemplos de aplicaciones de inteligencia artificial en diferentes tareas como:

- Problema de calendarización de horarios,
- Generación de pronósticos en una universidad,
- Sistema de recomendación de música,
- Aplicaciones en tomografía sísmica,
- Sistema recomendador de rutinas de ejercicio,
- Identificación de gases,
- Recolección de residuos sólidos,
- Clasificación para el análisis de sentimientos.

También se pueden encontrar en este volumen tres artículos que extienden y mejoran estrategias evolutivas y de toma de decisiones, además de dos artículos que proponen nuevos operadores genéticos semánticos.

Este volumen puede ser interesante para los investigadores y estudiantes de las ciencias de la computación y en particular a aquellos interesados en los temas relacionados con estrategias evolutivas y toma de decisiones.

Agradecemos a los autores de los artículos que constituyen este volumen. La calidad de su investigación y desarrollos tecnológicos contribuyen al valor de esta publicación. A nombre de la comunidad académica del Centro de Investigación e Innovación en Tecnologías de la Información y Comunicación (INFOTEC) y de la SMIA expresamos nuestro agradecimiento al Dr. Sergio Carrera Riva Palacio, Director Ejecutivo, y Dr. Juan Carlos Téllez Mosqueda, Director Adjunto de Innovación y Conocimiento, por apoyar de manera ingente la investigación y el desarrollo de la ciencia y la tecnología, sustentado todo ello en la responsabilidad y el compromiso social.

El proceso de revisión y selección de artículos se llevó a cabo usando el sistema libremente disponible EasyChair, www.EasyChair.org.

María de Lourdes Martínez Villaseñor
Mayo 2015

Table of Contents

Page

Modificación al algoritmo de Hooke-Jeeves para búsqueda local en variantes de evolución diferencial para resolver problemas de optimización con restricciones

Angel Juan Sánchez García[1], Homero Vladimir Ríos Figueroa[2], Gerardo Contreras Vega[1], Juan Carlos Pérez Arriaga[1] y María Karen Cortés Verdín[1]

[1] Facultad de Estadística e Informática, Universidad Veracruzana, Xalapa, Veracruz, México

[2] Centro de Investigación en Inteligencia Artificial, Universidad Veracruzana, Xalapa, Veracruz, México

{angesanchez, hrios, gcontreras, juaperez, kcortes}@uv.mx

Resumen. Este trabajo presenta una modificación del algoritmo de Hooke-Jeeves implementado en variantes de Evolución Diferencial para resolver problemas de optimización con restricciones. El algoritmo de Hooke-Jeeves promueve una mejor exploración y explotación en zonas prometedoras para encontrar mejores soluciones. El algoritmo de Hooke-Jeeves modificado es implementado en 4 variantes de Evolución Diferencial: DE/rand/1/bin, DE/best/1/bin, Modified Differential Evolution (MDE) y Differential Evolution Combined Variants (DECV). Este enfoque es probado en un conjunto de problemas de referencia sobre optimización con restricciones. Los resultados son discutidos y algunas conclusiones son establecidas.

Palabras clave: Hooke-Jeeves, optimización, restricción, evolución diferencial, búsqueda local.

1. Introducción

Los algoritmos evolutivos han sido ampliamente utilizados para resolver problemas de optimización [1,2]. Sin embargo, en sus versiones originales carecen de mecanismos para hacer frente a problemas con restricciones [3], por ello se han hecho modificaciones a estos algoritmos.

Evolución diferencial (ED) es un algoritmo evolutivo propuesto por Storm y Prices [4] para resolver problemas de optimización principalmente en espacios continuos. ED se diferencia de otros algoritmos evolutivos en que no utiliza una codificación binaria como los algoritmos genéticos [5] ni utiliza una función de densidad de probabilidad autoadaptativa como en la estrategia evolutiva [6].

Con el objetivo de obtener mejores resultados basados en el enfoque de ED, se han propuesto algunas variantes de este algoritmo caracterizadas por el tipo

de recombinación y el número y tipo de soluciones para calcular los valores de las mutaciones. A continuación se describen brevemente las variantes utilizadas para la incorporación de nuestro enfoque de búsqueda local.

La variante más popular es DE/rand/1/bin, donde *DE* significa Evolución Diferencial, *rand* indica que los individuos que son seleccionados para ser mutados se escogen de manera aleatoria, *1* es el número de pares de soluciones escogidos y *bin* significa que se utiliza una recombinación binomial [4]. En DE/best/1/bin la única diferencia con respecto a DE/rand/1/bin es que el vector base no se escoge aleatoriamente, si no que es el mejor vector de la población actual. Con respecto a Modified Differential Evolution (MDE) en [3], se propone que la incorporación de la información de la mejor solución y el padre actual, acoplado con un mecanismo que permita a cada padre generar más de un descendiente, aumenta la capacidad de exploración y explotación el algoritmo ED en problemas de optimización. También que la incorporación de un mecanismo de diversidad permite que ED se acerque a la región factible de una mejor manera como para alcanzar el óptimo global [3].

En Differential Evolution Combined Variants (DECV), los autores en [7] proponen la combinación de DE/rand/1/bin y DE/best/1/bin, donde DE/rand/1/bin es utilizado primero para generar un conjunto más diverso de direcciones de búsqueda y cambiar a DE/best/1/bin para centrar la búsqueda en la vecindad del mejor vector posible, con un criterio de cambio al obtener el 10 % de los vectores factibles. La investigación sobre búsqueda local aplicada a Evolución Diferencial para resolver problemas de optimización numérica con restricciones, es la principal motivación de este documento, en el cuál una modificación al algoritmo de Hooke-Jeeves es presentado para mejorar la búsqueda local.

Este documento está organizado como sigue: en la sección 2 la definición de los problemas de optimización numérica con restricciones es presentada, en la sección 3 son presentados algunos trabajos relacionados que dan pie a la motivación de este trabajo, en la sección 4 la descripción de la búsqueda local es presentada. En la sección 5 es descrita nuestra propuesta, mientras que en la sección 6 los resultados y la discusión son presentados. Por último en la sección 7 se presentan las conclusiones y el trabajo futuro propuesto.

2. Planteamiento del problema

Los problemas que se pretenden resolver, son problemas en general no lineales que se definen como sigue:

Encontrar x tal que optimice *f(x)*
Sujeta a:

$$g_i(x) \leq 0, i = 1, ..., p \tag{1}$$

$$h_j(x) = 0, j = 1, ..., q \tag{2}$$

donde x es el vector de parámetros $= [x_1, x_2, \ldots, x_n]^T$, p es el número de restricciones de desigualdad y q el número de restricciones de igualdad (p y q

pueden ser lineales o no lineales). Para manejar las restricciones de igualdad, usualmente son transformadas a desigualdades como sigue [8]: $|h_j(x) - \varepsilon| \leq 0$, donde ε es una tolerancia permitida (un valor muy pequeño). El valor de $\varepsilon = 1\text{x}10^{-4}$ [9] fue usado en este trabajo.

3. Trabajos relacionados

Otros autores han propuesto enfoques que se acoplan con algoritmos evolutivos para obtener mejores resultados. Un nuevo criterio de selección para soluciones candidatas, conjugado con un operador basado en el método Nelder-Mead, es el enfoque *HDESMCO* propuesto en [10], donde se puede ver la idea de incorporar algún mecanismo que ayude a *ED* a encontrar mejores soluciones.

En [11] se propone la inclusión de una búsqueda local asistida por un meta modelo basado en máquinas de soporte de vectores donde se involucra el método de Hooke-Jeeves. Este mecanismo de búsqueda local es incluido en algoritmos evolutivos multi-objetivo.

La búsqueda local, ha tomado gran importancia recientemente para tratar de alcanzar óptimos globales. En [12], los autores desarrollan un marco de optimización multiobjetivo basado en la clasificación no dominado y búsqueda local NSLS (por sus siglas en inglés). El NSLS se basa en iteraciones. En cada iteración, dada una población *P*, un método simple de búsqueda local es usado para obtener una mejor población *p'*, y luego la clasificación no dominada se adopta en $P \cup P'$ para establecer una nueva población en la próxima iteración. Además, el candidato más lejano se combina con la clasificación de los no dominados para elegir la nueva población y así mejorar la diversidad.

En [13] se propone un cómputo memético adaptativo como la sinergia de un algoritmo genético, una evolución diferencial y estimación de un algoritmo de distribución. La relación entre el número de soluciones producidas por los algoritmos en una generación definen características de adaptabilidad en la próxima generación. Lo importante de la relación de este trabajo con el nuestro, es que posteriormente, un subconjunto de las soluciones pasa por la evolución de búsqueda local utilizando el algoritmo de búsqueda de gradiente.

En [9] se incluye el método de Hook-Jeeves como mecanismo de búsqueda local aplicado a la modificación del algoritmo *Modified Bacterial Foraging Algorithm* (MBFOA).

4. Búsqueda local

El método de búsqueda local trabaja creando un conjunto de direcciones. Las direcciones de búsqueda deben ser independientes y cubrir todo el dominio de *f(x)*. Por lo tanto, en un problema N-dimensional, este método requiere por lo menos N direcciones de búsqueda linealmente independientes [14].

4.1. Método de patrones de búsqueda de Hooke-Jeeves

El método de patrones de búsqueda de Hooke-Jeeves [15] crea un conjunto de direcciones de búsqueda de manera iterativa. Este método fue propuesto en 1966 y fue uno de los primeros algoritmos en incorporar la historia previa de una secuencia de iteraciones en la generación de una nueva dirección de búsqueda. El algoritmo combina movimientos *exploratorios* y movimientos de *patrones* con alguna heurística.

4.2. Movimientos exploratorios

Los movimientos exploratorios examinan la vecindad del punto actual para encontrar el mejor punto alrededor del mismo [14]. Por lo tanto, los movimientos exploratorios examinan el comportamiento local de la función y buscan localizar la direccion de cualquier pendiente existente en la zona.

4.3. Movimientos de patrones

Este movimiento es realizado por dos puntos, el original y el nuevo. El movimiento de patrones utiliza la información generada en la exploración para escalar rápidamente las pendientes. Si el movimiento de patrones no toma la solución a una mejor región, entonces no tiene éxito y se reduce el alcance de la búsqueda exploratoria. Esto se repite hasta converger, es decir, cuando el movimiento prácticamente es mínimo. El nuevo punto es encontrado por el salto del mejor punto actual en la dirección que conecte el anterior mejor punto $x^{(k-1)}$ y el actual punto base $x^{(k)}$ como se muestra en 3:

$$xp^{(k+1)} = x^{(k)} + (x^{(k)} - x^{(k-1)}) \tag{3}$$

5. Nuestra propuesta

Cuando se habla de una búsqueda local existen por lo menos tres preguntas que se deben responder: ¿A quién se le aplica la búsqueda local? ¿Con qué frecuencia? y ¿Cuál será la condición de paro de la búsqueda?. La idea de nuestra propuesta es explorar sólo en las mejores soluciones y que esta exploración se haga de manera mesurada, es decir, no dedicar muchas evaluaciones en explorar las soluciones prometedoras. Si estas soluciones prometedoras no son los caminos que nos llevan al óptimo global, podremos ir tomando otras alternativas mediante el algoritmo de *ED* (en cualquier variante). Pero si las mejores soluciones nos llevan al óptimo global, entonces seguirán siendo exploradas cada vez que se realice la búsqueda local, puesto que seguirán siendo mejores soluciones con el paso del tiempo.

El movimiento de exploración se toma como en la versión original y se muestra en el algoritmo 1, donde un valor de $\alpha = 2$ es recomendado [14]. Nosotros proponemos que este movimiento exploratorio se le aplique al 3 % de la población.

Algorithm 1 Movimiento Exploratorio

1: Inicializar parámetros
2: Escoger un punto de inicio $x^{(0)}$
3: $k = 0$
4: **for** $i = 1$ a 10 **do**
5: Realizar un movimiento exploratorio con $x^{(k)}$ como punto base
6: **if** el movimiento exploratorio = éxito **then**
7: $x^{(k+1)} = x$
8: $k = k + 1$
9: realizar el movimiento de patrones: $x_p^{(k+1)} = x^{(k)} + (x^{(k)} - x^{(k-1)})$
10: **else**
11: $\Delta_i = \Delta_i / \alpha$
12: **end if**
13: **end for**

En este 3 % se encuentran las mejores soluciones. Se aplica solo un 3 % debido a que no tiene caso explorar más soluciones que no son tan prometedoras como las mejores. Así mismo, no recomendamos que el movimiento exploratorio sea solo en la mejor solución, sino tomar más de un camino prometedor. Para el tamaño de la exploración recomendamos que sea obtenido de la variable con menor rango como se muestra en (4):

$$(\text{Límite superior} - \text{límite inferior}) \ /100 \tag{4}$$

Lo anterior es debido a que la exploración dependerá del espacio de búsqueda donde se pueda explorar. La frecuencia de la aplicación de la búsqueda local es aplicada cada generación, con el fin de explorar más veces aquellas mejores soluciones después de cada generación. La parte donde difiere nuestra propuesta a la de Hooke-Jeeves original, es la condición de paro. Nosotros fijamos un número de iteraciones máximo que el algoritmo puede explorar. Lo anterior es realizado con el fin que no explore demasiado, puesto que alguna buena solución aparente puede llevarnos a un óptimo local. Si la solución debe seguir siendo explorada después de las 10 iteraciones propuestas, la siguiente vez que se ejecute este algoritmo de búsqueda, esa solución volverá a ser seleccionada para ser explorada siempre y cuando esté dentro del 3 % de las mejores soluciones. Además con esta restricción se evita la calibración de más parámetros.
El algoritmo de Hooke-Jeeves modificado es presentado en el algoritmo 2.

6. Experimentos y discusión de resultados

Dos experimentos fueron realizados para analizar el desempeño de nuestro enfoque de búsqueda local:

1. Una comparación entre las variantes de *ED* con y sin nuestro enfoque.
2. Una comparación contra los resultados de otro algoritmo que ocupa la búsqueda local de Hooke-Jeeves: MBFOA [9].

Algorithm 2 Método de búsqueda de patrones de Hooke-Jeeves modificado

1: $x = x^{(c)}$
2: $i = 1$
3: Calcular $f = f(x)$, $f^+ = (x_i + \Delta_i)$ y $f^- = (fx_i - \Delta_i)$
4: Encontrar $f_{min} = min(f, f^+, f^-)$ basado en las reglas de factibilidad
5: $x = f_{min}$
6: **if** $i < N$ **then**
7: $i = i + 1$
8: ir al paso 3
9: **end if**
10: **if** $x \neq x^c$ **then**
11: $Exito$
12: **else**
13: $Fallo$
14: **end if**

Ocho de los problemas conocidos [16] son usados en las comparaciones propuestas. Las características de los problemas abordados están resumidos en el cuadro 1. Las definiciones son descritas en [17] y son mostradas a continuación:

1. **g01:**
   Minimizar $f(\overrightarrow{x}) : 5\sum_{i=1}^{4} x_i - 5\sum_{i=1}^{4} x_i^2 - \sum_{i=5}^{13} x_i$ sujeta a:
   $g1(\overrightarrow{x}) = 2x_1 + 2x_2 + x_{10} + x_{11} - 10 \leq 0$
   $g2(\overrightarrow{x}) = 2x_1 + 2x_3 + x_{10} + x_{12} - 10 \leq 0$
   $g3(\overrightarrow{x}) = 2x_2 + 2x_3 + x_{11} + x_{12} - 10 \leq 0$
   $g4(\overrightarrow{x}) = -8x_1 + x_{10} \leq 0$
   $g5(\overrightarrow{x}) = -8x_2 + x_{11} \leq 0$
   $g6(\overrightarrow{x}) = -8x_3 + x_{12} \leq 0$
   $g7(\overrightarrow{x}) = -2x_4 - x_5 + x_{10} \leq 0$
   $g8(\overrightarrow{x}) = -2x_6 - x_7 + x_{11} \leq 0$
   $g9(\overrightarrow{x}) = -2x_8 - x_9 + x_{12} \leq 0$
   Donde $0 \leq x_i \leq 1 (i = 1, ..., 9)$, $0 \leq x_i \leq 100 (i = 10, 11, 12)$ y $0 \leq x_{13} \leq 1$.
2. **g02:**
   Maximizar $f(\overrightarrow{x}) : |\frac{\sum_{i=1}^{n} \cos^4(x_i) - 2\prod_{i=1}^{n} \cos^2(x_i)}{\sqrt{\sum_{i=1}^{n} ix_i^2}}|$ sujeta a:
   $g1(\overrightarrow{x}) = 0{,}75 - \prod_{i=i}^{n} x_i \leq 0$
   $g2(\overrightarrow{x}) = \sum_{i=i}^{n} x_i - 7{,}5n \leq 0$
   Donde $n = 20$ y $0 \leq x_i \leq 10 (i = 1, ..., n)$.
3. **g04:**
   Minimizar $f(\overrightarrow{x}) : 5{,}3578547x_3^2 + 0{,}8356891x_1x_5 + 37{,}293239x_1 - 40792{,}141$ sujeta a:
   $g1(\overrightarrow{x}) = 85{,}334407 + 0{,}0056858x_2x_5 + 0{,}0006262x_1x_4 - 0{,}0022053x_3x_5 - 92 \leq 0$
   $g2(\overrightarrow{x}) = -85{,}334407 - 0{,}0056858x_2x_5 - 0{,}0006262x_1x_4 + 0{,}0022053x_3x_5 \leq 0$
   $g3(\overrightarrow{x}) = 80{,}51249 + 0{,}0071317x_2x_5 + 0{,}0029955x_1x_2 + 0{,}0021813x_3^2 - 110 \leq 0$
   $g4(\overrightarrow{x}) = -80{,}51249 - 0{,}0071317x_2x_5 - 0{,}0029955x_1x_2 - 0{,}0021813x_3^2 + 90 \leq 0$

$g5(\overrightarrow{x}) = 9{,}300961 + 0{,}0047026x_3x_5 + 0{,}0012547x_1x_3 + 0{,}0019085x_3x_4 - 25 \leq 0$
$g6(\overrightarrow{x}) = -9{,}300961 - 0{,}0047026x_3x_5 - 0{,}0012547x_1x_3 - 0{,}0019085x_3x_4 + 20 \leq 0$
Donde $78 \leq x_1 \leq 102$, $33 \leq x_2 \leq 45$ y $27 \leq x_i \leq 45 (i = 3, 4, 5)$.

4. **g06:**
Minimizar $f(\overrightarrow{x}) : (x_1 - 10)^3 + (x_2 - 20)^3$ sujeta a:
$g1(\overrightarrow{x}) = -(x_1 - 5)^2 - (x_2 - 5)^2 + 100 \leq 0$
$g2(\overrightarrow{x}) = (x_1 - 6)^2 - (x_2 + 5)^2 - 82{,}81 \leq 0$
Donde $13 \leq x_1 \leq 100$ y $0 \leq x_2 \leq 100$.

5. **g07:**
Minimizar $f(\overrightarrow{x}) : x_1^2 + x_2^2 + x_1x_2 - 14x_1 - 16x_2 + (x_3 - 10)^2 + 4(x_4 - 5)^2 + (x_5 - 3)^2 + 2(x_6 - 1)^2 + 5x_7^2 + 7(x_8 - 11)^2 + 2(x_9 - 10)^2 + 45$ sujeta a:
$g1(\overrightarrow{x}) = -105 + 4x_1 + 5x_2 - 3x_7 + 9x_8 \leq 0$
$g2(\overrightarrow{x}) = 10x_1 - 8x_2 - 17x_7 + 2x_8 \leq 0$
$g3(\overrightarrow{x}) = -8x_1 + 2x_2 + 5x_9 - 2x_{10} - 12 \leq 0$
$g4(\overrightarrow{x}) = 3(x_1 - 2)^2 + 4(x_2 - 3)^2 + 2x_3^2 - 7x_4 - 120 \leq 0$
$g5(\overrightarrow{x}) = 5x_1^2 + 8x_2 + (x_3 - 6)^2 - 2x_4 - 40 \leq 0$
$g6(\overrightarrow{x}) = x_1^2 + 2(x_2 - 2)^2 - 2x_1x_2 + 14x_5 - 6x_6 \leq 0$
$g7(\overrightarrow{x}) = 0{,}5(x_1 - 8)^2 + 2(x_2 - 4)^2 + 3x_1^2 - x_6 - 30 \leq 0$
$g8(\overrightarrow{x}) = -3x_1 + 6x_2 + 12(x_9 - 8)^2 - 7x_{10} \leq 0$
Donde $-10 \leq x_1 \leq 10 (i = 1, ..., 10)$.

6. **g08:**
Maximizar $f(\overrightarrow{x}) : \frac{\sin^3(2\pi x_1)\sin(2\pi x_2)}{x_1^3(x_1x_2)}$ sujeta a:
$g1(\overrightarrow{x}) = x_1^2 - x_2 + 1 \leq 0$
$g2(\overrightarrow{x}) = 1 - x_1 + (x_2 - 4)^2 \leq 0$

7. **g09:**
Minimizar $f(\overrightarrow{x}) : (x_1 - 10)^2 + 5(x_2 - 12)^2 + x_3^4 + 3(x_4 - 11)^2 + 10x_5^6 + 7x_6^2 + x_7^4 + 4x_6x_7 - 10x_6 - 8x_7$ sujeta a:
$g1(\overrightarrow{x}) = -127 + 2x_1^2 + 3x_2^4 + x_3 + 4x_4^2 + 5x_5 \leq 0$
$g2(\overrightarrow{x}) = -282 + 7x_1 + 3x_2 + 10x_3^2 + x_4 - x_5 \leq 0$
$g3(\overrightarrow{x}) = -196 + 23x_1 + x_2^2 + 6x_6^2 - 8x_7 \leq 0$
$g4(\overrightarrow{x}) = 4x_1^2 + x_2^2 - 3x_1x_2 + 2x_3^2 + 5x_6 - 11x_7 \leq 0$
Donde $-10 \leq x_1 \leq 10 (i = 1, ..., 7)$.

8. **g10:**
Minimizar $f(\overrightarrow{x}) : x_1 + x_2 + x_3$ sujeta a:
$g1(\overrightarrow{x}) = -1 + 0{,}0025(x_4 + x_6) \leq 0$
$g2(\overrightarrow{x}) = -1 + 0{,}0025(x_5 + x_7 - x_4) \leq 0$
$g3(\overrightarrow{x}) = -1 + 0{,}01(x_8 - x_5) \leq 0$
$g4(\overrightarrow{x}) = x_1x_6 + 833{,}33252x_4 + 100x_1 - 83333{,}333 \leq 0$
$g5(\overrightarrow{x}) = -x_2x_7 + 1250x_5 + x_2x_4 - 1250x_4 \leq 0$
$g6(\overrightarrow{x}) = -x_3x_8 + 1250000 + x_3x_5 - 2500x_5 \leq 0$
Donde $100 \leq x_1 \leq 10000$, $1000 \leq x_i \leq 10000, (i = 2, 3)$ y $10 \leq x_i \leq 1000 (i = 4, ..., 8)$.

En los dos experimentos se realizaron 30 ejecuciones independientes por cada variante (con y sin búsqueda). Los parámetros utilizados fueron para todas las

variantes: tamaño de población = 200, CR= 0.5.

Para DE/rand/1 bin y DE/best/1/bin F = 0.5. Para MDE Fa = .4 y Fb = 0.6. Para DECV el porcentaje de vectores factibles = 10 %. Para el método modificado de Hooke-Jeeves $\alpha = 2$ y Δ como se propone en la sección 5 fueron utilizados. El número de evaluaciones máximas fueron 220,000.

Tabla 1. Características principales de los problemas de prueba, n indica la dimensión del problema, ρ es el tamaño estimado de la región factible con respecto a todo el espacio de búsqueda, LI y NI son el número de restricciones de desigualdades lineal y no lineal respectivamente y LE y NE son el número de restricciones de igualdades lineales y no lineales respectivamente.

Num	N	Función	ρ	LI	NI	LE	NE
g01	13	Cuadrática	0.0111 %	9	0	0	0
g02	20	No Lineal	99.9971 %	0	2	0	0
g04	5	Cuadrática	52.1230 %	0	6	0	0
g06	2	Cúbica	0.0000 %	0	2	0	0
g07	10	Cuadrática	0.0003 %	3	5	0	0
g08	2	No Lineal	0.8560 %	0	2	0	0
g09	7	Polinomial	0.5121 %	0	4	0	0
g10	8	Lineal	0.0010 %	0	3	0	0

Los resultados del primer experimento son mostrados en los cuadros 2 y 3, donde algunas variantes de ED, principalmente MDE fueron capaces de encontrar soluciones factibles y soluciones óptimas en la mayoría de los problemas. En los cuadros 2 y 3 se puede observar que se encontraron los óptimos globales cuando se le acopló nuestro método propuesto a alguna variante de *ED*. Las variantes que más óptimos globales encontraron fueron MDE y DE best/1/bin con la búsqueda local propuesta en $g01$, $g04$ y $g08$, y en $g01$, $g08$ y $g09$ respectivamente.

Con base en los cuadros 2 y 3, también se puede notar que en prácticamente todas las funciones, la variante de ED con la búsqueda local propuesta supera a la variante de ED sin búsqueda. También se puede notar en los resultados de este enfoque, que es sensible a los problemas con restricciones de igualdad, puesto que es donde no pudo desempeñarse de mejor manera. Con base en los cuadros 2 y 3, los mejores resultados son obtenidos con las variantes de ED cuando existen más restricciones que cuando tiene menos ($g01$, $g04$, $g09$).

También se afirma con base en las tablas que entre más sea el rango de las variables del problema, las desviación estándar suele ser mayor en nuestro enfoque con en el caso de $g04$ y $g06$.

En el cuadro 4 la comparación contra *MBFOA* es presentada. Se observa que en general, la búsqueda local ayuda a obtener el óptimo global. Se aprecia de igual manera en el cuadro 4 que en unos problemas de prueba $MBFOA$ es

mejor, y en otros problemas nuestra búsqueda con ED es mejor (principalmente con MDE).

7. Conclusiones y trabajo futuro

La implementación de una modificación al algoritmo de búsqueda local de Hooke-Jeeves acoplado con variantes de Evolución Diferencial fue presentada en este documento, con el objetivo de mejorar su desempeño en problemas de optimización numérica con restricciones. La búsqueda local propuesta restringe el número de movimientos exploratorios y de soluciones seleccionadas para aplicarles la búsqueda local, con el fin realizar más evaluaciones con el algoritmo de Evolución Diferencial y no enfocar demasiado la búsqueda en soluciones que aparentemente son prometedoras pero pudieran llevarnos a un óptimo local. Esta propuesta fue probada con las variantes de ED: *DE rand/1/bin*, *DE best/1/bin*, *MDE* y *DECV* sobre once de los problemas bien conocidos.

Los resultados sugieren que una búsqueda local en soluciones prometedoras, ayudan a explorar y explotar de mejor manera esas soluciones, lo que lleva a los algoritmos de Evolución Diferencial a encontrar mejores soluciones. Esta explotación y exploración es debido a que la búsqueda se hace en las N direcciones del problema. Además, la búsqueda local ayuda que explorar y explotar las mejores soluciones debido que no se tiene que esperar a que el algoritmo de ED vuelva a explorar todas las soluciones.

Como trabajo futuro proponemos lo siguiente: La aplicación de búsqueda local en este trabajo se aplica en cada generación al 3 % de la población, siendo ese 3 % las mejores soluciones con base en las reglas de factibilidad. Entonces se podrían hacer pruebas donde no se aplique la búsqueda local hasta que se encuentren soluciones factibles. Lo anterior debido a que cuando de aplica la búsqueda local a soluciones no factibles, se tomará como segundo criterio la aptitud, entonces se irán minimizando los valores de las aptitudes de las soluciones no factibles, y en ese caso de deberá seguir explorando con el algoritmo de Evolución Diferencial.

Por último se propone implementar búsquedas locales en otros paradigmas como inteligencia colectiva o sistemas inmunes artificiales.

Referencias

1. Zbigniew, M.: Genetic Algorithms + Data Structures = Evolution Programs. Springer-Verlag, third edition (1996)
2. Coello, C., Van Veldhuizen, D., Lamont, G.: Evolutionary Algorithms for Solving Multi-Objective Problems.Kluwer Academic Publishers, New York (2002)
3. Mezura, E., Velásquez, J., Coello, C.: Modified Differential Evolution for Constrained Optimization. In: Proceedings of the Congress on Evolutionary Computation, pp. 332–339 (2006)
4. Price, K.: An introduction to differential evolution. In: David Corne, Marco Dorigo, and Fred Glover, editors, New Ideas in Optimization, pp. 79–108, Mc Graw-Hill, UK (1999)

5. Goldberg, D.: Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley Publishing Co., Reading, Massachusetts (1989)
6. Schwefel, H.: Evolution and Optimization Seeking. John Wiley y Sons, New York (1995)
7. Mezura, E., Miranda, M., Gomez, R.: Differential Evolution in constrained numerical optimization: An empirical study. Information Sciences, vol. 180, no. 22, pp. 4223–4262 (2010)
8. Runarsson, T., Yao, X.: Stochastic Ranking for Constrained Evolutionary Optimization. In: IEEE Transactions on Evolutionary Computation, vol. 4, no. 3, pp. 284–294 (2000)
9. Mezura, E., López, E.: Adaptation and local search in the modified bacterial foraging algorithm for constrained optimization. In: Evolutionary Computation (CEC), IEEE Congress on (2012)
10. Menchaca, A., Coello, C.: A new Proposal to hybridize the Nelder-Mead Method to a differential evolution algorithm for constrained oprimization. In: Evolutionary Computation, CEC'09, IEEE Congress on (2009)
11. Zapotecas, S., Coello, C.: A multi-objective Meta-model assisted memetic algorithm with non gradient-based local search. In: GECCO'10, Portland, Oregon, USA (2010)
12. Chen, B., Zeng, W., Lin, U.,Zhang, D.: A New Local Search-Based Multiobjective Optimization Algorithm. IEEE Transactions on Evolutionary Computation, vol. 19, no. 1, pp. 50–73 (2015)
13. Ann Shim, V., Chen Tan, K., Tang, H.: Adaptive Memetic Computing for Evolutionary Multiobjective Optimization. Cybernetics, IEEE Transactions on, vol. 45, no. 4, pp. 610–621 (2015)
14. Deb, K.: Optimization for Engineering Design, Algorithms and Examples. Prentice-Hall, India (1995)
15. Hooke, R., Jeeves, T.: Direct search, solution of numerical and statistical problems. J. ACM, vol. 8, no. 2, pp. 212–229 (1961)
16. Mezura, E., Coello, C.: A Simple Multimembered Evolution Strategy to Solve Constrained Optimization Problems. In: IEEE Transactions on Evolutionary Computation, vol. 9, no. 1, pp. 1–17 (2005)
17. Runarsson, T., Yao, X.: Stochastic Ranking for Constrained Evolutionary Optimization. IEEE Transactions on Evolutionary Computation, vol. 4, no. 3, pp. 284–294 (2000)

Tabla 2. Comparación de variantes de ED: $DE/rand/1bin$ y $DE/best/1/bin$ sin el método propuesto contra variantes ED: $DE/rand/1bin$ y $DE/best/1/bin$ con el método de búsqueda propuesto . MLS significa ($ModifiedLocalSearch$), B es el mejor, M es la media, W es el peor, SD es la desviación estándar, PE es el promedio de evaluaciones.

Óptimo	DE-rand/1/bin	DE-rand/1/bin **(MLS)**	DE-best/1/bin	DE-best/1/bin **(MLS)**
g01	B: -13.799804	**B:-15.000**	**B:-15.000**	**B:-15.000**
-15.000	M: -13.365909	M:-14.9999	M:-14.352087	**M:-15.000**
	W: -12.414840	W: -14.999997	W: -12.814961	W: -14.99999
	SD: 0.606441	SD: 1E-6	SD:0.053651	SD: 5E-6
	PE: 220000	PE: 165574.67	PE:207069.32	**PE: 110638.708**
g02	B: -.7952141	B: -.7997510	B: -.8012429	B: -.8032455
-.8036191	M: -.7845126	M: -.795364	M: -.7824035	M: -.8015653
	W: -.775135	W: -.782245	W: -.726512	W: -.7920110
	SD: .3E-4	SD: .5E-5	SD: .03481	SD: .0056
	PE: 220000	PE: 220000	PE: 220000	PE: 220000
g04	B: -30659.5210	B: -30663.2145	B: -30662.5412	B: -30664.4368
-30665.5386	M: -30653.3678	M: -30655.2752	M: -30660.5547	M: -30662.1547
	W: -30650.3250	W: -30653.558	W: -30655.3690	W: -30660.5498
	SD: 4.5864	SD: 5.2926	SD: 3.6074	**SD: 2.0016**
	PE: 220000	PE: 220000	PE: 220000	PE: 220000
g06	B: -6068.3124	**B: -6961.81387**	B: -6038.00426	B: -6573.4572
-6961.81387	M: -5732.3644	M: -6325.3249	M: -5953.3654	M: -6181.7630
	W: -5521.9127	W: -5783.1378	W: -5682.9346	W: -5848.7244
	SD: 275.5913	SD: 589.3315	SD: 185.8932	SD: 362.6028
	PE: 220000	**PE: 219989.6584**	PE: 220000	PE: 220000
g07	B: 33.7865	B: 33.2865	B: 33.8963	B: 32.9634
24.3062	M: 34.2301	M: 33.6777	M: 33.2665	M: 33.4112
	W: 34.6255	W: 34.3665	W: 34.7456	W: 34.3512
	SD: 0.4197	SD: 0.5467	SD: 0.7422	SD:0.7082
	PE: 220000	PE: 220000	PE: 220000	PE: 220000
g08	B: -0.0904421	B: -0.09582503	B: -0.0955922	**B: -0.09582504**
-0.09582504	M: -0.09445202	M: -0.09582250	M: -0.0953254	M: -0.0957852
	W: -0.0726796	W: -0.09572035	W: -0.0949350	W: -0.0955210
	SD: 0.0115	SD: 5E-5	SD: 3E-4	SD: 1E-4
	PE: 220000	PE: 220000	PE: 220000	**PE: 219923.265**
g09	B: 681.502212	B: 681.168569	B: 681.079512	**B: 680.63005**
680.63005	M: 681.736654	M: 681.39645	M: 681.23487	M: 680.98856
	W:681.874212	W: 681.46584	W: 681.632254	W: 681.08658
	SD: 0.1880	SD: 0.1555	SD: 0.285	SD: 0.2403
	PE: 220000	PE: 220000	PE: 220000	**PE: 219956.015**
g10	B: 7210.65585	B: 7125.32963	B:7217.56321	B:7163.325122
7049.248020	M: 7246.5684	M: 7169.54523	M: 681.23487	M:7178.563324
	W:7259.22447	W:7256.56478	W:7232.21001	W:7210.25485
	SD: 25.1952	SD:66.7708	SD:34.9738	SD: 23.9407
	PE: 220000	PE: 220000	PE: 220000	PE: 220000

Tabla 3. Comparación de nuestros mejores resultados con los trabajos relacionados con variantes de ED, MDE y $DECV$ sin el método propuesto, contra variantes de ED, MDE y $DECV$ con el método de búsqueda propuesto. MLS significa ($ModifiedLocalSearch$), B es el mejor, M es la media, W es el peor, SD es la desviación estándar, PE es el promedio de evaluaciones.

Óptimo	MDE	MDE **(MLS)**	DECV	DECV **(MLS)**
g01	**B: -15.000**	**B: -15.000**	**B: -15.000**	**B: -15.000**
-15.000	**M: -15.000**	**M: -15.000**	M:-14.352087	**M: -15.000**
	W: -15.000	**W: -15.000**	W: -12.814961	W: -14.99999
	SD: 0	**SD:0**	SD:0.053651	SD: 5E-10
	PE: 200819.306	PE: 179356.675	PE: 208596.124	**PE: 156814.301**
g02	B: -.7964518	B: -.8034857	B: -.7955210	B: -.80154552
-.8036191	M: -.791150	M: -.8025412	M: -.7845223	M:-.7999514
	W:-.7851235	W: -.8015456	W: -.77619085	W:-.7922214
	SD: .3E-5	SD: 0.009108	SD: .0845	**SD: 7E-8**
	PE: 220000	PE: 220000	PE: 220000	PE: 220000
g04	B: -30633.3541	**B: -30665.5386**	B: -30638.2133	B: -30663.2154
-30665.5386	M: -30621.3547	M: -.30642.2457	M: -30627.3641	M: -30645.368
	W: -30583.3508	W: -30635.9288	W: -30615.3665	W: -30638.3545
	SD: 26.104	SD: 13.272	SD:11.5053	SD:12.8983
	PE: 220000	**PE: 219.669**	PE: 220000	PE: 220000
g06	B: -6228.8601	B: -6558.4572	B: -6018.2014	B: -6032.5844
-6961.81387	M: -5763.1574	M: -5836.7596	M: -5953.3654	M: -5925.3564
	W: -5322.3407	W: -5543.8102	W: -5472.5421	W: -5793.2654
	SD: 452.7075	SD: 522.1340	SD: 274.2695	**SD: 119.4095**
	PE: 220000	220000	PE: 220000	PE: 220000
g07	B: 33.6332	B: 28.3687	B: 33.5632	B: 33.2765
24.3062	M: 34.0253	M: 33.6366	M: 33.7445	M: 33.7963
	W: 34.2557	W: 34.1954	W: 33.9654	W: 34.5221
	SD: 0.3147	SD: 3.2148	**SD: 0.2014**	SD: 0.6256
	PE: 220000	PE: 220000	PE: 220000	PE: 220000
g08	B: -0.09582503	**B: -0.09582504**	B: -0.09582503	B: -0.09582501
-0.09582504	M: -0.09582501	M: -0.09582502	M: -0.09582369	M: -0.0956724
	W: -0.09581520	W: -0.09582501	W: -0.09575301	W: -0.09436584
	SD: 5 E -6	**SD: 1 E-8**	SD: 4E-5	SD: 8 E-4
	PE: 220000	**PE: 219967.124**	PE: 220000	PE: 220000
g09	B:680.949523	B: 680.88060	B: 680.999535	B: 681.003254
680.63005	M: 681.25548	M: 680.99542	M: 681.00953	M: 681.15464
	W: 681.23542	W: 681.09951	W: 681.116574	W: W: 681.65217
	SD:0.1711	SD: 0.1094	**SD: 0.0648**	SD: 0.3394
	PE: 220000	PE: 220000	PE: 220000	PE: 220000
g10	B:7265.33478	B:7122.48525	B:7277.35665	B:B:7284.34526
7049.248020	M:7272.76654	M: 7189.6542	M:7283.12589	M:7288.45221
	W:7259.22447	W: 7201.1745	W:7296.54752	W:72.96.54244
	SD:16.7426	SD:42.4978	SD: 9.8464	**SD: 2.9040**
	PE: 220000	PE: 220000	PE: 220000	PE: 220000

Tabla 4. Comparación de nuestros mejores resultados contra $MBFOA - AS - LS$.

Óptimo	MBFOA-AS-LS [9]	Mejores resultados (MLS)
g01	**B: -15.000**	**B: -15.000**
-15.000	M: -14.685	**M: -15.000**
	W: -12.833	**W: -15.000**
	SD: 7.2E-01	**SD:0**
		$MDE(MLS)$
g02	B: -.7925284	B: -.8034857
-.8036191	M: -0.62220	M:-.8025412
	W: -12.833	W: -.8015456
	SD: 1.09E-1	**SD: .009108**
		$MDE(MLS)$
g04	B: -30665.539	**B: -30665.5386**
-30665.5386	M: -30665.539	M: -.30642.2457
	W: -30665.539	W: -.8015456
	SD:1.48E-11	SD: 13.272
		$MDE(MLS)$
g06	**B: -6961.814**	B:-6961.81387
-6961.81387	**M: -6961.814**	M: -6325.3249
	W: -6961.814	W: -.8015456
	SD: 3.13E-6	SD: 589.3315
		$DErand/1/bin(MLS)$
g07	B: 24.349	B:28.3687
24.3062	M: 24.461	M: 33.6366
	W: 24.623	W: -.8015456
	SD: 8.14E-2	SD: 3.2148
		$MDE(MLS)$
g08	B: -0.095825	**B: -0.09582504**
-0.09582504	M: -0.095825	M: -0.0957852
	W: -0.095825	W: -0.0955210
	SD: 4.23E-17	SD: 1E-4
		$DEbest/1/bin(MLS)$
g09	B: 680.633	**B: 680.63005**
680.63005	M: 680.690	M: 680.98856
	W: 680.837	W: 681.08658
	SD: 6.61E-02	SD: 0.2403
		$DEbest/1/bin(MLS)$
g10	B:7051.648	B:7122.48525
7049.248020	M:7077.555	M: 7189.6542
	W:7142.653	W: 7201.1745
	SD: 2.5E-1	SD:42.4978
		$MDE(MLS)$

Un algoritmo para calcular #2SAT

Marco A. López Medina[1], J. Raymundo Marcial-Romero[1], Guillermo De-Ita[2], José A. Hernández[1]

[1] Facultad de Ingeniería, UAEM, México

[2] Facultad de Ciencias de la Computación, BUAP, Puebla, México

valgirmanda@gmail.com, jrmarcialr@uaemex.mx, deita@cs.buap.mx, xoseahernandez@uaemex.mx

Resumen. El problema de conteo de modelos en fórmulas booleanas pertenece a la clase #P-completo. Por tal motivo, no existe algoritmo qué, de forma eficiente, calcule el número exacto de modelos de una fórmula booleana. En este artículo, se presenta una implementación para contar el número de modelos de una fórmula booleana basada en su representación mediante grafo. Así mismo se mostrará que, para ciertas topologías del grafo, el algoritmo presentado establece el conteo de modelos de forma eficiente comparado con otras implementaciones reportadas en la literatura.

Palabras clave: #SAT, combinatoria, teoría de grafos.

1. Introducción

El problema de conteo de modelos, mejor conocido por #SAT consiste en calcular el número de modelos de una fórmula booleana dada, es decir, el número de asignaciones de verdad para las cuales la fórmula se evalúa como verdadera. En este caso el problema que se aborda directamente es #2SAT que es el conteo de modelos sobre una fórmula booleana en forma Normal Conjuntiva de dos variables (2-Conjunctive Normal Form). El problema #SAT pertenece a la clase #P completo, por tanto no existe algoritmo eficiente que resuelva el problema en general. Existen implementaciones que permiten calcular los modelos de una fórmula dada como por ejemplo, relsat [5] la cual es una herramienta basada en el algoritmo DPLL (Davis-Putnam-Logemann-Loveland algorithm) de búsqueda exhaustiva de asignaciones. La mejora que provee en comparación a otras herramientas es el poder procesar rápidamente fórmulas disjuntas, es decir, que las variables de una fórmula no intervengan en alguna otra. Cachet [9] es una herramienta basada en el modelo de conteo *Caching* que es sucesora del árbol de búsqueda de un modelo de conteo DPLL; ajustando variables y simplificando la fórmula, ya que se pueden encontrar sub-fórmulas que han aparecido en una rama del árbol de búsqueda. Si esto sucede, no es necesario volver a contar los

modelos, sólo es necesario saber el conteo de la rama de búsqueda anterior y utilizarlo.

Por otro lado sharpSAT [1] está basada en el modelo *caching* e implementa un mejor razonamiento en cada nodo que evalúa. Utiliza una técnica conocida como *look ahead* que permite hacer una prueba sobre alguna variable en la fórmula para ver si es necesario que sea siempre verdadera o falsa, permitiendo que solo se cuenten los modelos necesarios para el resto de la fórmula.

En este artículo, se propone una estrategía para el conteo de modelos convirtiendo la fórmula de entrada en un grafo. Así mismo, se presentan fórmulas booleanas para las cuales las implementaciones existentes no son capaces de regresar un resultado y la que aquí se presenta lo puede realizar.

2. Preliminares

Los conceptos utilizados pueden consultarse en [7]. Sea $X = x_1, ..., x_n$ un conjunto de n variables booleanas (es decir pueden tomar únicamente dos valores de verdad). Una literal es una variable x_i o la variable negada $\bar{x}_i$. Una cláusula es una disjunción de literales distintas. Una fórmula booleana en forma normal conjuntiva (CNF, Conjunctive Normal Form) F es una conjunción de cláusulas.

Sea $v(Y)$ el conjunto de variables involucradas en el objeto Y, donde Y puede ser una literal, una cláusula o una fórmula booleana. Por ejemplo, para la cláusula $c = \{x_1 \vee \bar{x}_2\}, v(c) = \{x_1, x_2\}$ y Lit(F) es el conjunto de literales que aparecen en una CNF F.

Una asignación s para F es una función booleana $s : v(F) \to \{0, 1\}$. Una asignación puede también considerarse como un conjunto de pares de literales no complementario. Si $x^e \in s$, siendo s una asignación, entonces s convierte x^e en verdadero y x^{1-e} en falso, $e \in \{0, 1\}$. Considerando una cláusula c y una asignación s como un conjunto de literales, se dice que c se satisface por s sí y solo si $c \cap s \neq \emptyset$ y si para toda $x^e \in c$, $x^{1-e} \in s$ entonces s falsifica a c.

Sea F una fórmula booleana en CNF, se dice que F se satisface por la asignación s si cada cláusula en F se satisface por s. Por otro lado, se dice que F se contradice por s si al menos una cláusula de F se falsifica por s. Un modelo de F es una asignación para $v(F)$ tal que satisface F. Se denota como $SAT(F)$ al conjunto de modelos de la fórmula F.

Dada una fórmula F en CNF, SAT consiste en determinar si F tiene un modelo, mientras que #SAT consiste en contar el número de modelos que tiene F sobre $v(F)$. Por otro lado, #2SAT denota #SAT para fórmulas en 2-CNF.

2.1. El grafo restringido de una 2-CNF

Existen algunas representaciones gráficas de una Forma Normal Conjuntiva, por ejemplo el grafo primal signado o también conocido como grafo restringido.

Sea F una 2-CNF, su grafo restringido se denota por $G_F = (V(F), E(F))$ con $V(F) = v(F)$ y $E(F) = \{\{v(x), v(y)\} : \{x \vee y\} \in F\}$, esto es, los vertices de G_F son las variables de F, y para cada cláusula $\{x \vee y\}$ en F existe una arista

$\{v(x), v(y)\} \in E(F)$. Para $x \in V(F), \delta(x)$ denota su grado, es decir el número de aristas incidentes en x. Cada arista $c = \{v(x), v(y)\} \in E(F)$ se asocia con un par (s_1, s_2) de signos, que se asignan como etiquetas de la arista que conecta las variables de la cláusula. Los signos s_1 y s_2 pertenecen a las variables x y y respectivamente. Por ejemplo la cláusula x^0, y^1 determina la arista con etiqueta "$x \overset{-+}{—} y$", que es equivalente a la arista "$y \overset{+-}{—} x$".

Sea $S = \{+, -\}$ un conjunto de signos. Un grafo con aristas etiquetadas en S es el par (G, ψ), dónde $G = (V, E)$ es un grafo restringido, y ψ es una función con dominio E y rango S. $\psi(e)$ se denomina a la etiqueta de la arista $e \in E$. Sea $G = (V, E, \psi)$ un grafo restringido con aristas etiquetadas en $S \times S$ y x y y nodos en V, si $e = \{x, y\}$ es una arista y $\psi(e) = (s, s')$, entonces $s(s')$ es el signo adyacente a $x(y)$.

Sea G un grafo conectado de n vértices, un árbol de expansión de G es un subconjunto de $n-1$ aristas tal que forman un árbol de G. Se denomina co-árbol al subconjunto de aristas que son el complemento de un árbol.

3. Conteo de modelos en grafos acíclicos

El conteo de modelos para fórmulas booleanas cuyo grafo restringido es acíclico se puede realizar en tiempo polinomial [4]. A continuación se presenta un algoritmo que, mediante un recorrido en preorder, permite contar modelos en un grafo acíclico.

3.1. #2SAT para fórmulas en 2-CNF que representan un camino

Se dice que un grafo G_F representa un camino para una 2-CNF F, si

$$F = \{C_1, C_2, \ldots, C_m\} = \{\{x_1^{\epsilon_1} \vee x_2^{\delta_1}\}, \{x_2^{\epsilon_2} \vee x_3^{\delta_2}\}, \ldots, \{x_m^{\epsilon_m} \vee x_{m+1}^{\delta_m}\}\},$$

donde $\delta_i, \epsilon_i \in \{0, 1\},\ i \in [1..m]$. Sea f_i una familia de cláusula de la fórmula F construida como sigue: $f_1 = \emptyset; f_i = \{C_j\}_{j<i}$, $i \in [1..m]$. Note que $n = |v(F)| = m + 1$, $f_i \subset f_{i+1}$, $i \in [1..m-1]$. Sea $SAT(f_i) = \{s : s \text{ satisface } f_i\}$, $A_i = \{s \in SAT(f_i) : x_i \in s\}$, $B_i = \{s \in SAT(f_i) : \overline{x}_i \in s\}$. Sea $\alpha_i = |A_i|;\ \beta_i = |B_i|$ y $\mu_i = |SAT(f_i)| = \alpha_i + \beta_i$.

Para cada nodo $x \in G_F$ se calcula el par (α_x, β_x), donde α_x indica el número de veces que la variable x toma el valor 'verdadero' y β_x indica el número de veces que la variable x toma el valor 'falso' en el conjunto de modelos de F. El primer par es $(\alpha_1, \beta_1) = (1, 1)$ ya que x_1 puede tomar el valor verdadero o falso para satisfacer a f_1. Los pares (α_x, β_x) asociados a cada nodo x_i, $i = 2, \ldots, m$ se calculan de acuerdo a los signos (ϵ_i, δ_i) de las literales en la cláusula c_i por la siguiente ecuación de recurrencia:

$$(\alpha_i, \beta_i) = \begin{cases} (\beta_{i-1}, \alpha_{i-1} + \beta_{i-1}) & \text{if } (\epsilon_i, \delta_i) = (-, -) \\ (\alpha_{i-1} + \beta_{i-1}, \beta_{i-1}) & \text{if } (\epsilon_i, \delta_i) = (-, +) \\ (\alpha_{i-1}, \alpha_{i-1} + \beta_{i-1}) & \text{if } (\epsilon_i, \delta_i) = (+, -) \\ (\alpha_{i-1} + \beta_{i-1}, \alpha_{i-1}) & \text{if } (\epsilon_i, \delta_i) = (+, +) \end{cases} \tag{1}$$

Note que, si $F = f_m$ entonces $\#SAT(F) = \mu_m = \alpha_m + \beta_m$. Se denota con $\rightarrow$ la aplicación de una de las cuatro reglas de recurrencia (1).

Ejemplo 1 *Sea* $F = \{(x_1, x_2), (\overline{x}_2, \overline{x}_3), (\overline{x}_3, \overline{x}_4), (x_4, \overline{x}_5), (\overline{x}_5, x_6)\}$ *una fórmula en 2-CNF cuyo grafo restringido representa un camino. Las series* $(\alpha_i, \beta_i), i \in [1..6]$*, se calculan como:* $(\alpha_1, \beta_1) = (1, 1) \rightarrow (\alpha_2, \beta_2) = (2, 1)$ *ya que* $(\epsilon_1, \delta_1) = (+, +)$*, y la regla 4 tiene que aplicarse. En general, aplicando la regla correspondiente de la recurrencia (1) de acuerdo a los signos de* (ϵ_i, δ_i)*,* $i = 2, ..., 5$*, se tiene que* $(2, 1) \rightarrow (1, 3) \rightarrow (3, 4) \rightarrow (3, 7) \rightarrow (\alpha_6, \beta_6) = (10, 7)$*, y entonces,* $\#SAT(F) = \mu_6 = \alpha_6 + \beta_6 = 10 + 7 = 17$.

3.2. Conteo en grafos acíclicos

Sea F una fórmula en 2-CF donde su grafo asociado G_F es acíclico. Se puede asumir que G_F tiene un nodo raíz, un recorrido del grafo permite generar un árbol que tiene un nodo raíz ya que es acíclico. Un árbol tiene tres clases de nodos: raíz, interior y hojas.

Se denota con (α_v, β_v) el par asociado con el nodo v ($v \in G_F$). Se calcula $\#SAT(F)$ mientras se recorre G_F en post-order con el siguiente algoritmo.
Algoritmo Conteo_Modelos_para_arbol(G_F)
Entrada: G_F - un grafo que representa un árbol.
Salida: El número de modelos de F
Procedimiento:
Recorrer G_F en post-order, para cada nodo $v \in G_F$, asignar:

1. $(\alpha_v, \beta_v) = (1, 1)$ si v es un nodo hoja en G_F.
2. Si v es un nodo interior con una lista de nodos hijos asociados, i.e., $u_1, u_2, ..., u_k$ son los nodos hijos de v, una vez que se han visitado los hijos, los pares calculados son $(\alpha_{u_j}, \beta_{u_j})$ $j = 1, ..., k$. Sean $e_1 = v^{\epsilon_1} u_1^{\delta_1}, e_2 = v^{\epsilon_2} u_2^{\delta_2}, \ldots, e_k = v^{\epsilon_k} u_k^{\delta_k}$ las aristas que conectan v con cada uno de sus nodos hijos. El par $(\alpha_{e_j}, \beta_{e_j})$ se calcula para cada arista e_j basado en la recurrencia (1) donde $\alpha_{e_j - 1}$ is α_{u_j} y $\beta_{e_j - 1}$ es β_{u_j} para $j = 1, \ldots k$. Entonces, sea $\alpha_v = \prod_{j=1}^{k} \alpha_{e_j}$ y $\beta_v = \prod_{j=1}^{k} \beta_{e_j}$. Note que este paso incluye el caso en que v tiene solo un nodo como hijo.
3. Si v es el nodo raíz de G_F entonces regresar $(\alpha_v + \beta_v)$.

Este procedimiento regresa el número de modelos de F en tiempo $O(n + m)$ [8] el cual es el tiempo para recorrer un grafo en post-order.

Ejemplo 2 *Si* $F = \{(x_1, x_2), (x_2, x_3), (x_2, x_4), (x_2, x_5), (x_4, x_6), (x_6, x_7), (x_6, x_8)\}$ *es una fórmula en 2-CF, si* x_1 *es el nodo raíz, después de realizar un recorrido post-order el número de modelos a cada nivel del árbol se muestran en la Figura 1. El procedimiento* $Conteo\_Modelos\_para\_arbol$ *regresa* $\alpha_{x_1} = 41$*,* $\beta_{x_1} = 36$ *y el número total de modelos es:* $\#SAT(F) = 41 + 36 = 77$.

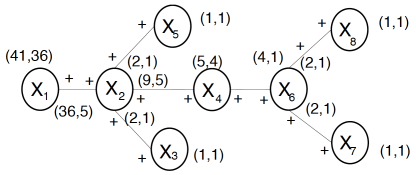


Fig. 1. Conteo de modelos en grafos que representan árboles

4. Conteo de modelos en grafos cíclicos

El problema #2SAT es #P-completo, para fórmulas cuyo grafo restringido es cíclico. En [4] se presenta un algoritmo para el conteo de modelos basado en el números de aristas del co-árbol. Por cada arista del co-árbol el algoritmo duplica el grafo de entrada eliminando la arista del co-árbol. Por lo tanto la complejidad del algoritmo esta dada por 2^r donde r es el número de aristas del co-árbol. Sin embargo, ya que las aristas del co-árbol son aquellas que forman cíclos, un grafo con r cíclos requiere 2^r árboles.

En esta sección mostramos cómo se puede disminuir el número de árboles generados para así hacer más eficiente el procedimiento para cierto tipo de grafos. Los árboles de expansión que se mencionan a continuación se construyen utilizando el método Depth First Search (DFS). El método DFS garantiza que todas las aristas del co-árbol son de retroceso es decir, que si se incorporan al árbol conectan nodos descendientes con nodos ancestros y no nodos que están en diferentes ramas del árbol. Por tal motivo se puede dividir el grafo original en r sub-árboles cada uno con su respectivo co-árbol, donde r es el número de nodos hijos asociados al nodo que se eligió como raíz en el grafo original.

Definición 1 *Sea F una fórmula cuyo grafo restringido $G_F = \rho(F)$ es cíclico. Sea T el árbol de expansión de G_F y $\overline{T}$ su co-árbol; se construye una familia de conjuntos $\mathcal{P}$ de $\overline{T}$ como sigue: un par de aristas $e_1, e_2 \in \overline{T}$ pertenecen al mismo conjunto sí y sólo si $path(e_1) \cap path(e_2) \neq \emptyset$ donde $path(e_i), i = 1, 2$ es el camino de los vértices de las aristas e_1 y e_2 en el árbol T.*

Lema 1 *La familia de conjuntos $\mathcal{P}$ de la Definición 1 es una partición de $\overline{T}$.*

***Prueba* 1** *Sean $X, Y \in \mathcal{P}$, es obvio que $X \cap Y \neq \emptyset$ ya que para cada par de aristas $e_1, e_2 \in \overline{T}$ hay un único camino en T. Dado que cada arista $e \in \overline{T}$ pertenece a una única partición entonces $\bigcup_{X \in \mathcal{P}} X = \overline{T}$*

Ahora se puede construir una partición de G_F.

Definición 2 *1. Para cada $P \in \mathcal{P}$, se construye un subgrafo como sigue: $\forall e \in P$,*

$$G_P(V(path(e)), E(path(e) \cup e))$$

2. Se define $G_R = G_F \setminus \bigcup G_{P \in \mathcal{P}}$.

Lema 2 *Los conjuntos $E(G_P), P \in \mathcal{P}$ junto con $E(G_R)$ forman una partición de $E(G_F)$.*

Lema 3 *Para cada par de grafos G_{P_1}, G_{P_2}, del Lema 2, ya sea que $V(G_{P_1}) \cap V(G_{P_2}) = \emptyset$ o $V(G_{P_1}) \cap V(G_{P_2}) = \{v\}$ es decir es vacío o un conjunto de un solo elemento.*

***Prueba* 2** *Por contradicción, supóngase que $V(G_{P_1}) \cap V(G_{P_2}) \neq \emptyset$ y $V(G_{P_1}) \cap V(G_{P_2}) \neq \{v\}$ lo que significa que hay al menos dos vértices v_1, v_2 en la intersección, lo que significa que la arista $e = (v_1, v_2)$ pertenece a la intersección contradiciendo la hipótesis de que G_{P_1} and G_{P_2} tienen un conjunto de aristas disjuntas.*

Teorema 1 *Para cada par de grafos G_{P_1}, G_{P_2} del lema 2*

1. *Si $G_{P_1} \cap G_{P_2} = \emptyset$ entonces los modelos que representa G_{P_1} son independientes de los modelos que representa G_{P_2} es decir $Models(G_{P_1} \cup G_{P_2}) = Models(G_{P_1}) \times Models(G_{P_2})$.*
2. *Si $G_{P_1} \cap G_{P_2} = \{v\}$ entonces*

$$\begin{aligned} Modelos(G_{P_1} \cup G_{P_2}) = Modelos(G_{P_1}|_{v^1}) \times & Modelos(G_{P_2}|_{v^1}) \\ + & \\ Modelos(G_{P_1}|_{v^0}) \times & Modelos(G_{P_2}|_{v^0}) \end{aligned}$$

***Prueba* 3** *1. Si $G_{P_1} \cap G_{P_2} = \emptyset$ ninguno de los vértices o aristas son compartidos. Ya que cada vértice de los grafos representan una variable de la fórmula de entrada, $\rho^{-1}(G_{P_1}) \cap \rho^{-1}(G_{P_2}) = \emptyset$, es decir no hay variables en común de las fórmulas representadas por cada sub-grafo. Es bien sabido que los modelos de fórmulas sin variables en común se pueden calcular como el producto de los modelos de cada fórmula.*

2. *Que la intersección sea un conjunto con un solo elemento significa que si $F_1 = \rho^{-1}(G_{P_1})$ y $F_2 = \rho^{-1}(G_{P_2})$ entonces $\nu(F_1) \cap \nu(F_2) = \{x_1\}$, es decir hay una sola variable en común para ambas sub-fórmulas. Una estrategia branch and bound se puede utilizar, donde una rama cuenta los modelos donde x_1 se fija con el valor verdadero y la otra rama cuenta los modelos donde x_1 se fija con el valor falso en ambas sub-fórmulas y al final se suman los modelos. Una vez que x_1 se fijo con un valor ya sea verdadero o falso, no existen mas variables en común entre las sub-fórmulas , así que por 1, sus modelos se pueden calcular de forma independiente y posteriormente realizar el producto.*

Del Teorema 1 se puede concluir que el número de modelos de cada uno de los $G_{P \in \mathcal{P}}$ se puede calcular de forma independiente.

Dado que G_F es conectado, hay sub-grafos G_{P_1} y G_{P_2} tal que $V(G_{P_1}) \cap V(G_{P_1}) = \emptyset$ por lo que debe existir un camino en G_R que los una. Afortunadamente, los modelos de un camino se pueden calcular en tiempo polinomial con el procedimiento que se presentó en la sección 3.1.

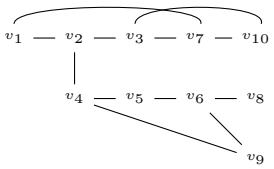
Así, se concluye que

Teorema 2 *Sea F una fórmula en 2-CNF, sea $G = \rho(F)$ su grafo restringido. Si P es una partición de G como se estableció en la Definición 1 y $\overline{T_i}, i = 1, \ldots r$ son las particiones que contienen aristas en el co-árbol, entonces la complejidad en tiempo para calcular $Modelos(G)$ es $O(2^{max\{|E(\overline{T_i})|\}} \cdot poly(|E(T)|))$, donde poly es una función polinomial.*

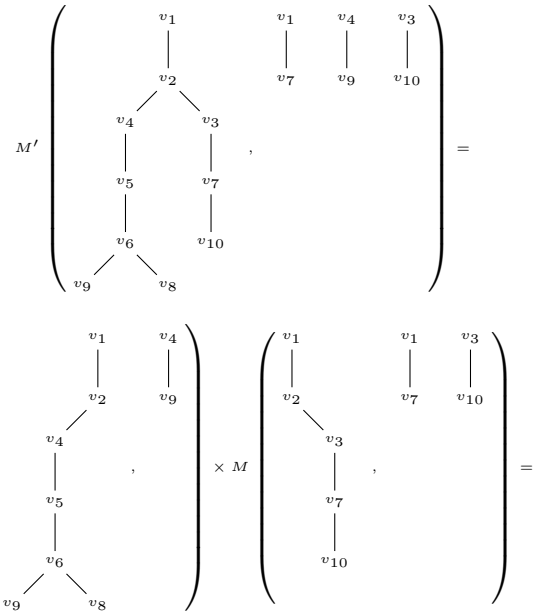
***Prueba* 4** *Por el Teorema 1 cada $\overline{T_i}$ se puede calcular de forma independiente. Así la duplicación máxima del grafo esta dado por $k = max\{|\ E(\overline{T_i})|\}$.*

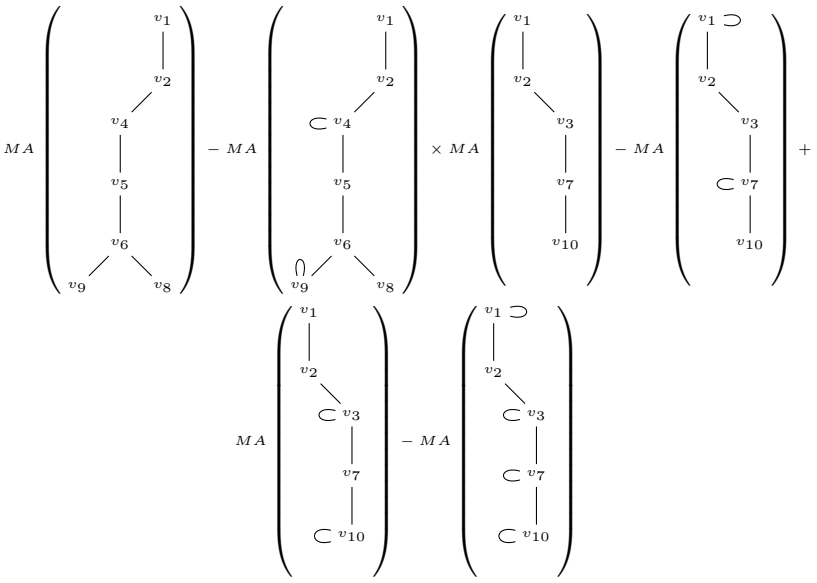
4.1. Ejemplo

Sea $F = \{\{v_1, v_2\}, \{v_1, v_7\}, \{v_2, v_4\}, \{v_2, v_3\}, \{v_3, v_7\}, \{v_3, v_{10}\}, \{v_4, v_5\}, \{v_4, v_9\}, \{v_5, v_6\}, \{v_7, v_{10}\}, \{v_6, v_8\}, \{v_6, v_9\}\}$. El grafo restringido $\rho(F)$ es (se omiten los signos ya que todos son positivos):



La aplicación de DFS del grafo restringido da la siguiente descomposición en árbol y co-árbol donde la artista del co-árbol (v_4, v_9) pertenece a una de las particiones y las aristas del co-árbol $(v_1, v_7), (v_3, v_{10})$ pertenecen a la otra partición. Así sus modelos se pueden calcular de forma independiente como se muestra abajo.





5. Resultados

De las herramientas reportadas en la literatura, aquella que ha dado mejores resultados para el conteo de modelos ha sido sharpSAT, por lo tanto en este artículo se realiza una comparativa con esta herramienta.

El algoritmo implementado, construye el árbol de expansión y posteriormente, divide el árbol de expansión generado en sub-árboles, cada uno con sus respectivos co-árboles para realizar el conteo de modelos. Por cada sub-árbol generado se aplica el procedimiento descrito en la sección anterior y con los árboles generados, se procede a contar sus modelos. Ya que cada sub-árbol se realiza por separado, la complejidad del cálculo disminuye; por ejemplo, en la prueba número 6 de la tabla 1, que contiene 30 cíclos, en lugar de tener 2^{30} árboles se generan dos sub-grafos con 2^{15} cíclos cada una por lo que el número de árboles generados es $2^{15} + 2^{15} = 2^{16}$ árboles en lugar de 2^{30}.

En la tabla 2 se muestran pruebas con grafos que tienen un mayor número de ciclos desde 120 hasta 26660. Ninguno de los algoritmos que se basa en el número de ciclos podría dar una respuesta en un tiempo considerable ya que por ejemplo la primera prueba requeriría 2^{120} cálculos. Sin embargo, ya que el grafo se puede particionar de tal forma que cada sub-grafo tenga únicamente 5 ciclos, se pueden hacer los cálculos de forma independiente para finalmente sumar los resultados de cada uno de ellos. Lo mismo sucede con las otras pruebas que se presentan en la tabla 2. Cabe mencionar que, dé los grafos reportados en la tabla 2 ninguna de las herramientas descritas en la introducción fue capaz de regresar un resultado por lo tanto se omitió la columna de sharpSAT para no dejarla vacía.

Tabla 1. Pruebas del algoritmo general en grafos que contienen un número pequeño de ciclos

Nodos	Cláusulas	Ciclos	Sharp SAT		Este artículo	
			Modelos	Tiempo (s)	Modelos	Tiempo (s)
13	15	3	401	0	401	0
19	18	0	15660	0	15660	0
19	33	15	3512	0.1	3512	19
19	34	15	-	-	5520	18
37	51	15	-	-	28058400	18
37	66	30	-	-	6294080	37
60	59	0	1.33×10^{13}	0	1.33×10^{13}	0
70	69	0	1.85×10^{16}	0	1.85×10^{16}	0
85	84	0	4.38×10^{19}	0	4.38×10^{19}	0

Tabla 2. Grafos con contienen un mayor número de ciclos

#Nodos	# Cláusulas	# Sub-árboles	Ciclos por Sub-árbol	Ciclos	Modelos
125	240	24	5	120	1.96319612718888E+020
250	490	49	5	245	2.57058291067303E+041
500	990	99	5	495	4.62068181689974E+083
1000	1990	199	5	995	1.49454832733069E+168
2000	3990	399	5	1995	1.56357229190873E+323
2000	5328	333	10	3330	2.61605773705871E+281
4000	7990	799	5	5593	1.711330818417309E+660
4000	10656	666	10	6660	6.843758083624728 E+547
8000	21328	1333	10	13330	3.278591729502E+1114
16000	42656	2666	10	26660	1.074916372876E+2241

6. Conclusiones

Aunque el problema #SAT es en general #P-completo, existen diferentes instancias que se pueden resolver de forma eficiente. Si el grafo restringido de la entrada tiene ciclos, se muestra un procedimiento que permite calcular el número de modelos con complejidad del orden $O(2^{max\{|E(\overline{T_i})|\}} \cdot poly(|E(T)|))$, donde *poly* es una función polinomial y los $T_i, \overline{T_i}$ son los árboles y co-árboles respectivamente de la descomposición del grafo original . Este es un método que muestra que ciertas instancias se pueden resolver de manera eficiente.

Finalmente, se ha mostrado que la implementación del algoritmo produce mejores resultados que al menos otra implementación que se reporta en la literatura y que para cierta clase de grafos que tiene un número considerable de ciclos, el procedimiento regresa el número de modelos de forma eficiente.

Referencias

1. Bubley R.: Randomized Algorithms: Approximation, Generation, and Counting. Distinguished dissertations Springer (2001)
2. Bubley R., Dyer M.: Graph Orientations with No Sink and an Approximation for a Hard Case of #SAT. In: Proc. of the Eight Annual ACM-SIAM Symp. on Discrete Algorithms, pp. 248–257 (1997)
3. Dahllöf V., Jonsson P., Wahlström M.: Counting Satisfying Assignments in 2-SAT and 3-SAT. LNCS 2387, pp. 535–543 (2002)
4. De Ita G., Marcial-Romero J.R., Mayao Y.: An Enumerative Algorithm for #2SAT. Electronic Notes in Discrete Mathematics, vol. 46, pp 81–88 (2015)
5. D. Gonçalves.: Covering planar graphs with forests, one having a bounded maximum degree. Electronic Notes in Discrete Mathematics, 31, pp. 161–165 (2008)
6. Garey M., Johnson D.: Computers and Intractability a Guide to the Theory of NP-Completeness. W.H. Freeman and Co. (1979)
7. J. A. Bondy and U.S.R. Murty.: Graph Theory. Springer Verlag, Graduate Texts in Mathematics (2010)
8. Levit V.E., Mandrescu E., The independence polynomial of a graph - a survey, To appear, Holon Academic Inst. of Technology (2005)
9. M. Garey and D. Johnson: Computers and Intractability: a Guide to the Theory of NP-Completeness. W.H. Freeman and Co. (1997)
10. R. Tarjan: Depth-first search and linear graph algorithms. SIAM Journal on Computing, 1, pp. 146–160 (1972)
11. Telikepalli Kavitha, Christian Liebchen, Kurt Mehlhorn, Dimitrios Michail, Romeo Rizzi, Torsten Ueckerdt, and Katharina A. Zweig: Cycle bases in graphs characterization, algorithms, complexity, and applications. Computer Science Review, 3: pp. 199–243 (2009)
12. Vadhan Salil P.: The Complexity of Counting in Sparse, Regular, and Planar Graphs. SIAM Journal on Computing, Vol. 31, No.2, pp. 398–427 (2001)

Comparativa de algoritmos bioinspirados aplicados al problema de calendarización de horarios

Lucero de Montserrat Ortiz Aguilar[1], Juan Martín Carpio Valadez[1], Héctor José Puga Soberanes[1], Claudia Leticia Díaz González[1], Carlos Lino Ramírez[1] y Jorge AlbertoSoria-Alcaraz[2]

[1] Instituto Tecnológico de León, León, Guanajuato, México

[2] Universidad de Guanajuato, Guanajuato, Guanajuato, México

{ldm_oa,jmcarpio61}@hotmail.com, pugahector@yahoo.com,{posgrado,carloslino}@itleon.edu.mx, jorge.soria@ugto.mx

Resumen. El problema de calendarización de eventos está presente en diversas organizaciones como lo son escuelas, hospitales, centros de transporte, etc. La calendarización de actividades en una universidad tiene como propósito el garantizar que todos los estudiantes tomen sus asignaturas requeridas apegándose a los recursos que están disponibles. El conjunto de restricciones que debe contemplarse en el diseño de horarios involucra a los alumnos, maestros e infraestructura. En este trabajo se muestra que mediante la aplicación de algoritmos Genéticos, Memético y Sistema Inmune se generan soluciones aceptables, para el problema de calendarización de tareas, Los algoritmos son aplicados a instancias reales del Instituto Tecnológico de León y sus resultados son comparables con los de un experto humano.

Palabras clave: algoritmo genético, algoritmo memético, sistema inmune, faculty timetabling, course timetabling.

1. Introducción

La calendarización de tareas dentro de las organizaciones es uno de los problemas más comunes y difíciles de tratar, debido a que se busca asignar diversas actividades y recursos en un espacio de tiempo [1]. En las Universidades, se busca generar un diseño de horarios que cumpla con las restricciones de los alumnos, profesores, plan de estudios e inmuebles de la institución. Además de que el problema de calendarización depende del tipo de escuela, universidad y sistema de educación, por lo cual no existe un diseño de horarios que pueda ser aplicado de manera generalizada a todos los casos [2].

En general el problema de calendarización de horarios se define a partir de un conjunto de eventos (clases, cursos, exámenes), los cuales deben ser asignados en un conjunto de espacios de tiempo y que están sujetos a un conjunto de restricciones [3].

La calendarización Universitaria de acuerdo con Adriaen et.al [4] se clasifica en 5 grupos:

1. **Faculty Timetabling (FTT)**. Es la asignación de maestros a materias.
2. **Class-Teacher Timetabling (CTTT)**. Es asignación de materias con el menor conflicto temporal posible entre grupos de alumnos.
3. **Course Timetabling (CTT)**. Es la asignación de materias con el menor conflicto temporal posible entre alumnos individuales.
4. **Examination Timetabling (ETT)**. Es la asignación de exámenes a los alumnos, de tal forma que el alumno no aplique dos pruebas al mismo tiempo.
5. **Classroom assignment (CATT)**. Después de asignar las clases a los maestros, se asignan *class-teacher* a los salones.

En este trabajo se enfoca a generar soluciones aceptables al problema de calendarización de horarios, mediante el uso de algoritmos Metaheurísticos. Existen una diversa cantidad de enfoques que han sido utilizados para resolver el problema de calendarización de horarios como el coloreo de grafos [5], programación de Satisfacción de Restricciones(CSP) Métodos Basados[7], IP/LP (programación entera/programación lineal) [6], Algoritmos Genéticos [2, 8, 9], Algoritmos Meméticos [10, 11], Búsqueda Tabú [12, 13], Recosido Simulado [14], Búsqueda Local [15], Mejor-Peor sistema de Hormigas (BWAS) y las optimización por colonia de hormigas[16] y enfoque hiperheurístico [17].

Existen una gran cantidad diferentes problemas al menos de clase NP, los cuales pueden ser resueltos con diversos algoritmos Metaheurísticos, pero como nos indica el Teorema de No-Free Luch [18] no existe una Metaheurística que supere en rendimiento a todas las demás para todos los problemas conocidos de la clase NP. Debido a lo anterior, en este trabajo se hace una comparativa entre algoritmo Genético, Memético y Sistema Inmune, aplicando pruebas estadísticas no paramétricas.

Las instancias utilizadas pertenecen a datos reales del Instituto Tecnológico de León (ITL), donde la calendarización de horarios se elabora por un experto y se busca mediante los algoritmos Metaheurísticos generar soluciones que compitan con las propuestas el experto humano.

2. El problema de calendarización de tareas

El calendarización de tareas puede ser definido como el proceso de asignar clases a recursos como lo son de tiempo, espacio (salones) y maestros (personal), mientras se satisfaga un conjunto restricciones [19].

Existen dos tipos de restricción [20]:

- **Duras.** Es la restricción que absolutamente no puede ser quebrantada. Algunos ejemplos de restricciones duras son [12]: disponibilidad del salón, conflictos entre alumnos, disponibilidad de recursos (maestros, aulas).
- **Blandas.** El conjunto de restricciones que se prefieren satisfacer pero no se supone satisfacerlas todas. Algunos ejemplos de restricciones blandas son [12]: capacidad del Salón, mínimo de días ocupados, Etc.

Una definición del problema de calendarización de tareas está dada por Lewis Rhydian [21]: Dada una 4-tupla (e, t, p, S) la cual es la representación de una posible solución, en donde $E = \{e_1, e_2, \dots, e_n\}$ es un conjunto de eventos (clases o materias), $T = \{t_1, t_2, \dots, t_s\}$ es un conjunto de periodos de tiempo, $P = \{p_1, p_2, \dots, p_m\}$ es un conjunto de lugares (salones), $A = \{a_1, a_2, \dots, a_o\}$ un conjunto de usuarios (estudiantes registrados en cursos) y $S \subseteq A$ un subconjunto de estudiantes; la 4-tupla tiene asociada una función de costo $f(t)$. El problema es buscar las 4-tuplas (e, t, p, S) o soluciones tales que minimicen la función de costo $f(t)$ asociada.

2.1. Metodología API-Carpio

La metodología API-Carpio [22] describe el proceso de calendarización de horarios educativos como:

$$f(x) = FA(x) + FP(x) + FI(x) \tag{1}$$

Dónde:

$FA(x) =$ Número de estudiantes en conflicto dentro del horario x, (CTT).

$FP(x) =$ Número de profesores en conflicto dentro del horario x, (FTT).

$FI(x) =$ Número de aulas y laboratorios en conflicto dentro del horario x,(CATT).

En este trabajo se restringe a tomar solamente hasta $FA(x)$ la cual está definida como:

$$FA = \sum_{j=1}^{k} FA_{V_j} \tag{2}$$

dónde:

$$FA_{V_j} = \sum_{s=1}^{\left(M_{V_j}\right)-1} \sum_{l=1}^{\left(M_{V_i}\right)-s} (A_{j,s} \wedge A_{j,s+l}) \tag{3}$$

Teniendo:

$FA_{V_j} =$ Número de estudiantes en conflicto dentro del vector V_j.

$V_j =$ Es un vector de tiempo que contiene diferentes materias.

$A_{j,s} \wedge A_{j,s+l} =$ Número de alumnos que demandan la inscripción simultánea de las materias $M_{j,s} \wedge M_{j,s+1}$.

2.2. Metodología del diseño

La metodología del diseño propuesta por Soria et al. [23] permite que diferentes políticas de la calendarización de tareas y listas de restricciones sean modelados mediante la conversión de todas las restricciones de tiempo y espacio en un simple tipo de restricción: conflictos de estudiantes. En esta se proponen estructuras como lo son la matriz MMA, lista LPH, lista LPA y lista LPS. Las primeras tres representan las restricciones duras y la última representa las restricciones blandas. En este trabajo se

utilizaran dos de las cuatro estructuras las cuales son: matriz MMA y lista LPH. En [23] nos definen que sus estructuras son las siguientes:

Matriz MMA: Contiene el número de conflictos (entre alumnos) posibles si dos lecciones son asignadas en el mismo espacio de tiempo. La figura 1a muestra un ejemplo de matriz MMA.

Lista LPH: Esta lista nos da información acerca de cada lección (clase, evento o materia) en que posible espacio de tiempo puede ser asignada. La figura 1b muestra un ejemplo de LPH.

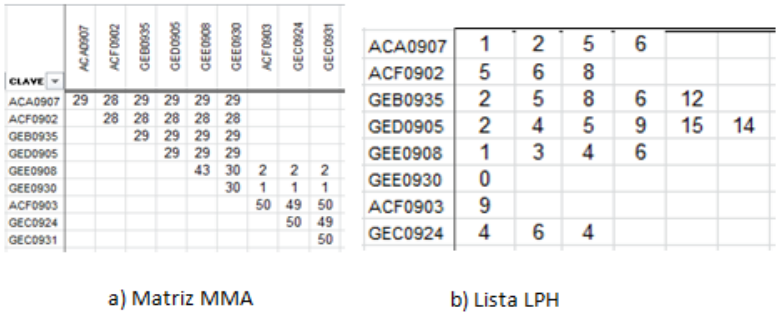


Fig. 1. Ejemplo de MMA y LPH

3. Algoritmo genético

Los algoritmos Genéticos fueron desarrollados por J. Hollan en los 70s, con el fin de entender el proceso adaptativo del sistema natural, para luego aplicarlo en la optimización y Machine Learning en los 1980s [25]. Los algoritmos Genéticos son métodos adaptativos, generalmente usados en problemas de búsqueda y optimización de parámetros, basados en la reproducción sexual y en el principio de supervivencia del más apto. En [26] define el algoritmo 1 que corresponde a un Genético Simple.

La representación de las soluciones candidatas se muestra en la figura 2, donde en cada posición representan un evento o materia y la columna representa el espacio de tiempo asignado de la LPH.

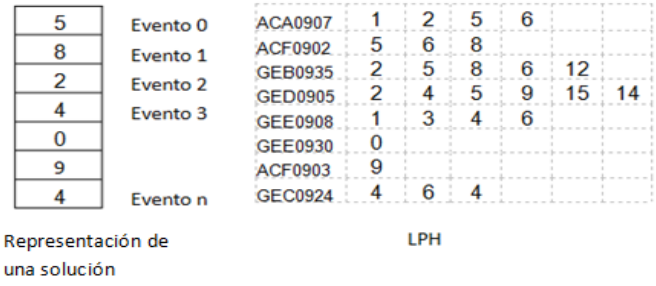


Fig. 2. Representación de las soluciones candidatas

Algoritmo 1 Genético Simple

Requiere: Función Objetivo $F(x), x = (x_1, \dots, x_n)^T$

```
Generar una población Inicial
Mientras t < Max número de generaciones hacer
    Generar una nueva solución por la cruza y la muta
        Cruza
        Muta
    Seleccionar la mejor solución para la siguiente generación (elitismo)
Fin del lazo
Regresar La mejor solución de la población
```

En [24] menciona que el Genético es una de las estrategias más usadas y en este trabajo se empleó la selección por Ruleta pesada. Esta consiste en asignar a cada individuo una parte proporcional de probabilidad en relación a la función de aptitud. Teniendo f_i el fitness del individuo p_i en la población P y La fórmula de probabilidad se puede consultar en [27]. La cruza es el proceso probabilístico que cambia la información entre dos cromosomas (padres) para generar dos cromosomas hijos [30] y la empleada en este trabajo es la cruza a un punto descrita en [24], donde el sitio de cruza k es seleccionado aleatoriamente y los dos hijos son creados intercambiando los segmentos de los padres. Y para el operador de muta tenemos que será a un punto donde seleccionaremos aleatoriamente una posición k y cambiaremos el valor de esa posición por otro que se encuentre en la LPH [24].

4. Algoritmos meméticos

En 1976 Dawkins diseñó el concepto de meme, el cual a diferencia del gen puede ser modificado por su portador. Supuso que existe un progreso como un gen del algoritmo genético que es transferido a la próxima generación, es decir, las características obtenidas se transfieren de una generación anterior a una siguiente, junto con los cambios de población. Moscato en [32] donde describe lo que es el algoritmo Memético.

Algoritmo 2 Memético

Requiere: Función Objetivo $F(x), x = (x_1, \dots, x_n)^T$

```
Generar una población Inicial
Mientras t < Max número de generaciones hacer
    Generar una nueva solución por la cruza y la muta
    Hacer Cruza
    Hacer búsqueda Local
    Hacer Muta
    Seleccionar la mejor solución para la siguiente generación (elitismo)
Fin del lazo
Regresar La mejor solución de la población
```

Un meme es una unidad de datos que puede recrearse a sí misma. Estas unidades se transmiten entre las personas y cualquier otra, que se pueden ajustar con ella, y es capaz de salvar la unidad de datos; mientras que un gen se mantiene sin cambios durante la transmisión [33]. Los componentes de un Algoritmo Memético son [24]: Algoritmo Genético y Búsqueda Local.

La búsqueda local es una modificación que se puede llegar a hacer toda la población de individuos con la que trabaja el algoritmo. En este se realiza una copia de cada individuo, donde esta copia es alterada de alguna forma; si la copia de un individuo en específico es mejor que la original, la copia remplaza al individuo original. En el algoritmo 2 se describen los pasos de un Memético.

5. Sistema inmune

Estos se basan en imitar el comportamiento del sistema inmunológico humano, el cual se encarga de proteger al cuerpo de los patógenos externos e internos y su tarea principal es reconocer las células en el cuerpo clasificarlas como propias y no propias. Los algoritmos de Sistema Inmune artificial han sido aplicados con éxito en diversos problemas de optimización [29].

Algoritmo 3 Sistema Inmune Artificial

Requiere: tamaño máximo de población, número de clones

1: Generar una población Inicial
2: **Mientras** $t < Max$ número de generaciones **hacer**
3: Selección.
4: Clonar.
5: Hipermuta
6: **Si tamaño de la población > max_tamaño entonces**
7: Poda (autorregulación)
8: **Fin**
9: **Fin del lazo**
10: Regresar La mejor solución de la población

Básicamente el proceso del algoritmo de Sistema Inmune Artificial consiste en generar aleatoriamente una población de soluciones candidatas; después seleccionamos un porcentaje de los mejores individuos, los cuales son clonados, luego a estos individuos se les aplica una hipermuta y finalmente continuamos hasta llegar a nuestra solución objetivo, pero para evitar que nuestra población crezca sin medida se pone una poda la cual nos permitirá regresar al tamaño inicial de la población [34].

El algoritmo 3 propuesto por [31], corresponde al Sistema Inmune Artificial.

6. Experimentación y resultados

Las instancias usadas para las pruebas con las Metaheurísticas antes mencionadas pertenecen a ITL, estas corresponden a dos planes educativos distintos, pertenecientes al año del 2009 y 2014, cuentan con aproximadamente de 46 a 58 clases (eventos) y una cantidad de 9 a 11 espacios de tiempo respectivamente. La configuración utilizada en los algoritmos Genético, Memético y SI se muestran en la tabla 1, donde tenemos que las llamadas a función fueron 300,000, la población inicial para cada uno fue la misma. El criterio de paro de los algoritmos fue el de llamadas a función.

Tabla 1. Datos para la configuración Inicial del AG, AM y SI

Parámetro	Genético	Memético	Sistema Inmune
Población	20	20	20
Llamadas a función	300,000	300,000	300,000
Elitismo	0.1	0.1	NA
Cruza	0.9	0.9	NA
Muta	0.15	0.15	NA
Poda	NA	NA	100

La tabla 2 muestra los resultados obtenidos (conflictos) de acuerdo a la función objetivo mostrada en 1, donde se puede observar la media, la mediana y la deviación estándar de las ocho instancias evaluadas con el AG, AM, SI, y los conflictos del experto.

Tabla 2. Resultados las diferentes Metaheurísticas y experto aplicados a las instancias.

Instancia	Media			Mediana			Desviación estándar			Experto
-	AG	AM	SI	AG	AM	SI	AG	AM	SI	
1	281.3	282.6	295.7	279	282	304	16.24	16.10	18.7	577
2	161.6	162.5	194.2	162	160	209	14.73	14.00	13.54	308
3	247.6	265.0	293.9	250	262	325	19.71	19.84	20.58	444
4	160.6	150.9	177.3	147	152	189	11.03	10.61	9.07	249
5	129.1	135.7	164.7	131	137	178	10.34	8.02	9.59	300
6	73.9	77.9	89.0	74	77	99	6.81	7.19	7.73	157
7	75.0	79.5	180.9	75	81	190	7.69	9.47	9.92	138
8	91.1	97.0	104.7	89	95	109	12.8	16.99	11.55	258

Se puede observar en la tabla 2 que las desviaciones estándar de algunas instancias no hay gran diferencia entre algoritmos, como son el caso de la instancia 2, 3 y 6, ya que lo que se busca es encontrar algoritmos que tengan soluciones aceptables con una baja desviación estándar, lo que quiere decir que los datos están muy cercanos a la media, lo cual es indicativo de reproducibilidad de resultados por parte de los algoritmos Metaheurísticos utilizados. Para aplicar las pruebas estadísticas no paramétricas se aplicó el test de Friedman [28], de donde se tomarán las medianas de los algoritmos genético, Memético y SI y después aplicar el test y conocer cuál fue el algo-

ritmo con el mejor rendimiento se aplicó la prueba de suma de rangos con signo de Wilcoxon, para contrastar los resultados del algoritmo con mejor rendimiento y los resultados del experto humano.

En la tabla 3 se muestran los rangos y resultados del test ómnibus de Friedman donde $h_0 =$ No existen diferencias en el desempeño de los algoritmos y $h_a =$ Existen diferencias entre Algoritmos, de donde el valor *P* es menor en los tres casos, que el valor de $\alpha = 0.05$, por lo que no tenemos suficiente evidencia para aceptar h_0. Tomando como algoritmo de control al Genético, debido a que es el que tiene el menor rango hacemos las pruebas post-hoc, con un valor de $\alpha = 0.05$.

Tabla 3. Rangos, estadísticos y valor p para AG, AM y SI.

Algoritmos	**Friedman**	**Friedman Alineado**	**Quade**
Genético	**1.125**	**59**	**1.083**
Memético	1.875	77.1	1.916
SI	3.000	164	3.000
Estadístico	14.25	12.19	19.26
Valor *P*	0.0008	0.0079	6.565E-05

En la tabla 4 se muestran los valores z y los valores p con ajuste Bonferroni [28]. Como podemos observar en las pruebas a pares para el caso del AG vs el AM en los tres test el valor p es menor que el $\propto$ por lo cual no tenemos suficiente evidencia para rechazar h_0, mientras que para el caso de AG vs SI nuevos valores p son menores que el α, por lo cual nos dice que existe diferencia en el comportamiento de los algoritmos

Tabla 4. Pruebas Post-Hoc. Tomando como algoritmo de control al Algoritmo Genético.

Algoritmo	**Friedman**		**Friedman Alineado**		**Quade**	
-	z	Bonferroni	z	Bonferroni	z	Bonferroni
AG vs AM	1.50	0.2672	1.06	0.5723	1.48	0.2749
AG vs SI	3.75	0.0003	6.18	1.227E-09	3.41	0.0012

Como no existe diferencia en el comportamiento del Genético y el Memético utilizaremos al que tuvo el rango más pequeño en las pruebas y en este caso fue el Genético (ver tabla 3) para compararlo en la prueba de suma de rangos de con signo Wilcoxon y ver si los resultados del genético con los resultados del experto provienen de poblaciones con medianas iguales. Los resultados se muestran en la tabla 5.

Al Aplicar la prueba de suma de rangos con signo de Wilcoxon (ver tabla 5), bajo las hipótesis: $h_0 =$ No existen diferencias en las medianas vs. $h_a =$ Existen diferencias entre las medianas, donde se tomó un valor de significancia $\alpha = 0.05$ y 8 grados de libertad, encontramos que existe evidencia suficiente para rechazar h_0 . Por lo tanto, tenemos que efectivamente los datos provienen de poblaciones con medianas diferentes, es decir, el algoritmo Genético mejora los resultados obtenidos por el experto humano.

Tabla 5. Prueba de suma de rangos con signo de Wilcoxon entre Genético y resultados del Experto humano

Instancia	Genético	Experto	Rankeo	Signo
1	279	577	8	-
2	162	308	4	-
3	250	444	7	-
4	147	249	3	-
5	131	300	5.5	-
6	74	157	2	-
7	75	138	1	-
8	89	258	5.5	-
W^- =	36	W =	0	
W^+=	0	W_0=	4	

7. Conclusiones

El uso de algoritmos de optimización en la calendarización de horarios nos permiten minimizar el número de conflictos entre los diferentes recursos de la institución como lo son los docentes, el inmueble; permitiendo que los estudiantes puedan tener un calendario que satisfaga sus necesidades. En este trabajo se muestra primero una comparación entre diferentes Metaheurísticas, que nos permiten encontrar soluciones que resuelven el problema de la calendarización de tareas; posteriormente se determina cual es el que obtuvo mejor desempeño y se hace una comparación con los resultados del experto mediante una prueba estadística no paramétrica.

Las instancias utilizadas pertenecen a insumos reales del ITL, de donde se obtuvieron el número de conflictos de la solución propuesta por el experto encargado del diseño de los horarios; permitiendo hacer una comparación entre los resultados generados por las Metaheurísticas vs. los del experto humano. Los resultados muestran que el Algoritmo Genético fue él tuvo mejor desempeño, sin embargo, la prueba estadística de Friedman nos señala que no existe suficiente evidencia discernible entre el comportamiento de este y el Memético, pero debido al valor de los rangos fue el que se tomó para comparación con los resultados del experto.

Por último al hacer la prueba de rangos con signo de Wilcoxon entre el Genético y los resultados del experto, nos indica que existe diferencia de posición entre las distribuciones de resultados del Genético y el experto humano, por lo que, basados en los resultados podemos afirmar que el algoritmo Genético mejoró los resultados para este conjunto de instancias.

Como trabajo futuro se propone el integrar otros algoritmos Metaheurísticos empleando otro tipo de selección, cruza y muta para el caso del genético y Memético e implementar otras versiones de Sistema Inmune que nos generen mejores soluciones y aplicar la prueba ómnibus incorporando los resultados del experto humano junto a los Metaheurísticos. Además de tomar un enfoque que involucre las variables restan-

tes de la Metodología API-Carpio, como lo es la suma de los conflictos del maestro, junto con los de las aulas.

Agradecimientos. Agradecimientos al Consejo Nacional de Ciencia y Tecnología (CONACYT) México por el apoyo brindado en esta investigación con el número de beca 308646 y a la División de Estudios de Posgrado del Instituto Tecnológico de León.

Referencias

1. Jorge A, S., Martin, C. J., & Hugo, T.: Academic timetabling design using hyper-heuristics. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 43–56 doi:10.1007/978-3-642-15534-5_3 (2011; 2010)
2. Asratian, A. S., de Werra, D.: A generalized class–teacher model for some timetabling problems, University of Technology, Department of Engineering Sciences and Mathematics, Mathematical Science, & Mathematics, European Journal of Operational Research, pp. 531–542 (2002) doi:10.1016/S0377-2217(01)00342-3.
3. Soria-Alcaraz Jorge, A., Martín, C., Héctor, P., & Sotelo-Figueroa, M. A.: Comparison of metaheuristic algorithms with a methodology of design for the evaluation of hard constraints over the course timetabling problem. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 289–302, doi:10.1007/978-3-642-33021-6_23 (2013)
4. Adriaen, M., Causmaecker, P., Demeester, P.: Tackling the university course timetabling problem with an aggregation approach. In: Burke, K., Rudova, H. (eds.) Proceedings PATAT 2006, pp. 330–335 (2006)
5. De Werra, D.: An introduction to timetabling. European Journal of Operational Research, vol. 19, no. 2, pp. 151–162 (1985).
6. Obit, J. H., Ouelhadj, D., Landa-Silva, D., Vun, T. K., & Alfred, R.: Designing a multi-agent approach system for distributed course timetabling, pp. 103–108, doi:10.1109/HIS.2011.6122088 (2011)
7. Lewis, M. R. R.: Metaheuristics for university course timetabling. Ph.D. Thesis, Napier University (2006)
8. Deng, X., Zhang, Y., Kang, B., Wu, J., Sun, X., & Deng, Y.: An application of genetic algorithm for university course timetabling problem, pp. 2119–2122, doi:10.1109/CCDC.2011.5968555 (2011)
9. Mahiba, A.A. & Durai, C.A.D.: Genetic algorithm with search bank strategies for university course timetabling problem. Procedia Engineering, vol. 38, pp. 253–263 (2012)
10. Soria-Alcaraz, J. A.; Carpio, J. M.; Puga, Hé.; Melin, P.; Terashima-Marn, H.; Reyes, L. C. & Sotelo-Figueroa, M. A. Castillo, O.; Melin, P.; Pedrycz, W. & Kacprzyk, J.: Generic Memetic Algorithm for Course Timetabling. In: ITC2007 Recent Advances on Hybrid Approaches for Designing Intelligent Systems, Springer, vol. 547, pp. 481–492 (2014)
11. Nguyen, K., Lu, T., Le, T., & Tran, N.: Memetic algorithm for a university course timetabling problem. pp. 67–71, doi:10.1007/978-3-642-25899-2_10 (2011)
12. Aladag, C., & Hocaoglu, G.: A tabu search algorithm to solve a course timetabling problem. Hacettepe journal of mathematics and statistics, pp. 53–64 (2007)
13. Moscato, P.: On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts: Towards Memetic Algorithms. Caltech Concurrent Computation Program (report 826) (1989)

14. Frausto-Solís, J., Alonso-Pecina, F., & Mora-Vargas, J.: An efficient simulated annealing algorithm for feasible solutions of course timetabling. Springer, pp. 675–685 (2008)
15. Joudaki, M., Imani, M., & Mazhari, N.: Using improved Memetic algorithm and local search to solve University Course Timetabling Problem (UCTTP). Doroud, Iran: Islamic Azad University (2010)
16. Thepphakorn, T., Pongcharoen, P., & Hicks, C.: An ant colony based timetabling tool. International Journal of Production Economics, vol. 149, pp. 131–144 doi:10.1016/j.ijpe.2013.04.026 (2014)
17. Soria-Alcaraz, J., Ochoa, G., Swan, J., Carpio, M., Puga, H., & Burke, E.: Effective learning hyper-heuristics for the course timetabling problem. European Journal of Operational Research, pp. 77–86, doi:10.1016/j.ejor.2014.03.046 (2014)
18. Wolpert, H., Macready, G.: No free lunch Theorems for Search. Technical report, The Santa Fe Institute, vol. 1 (1996)
19. Lai, L. F., Wu, C., Hsueh, N., Huang, L., & Hwang, S.: An artificial intelligence approach to course timetabling. International Journal on Artificial Intelligence Tools, pp. 223–240, doi:10.1007/s10479-011-0997-x (2008)
20. McCollum, B., McMullan, P., Parkes, A. J., Burke, E. K., & Qu, R.: A new model for automated examination timetabling. Annals of Operations Research, pp. 291–315 (2012; 2011)
21. Conant-Pablos, S.E., et al.: Pipelining Memetic algorithms, constraint satisfaction, and local search for course timetabling. In: MICAI Mexican International Conference on Artificial Intelligence, vol. 1, pp 408–419 (2009)
22. Carpio-Valadez, J.M.: Integral Model for optimal assignation of academic tasks. In: Encuentro de investigacion en ingenieria electrica, ENVIE, Zacatecas, pp. 78–83 (2006)
23. Soria-Alcaraz, J. A., Martin, C., Héctor, P., Hugo, T., Laura, C. R., & Sotelo-Figueroa, M. A.: Methodology of design: A novel generic approach applied to the course timetabling problem. pp. 287–319, doi:10.1007/978-3-642-35323-9-12 (2013)
24. Talbi, E. Metaheuristics: From design to implementation. US: Wiley (2009)
25. Goldberg, D. E.: Genetic algorithms in search, optimization, and machine learning. Reading, Mass: Addison-Wesley Pub. Co. (1989)
26. Yang, X.-S. Nature-inspired metaheuristic algorithms. Luniver press (2010)
27. Abdoun O., Abouchabaka J.: A Comparative Study of Adaptive Crossover Operators for Genetic Algorithms to Resolve the Traveling Salesman Problem. International Journal of Computer Applications (2011)
28. Derrac, J., García, S.: A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence. Swarm and Evolutionary Computation (2011)
29. Azuaje, F.: Review of "Artificial immune systems: A new computational intelligence approach." Journal Neural Networks, vol. 16, no.8, Elsevier, pp. 1229–1229 (2003)
30. Maulik, U., Bandyopadhyay, S.: Genetic algorithm-based clustering technique. Pattern Recognition, Vol. 33, pp. 1455–1465 (2000)
31. Herrera Lozada, J. C., Calvo, H., & Taud, H.: A micro artificial immune system. (2011)
32. Lü, Z., & Hao, J.: Adaptive tabu search for course timetabling. European Journal of Operational Research, pp. 235–244, doi:10.1016/j.ejor.2008.12.007 (2010)
33. Araujo, L., Cervigón, C.: Algoritmos Evolutivos, Un enfoque práctico. Alfaomega (2009)
34. Villalobos Arias, M., Coello Coello, C. A., & Hernández Lerma, O.: Convergence analysis of a multiobjective artificial immune system algorithm. (2004)

Pronóstico difuso del examen general de egreso de licenciatura para ingeniería en computación de la Universidad Autónoma del Estado de México

Sandra Silvia Roblero Aguilar, Héctor Rafael Orozco Aguirre

Universidad Autónoma del Estado de México,
Centro Universitario UAEM Valle de México,
Atizapán de Zaragoza, Estado de México, México

ssrauaemex@hotmail.com, hrorozcoa@uaemex.mx

Resumen. Este artículo tiene como finalidad dar a conocer un mecanismo computacional, que permite generar un pronóstico utilizando lógica difusa, para los pasantes de la carrera de Ingeniería en Computación de la Universidad Autónoma del Estado de México (UAEM), a través de la opción de titulación por el Examen General de Egreso de Licenciatura (EGEL) aplicado por el Centro Nacional de Evaluación para la Educación Superior (CENEVAL).

Palabras clave: lógica difusa, pronóstico, examen general de egreso de licenciatura (EGEL), eficiencia terminal, eficiencia de titulación, UAEM.

1. Introducción

Actualmente, en México la Eficiencia Terminal (ET) es el principal indicador empleado por el Sistema Educativo Nacional con el objetivo de evaluar la labor de formación profesional de cualquier Institución Educativa (IE). La Dirección General de Planeación y Estadística Educativa de la Secretaría de Educación Pública (SEP) la define como: "la relación porcentual entre los egresados de un nivel educativo dado y el número de estudiantes que ingresaron al primer grado de este nivel educativo n años antes" [1], que matemáticamente se expresa así:

$$ET(n) = (Egresos(n)/Nuevo\ Ingreso(n)) * 100 \quad (1)$$

Con la finalidad de que exista una evaluación real de las Instituciones de Educación Superior (IES), es necesario tomar en cuenta aparte de la ET, el número de pasantes que logran graduarse, es decir, aquellos que impactan positivamente en el índice de la Eficiencia de Titulación (ETi), que está dada por la expresión:

$$ETi(n) = (Titulados(n) /Nuevo\ Ingreso(n)) * 100 \quad (2)$$

Dado lo anterior, para poder incrementar la ETi, las IES consideran como válidas a distintas modalidades en sus procesos de titulación, en estas se incluyen: aprovecha-

miento académico, créditos de estudios avanzados, el Examen General de Egreso de Licenciatura (EGEL), entre otras, como lo son los trabajos escritos.

Al hacer un análisis de los resultados presentados en el anuario estadístico de la Asociación Nacional de Universidades e Instituciones de Educación Superior (ANUIES), correspondiente al ciclo escolar 2012-2013, que es el último reportado hasta noviembre de 2014 [2], y de la Agenda Estadística de la Universidad Autónoma del Estado de México (UAEM) para el mismo ciclo escolar [3], se tiene la siguiente tabla comparativa:

Tabla 1. Comparativo de indicadores de la ET y la ETi del ciclo escolar 2012-2013

	Matricula	Nuevo Ingreso	Egresados	Eficiencia Terminal	Titulación	Eficiencia de Titulación
Nacional	**3,309.221**	**877,476**	**469,573**	**53.5%**	**343,613**	**39.1%**
UAEM	**45,161**	**12,235**	**5,674**	**46.4%**	**4,056**	**33.1%**
CU UAEM VM[1]	**3,053**	**864**	**396**	**45.8%**	**189**	**21.8%**
ICO[2]	**211**	**84**	**17**	**20.2%**	**9**	**10.7%**

Como se puede observar en la tabla anterior, en el CU UAEM VM, la carrera de ICO requiere una atención inmediata, para que se pueda establecer mecanismos que conlleven a incrementar la ETi. Adicionalmente, es importante mencionar que el Reglamento de Evaluación Profesional de la UAEM [4], considera como meta del plan rector la necesidad de mejorar la capacidad profesional de egresados para favorecer su inserción laboral y progreso profesional, fijándose como porcentajes lograr un índice de titulación por cohorte de 23.5%, de los cuales el 29% de los egresados que se titulan lo hagan a través del EGEL. De acuerdo a esto, si 864 alumnos son de nuevo ingreso en el CU UAEM VM, esto implica que para cumplir con la meta, entonces 203 egresados se deben titular y de estos 59 por EGEL. Para la carrera de ICO, si 84 son de nuevo ingreso, 20 se van a titular y 6 lo harán por el EGEL.

Si se desea impulsar al EGEL como una modalidad de titulación, se tendrán que considerar: los contenidos de los Programas Educativos (PE), las necesidades del campo laboral y que los alumnos adquieran las competencias profesionales a ser evaluadas. De manera tradicional, la preparación para el EGEL no contempla que las áreas y sub-áreas de evaluación hayan sido cubiertas en los contenidos de los PEs. Esto dice, que el anticipar un resultado es hasta cierto punto ciego, ya que no se sabe con exactitud qué Unidades de Aprendizaje (UDAs) corresponden a cada área, cuál es la más fortalecida y las que necesitan ser reforzadas. Para ello, se propone un mecanismo computacional basado en lógica difusa para generar un pronóstico del resultado del EGEL. Esto es para contar con un resultado aproximado que pudiera obtener cualquier sustentante de ICO. Dicho mecanismo, se describirá en la sección 4 de este artículo.

[1] Centro Universitario UAEM Valle de México.
[2] Ingeniería en Computación.

2. Aspectos generales del EGEL CENEVAL

2.1. Descripción

Como asociación civil sin fines de lucro, el Centro Nacional de Evaluación para la Educación Superior (CENEVAL), tiene la encomienda de diseñar y aplicar instrumentos de evaluación de conocimientos, habilidades y competencias, así como el análisis y la difusión de resultados. Uno de esos instrumentos es el EGEL, el cual consiste en una prueba a nivel nacional, que se especializa por carrera profesional. Esta prueba tiene como objetivo identificar en qué medida los egresados de una licenciatura en específico cuentan con los conocimientos y las habilidades esenciales para el inicio del ejercicio profesional en el campo laboral.

El encargado del diseño, revisión y actualización de cada variante del EGEL, es un Consejo Técnico (CT), que se conforma por representantes de IES públicas y privadas, así como por colegios o asociaciones de profesionales e instancias empleadoras del sector público o privado del país. Cada CT se encarga de validar a una o más licenciaturas en particular, con base en las necesidades de las instancias empleadoras del sector público o privado.

El EGEL en el caso de Ingeniería Computacional [5], es el examen aplicado a ICO de la UAEM, mismo que está conformado por 186 reactivos de opción múltiple con cuatro opciones de respuesta, de las cuales sólo una es la correcta. Este examen comprende las siguientes cinco áreas de conocimiento:

1. Selección de Sistemas Computacionales para Aplicaciones Específicas (SSCAE).
2. Nuevas Tecnologías para la Implementación de Sistemas de Cómputo (NTISC).
3. Desarrollo de Hardware y su Software Asociado para Aplicaciones Específicas (DHSAAE).
4. Adaptación de Hardware y/o Software para Aplicaciones Específicas (AHSAE).
5. Redes de Cómputo para Necesidades Específicas (RCNE).

Dicho examen, también puede ser empleado para evaluar otras carreras, tal es el caso de Ingeniería en Sistemas y Comunicaciones del CU UAEM VM.

2.2. Índice CENEVAL

Las calificaciones están expresadas en una escala especial llamada Índice CENEVAL (IC) [5], que va de los 700 puntos, calificación más baja; a los 1300 puntos, calificación más alta. Este índice se utiliza en las pruebas con referencia a criterio, con el objetivo de categorizar el resultado obtenido en alguno de los rangos de puntuación, mismos que serán explicados más adelante.

En lo que respecta a los resultados, se establece una media teórica de 1000 puntos igual a 50% de aciertos y una desviación estándar de 100 puntos igual a 16.67%. Lo

anterior debido a que las pruebas se diseñan para tener una distribución de las puntuaciones con esa media y desviación estándar.

2.3. Rangos de puntuación

En cada una de las áreas del examen se consideran tres niveles de desempeño: Aún No Satisfactorio (ANS), Satisfactorio (DS) y Sobresaliente (DSS), en los cuales se clasifica a los sustentantes en función del desempeño mostrado, de conformidad con los siguientes rangos de puntuación:

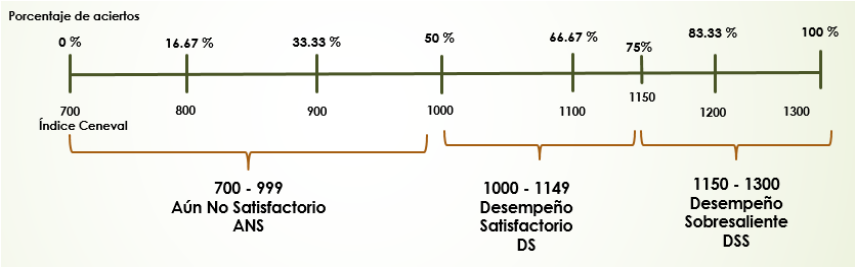


Fig. 1. Distribución de las puntuaciones del EGEL

En la escala de 0-100, el CT fija los cortes que corresponden a los puntajes 1000 y 1150 de IC. Esto hace que independientemente de en qué lugar el CT haya fijado los puntos de corte, el primero de estos siempre es 1000 y el segundo siempre es 1150. Con lo cual, sin importar el área a la cual se esté haciendo referencia, una calificación superior o igual a 1000 puntos indica un nivel de DS y una superior o igual a 1150, un nivel de DSS. Es decir, en cada una de las áreas del examen, la descripción de los niveles de desempeño permite conocer qué problemas y situaciones es capaz de resolver un sustentante cuando alcanza un DS, y cuáles, cuando alcanza un DSS.

Considerando el nivel de desempeño alcanzado por el sustentante en cada una de las áreas, se determina si éste se hace acreedor a algún Testimonio de Desempeño (TD), con base en los criterios establecidos por el CT, que a continuación se detallan:

- Testimonio de Desempeño Satisfactorio (TDS): al menos cuatro áreas con DS o DSS.
- Testimonio de Desempeño Sobresaliente (TDSS): de las cinco áreas, al menos dos con DSS y las restantes con DS.

Cabe destacar que para este caso, el EGEL está orientado a determinar si los sustentantes son capaces de utilizar lo que han estudiado y aprendido en su licenciatura en situaciones similares a las que se enfrentarán en el ejercicio profesional, así como el hecho de que su contenido se encuentra definido en forma precisa y se validó socialmente.

2.4. Resultados del EGEL para el ciclo escolar 2012-2013

El total de sustentantes que presentaron el EGEL en Ingeniería Computacional durante este ciclo escolar a nivel nacional fue de 3,350 sustentantes [5]. De ellos, 7.8% (260 sustentantes) obtuvo un TDSS; 37% (1,240 sustentantes) un TDS; y 55.2% (1,850 sustentantes) no obtuvo testimonio. De nueva cuenta, en el CU UAEM VM como referencia, para el mismo ciclo, de 238 sustentantes, el 4.62% (11 sustentantes) alcanzó un TDSS; 48.7% (116 sustentantes) un TDS; y 46.6% (111 sustentantes) se quedó sin testimonio. Para la carrera de ICO, los sustentantes que presentaron el EGEL fueron 18 sustentantes. De los cuales, el 11.1% (2 sustentantes) consiguió un TDSS; el 33.3% (6 sustentantes) un TDS; y el 55.56% (10 sustentantes) no obtuvo testimonio.

En la siguiente figura, se observa una gráfica del comparativo de resultados que en términos generales se puede decir, que aproximadamente el 55% de los sustentantes no alcanza el puntaje mínimo requerido por el EGEL para obtener un TDS y por ende, los sustentantes no pueden titularse por esta modalidad.

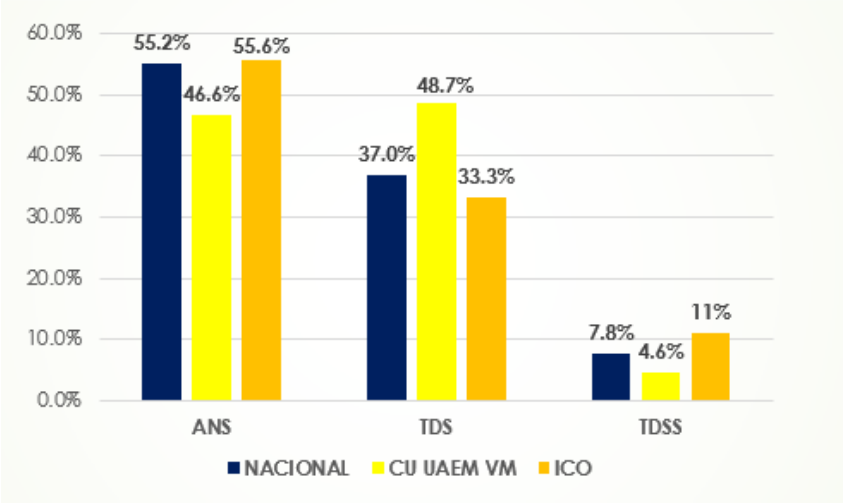


Fig. 2. Comparativo de los resultados del EGEL

3. Lógica difusa

3.1. Panorama general

El ser humano muestra dificultad para tomar decisiones cuando se tiene información imprecisa. La lógica difusa [6] fue creada para emular la lógica humana y tomar decisiones acertadas a pesar de la poca información disponible. Esta es una herramienta flexible que se basa en reglas lingüísticas dictadas por expertos y que tiene

como objetivo principal la formalización o mecanización de un sistema lógico para la evaluación y generación de decisiones.

La graduación y granulación (ver fig. 3) forman el núcleo de la lógica difusa, siendo sus principales características distinguibles [7]. Más específicamente, cuando se le emplea, todo dato o información es o se permite que sea graduado, es decir, es una cuestión de grado o, equivalentemente, difuso. Además, de forma similar todo es o se permite por igual que sea granular, un gránulo puede ser un grupo de atributos-valores unidos de manera indistinta, semejante, próxima o funcional. De una manera cualitativa, graduación y granulación juegan papeles fundamentales en la cognición humana.

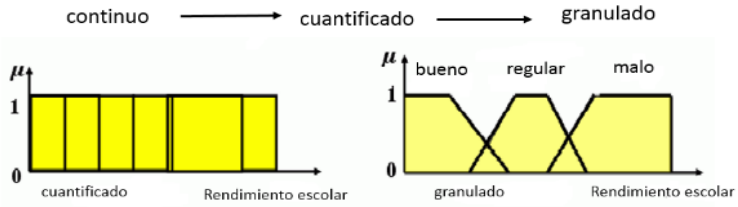


Fig. 3. Granulación y graduación de una variable lingüística

La lógica difusa es [8] un conjunto de principios matemáticos basados en grados de membresía o pertenencia, cuya función es modelar información. Este modelado se hace con base en reglas lingüísticas que aproximan una función mediante la relación de entradas y salidas del sistema. Esta lógica presenta rangos de pertenencia dentro de un intervalo entre 0 y 1, a diferencia de la lógica convencional, en la que el rango se limita a dos valores: el cero o el uno. La lógica difusa consta de tres etapas para obtener el resultado deseado (ver fig. 4). Estas se explican a continuación:

- Etapa 1: se basa en un proceso donde las variables tienen un grado de incertidumbre metalingüístico. Es decir, el rango de valores de cada variable puede clasificarse por conjuntos difusos, originando el universo del discurso. Con ello, los valores pasan a un proceso de fusificación que los categoriza en un rango de pertenencia entre 0 y 1 que pertenece a un conjunto difuso. Los conjuntos difusos son caracterizados mediante funciones de pertenencia, las cuales están sintonizadas al punto de operación adecuado para el funcionamiento del sistema, es decir, las reglas de inferencia que serán empleadas.
- Etapa 2: se proponen reglas lingüísticas conocidas como de inferencia. Con esto, el grado de pertenencia de cada una de las variables se evalúa en un subconjunto de estas reglas. Cada subconjunto se usa para determinar una consecuencia, es decir, asignar un grado de pertenencia a un conjunto difuso que caracteriza o da las salidas para las variables de entrada.
- Etapa 3: consiste en determinar los valores óptimos de salida, mediante un mecanismo conocido como defusificación, el cual consiste en pasar el grado de perte-

nencia, proveniente de la consecuencia de la regla de inferencia activada, a un valor nítido o real, es decir, con el fin de obtener un valor cuantificable.

Fig. 4. Esquema general de un mecanismo de inferencia difuso

3.2. Funciones de pertenencia

Las funciones de pertenencia son una forma de representar gráficamente un conjunto difuso sobre un universo. De esta manera, la función de pertenencia de un conjunto indica el grado en que cada elemento de un universo dado, pertenece a dicho conjunto. Es decir, la función de pertenencia de un conjunto A sobre un universo X será de la forma: $\mu A:X \rightarrow [0,1]$, donde $\mu A\ (x) = r$, si r es el grado en que X pertenece al conjunto A.

Si el conjunto es nítido, su función de pertenencia (función característica) tomará los valores en {0,1}, mientras que si es difuso, los tomará en el intervalo [0,1]. Si $\mu A(x) = 0$ el elemento no pertenece al conjunto, si $\mu A(x) = 1$ el elemento sí pertenece totalmente al conjunto. La función característica del conjunto de los elementos que verifican un predicado clásico está perfectamente determinada. No ocurre lo mismo cuando se intenta obtener la función de pertenencia de un conjunto formado por los elementos que verifican un predicado difuso. Dicha función dependerá del contexto (o universo) en el que se trabaje, del experto, del usuario, de la aplicación a construir, etc.

A la hora de determinar una función de pertenencia, normalmente se eligen funciones sencillas, para que los cálculos no sean complicados. En particular, en aplicaciones en distintos entornos, son muy utilizadas las siguientes:

- Función triangular: definida mediante el límite inferior a, el superior b y el valor modal m, tal que $a<m<b$. La función no siempre es simétrica.
- Función trapezoidal: definida por sus límites inferior a, superior d, y los límites de soporte inferior b y superior c, tal que $a<b<c<d$. En este caso, si los valores de b y c son iguales, se obtiene una función triangular.

3.3. ¿Por qué emplear la lógica difusa?

Principalmente, la lógica difusa está enfocada a la toma de decisiones cuando existen datos o conocimientos inciertos, habiendo bastantes aplicaciones para la vida real y donde se reemplaza al operador humano por un sistema difuso basado en reglas. En este artículo, se eligió a esta debido a que ofrece varias ventajas, descritas como sigue:

- Al momento de dar un pronóstico, proporciona una manera sencilla y eficaz para extraer conclusiones de vaguedad, ambigua o información imprecisa. Por lo tanto, simula la toma de decisiones humanas y puede trabajar a partir de datos aproximados para obtener soluciones precisas.
- Incorpora una forma alternativa de pensar, lo que permite que todo pronóstico sea modelado con un nivel de abstracción que refleje conocimiento y experiencia a partir de reglas de inferencia.
- Este tipo de lógica permite expresar conocimiento con conceptos subjetivos, tales como los resultados que pueden ser obtenidos en el EGEL: ANS, TDS, TDSS, los cuales pueden ser mapeados de manera exacta dentro de rangos difusos.
- Es un método eficiente que rápidamente proporciona uno o más pronósticos como soluciones.
- Esta ofrece varios beneficios, tales como el rendimiento, simplicidad, bajo costo y productividad en el modelado de un sistema de pronóstico difuso, el cual es explicado en la siguiente sección.

4. Propuesta de modelo de pronóstico difuso

4.1. Análisis

El plan de estudios de ICO de la UAEM está comprendido de 430 a 450 créditos, de los cuales en el CU UAEM VM se cursan 434. En total, 355 horas son teóricas y 79 son prácticas, repartidas entre 64 UDAs.

El conjunto total de UDAs a partir de este momento será el siguiente universo del discurso X = {programación estructurada, seguridad en redes, fundamentos de bases de datos, métricas de software, …, programación paralela y distribuida} y el predicado P = "tener membresía para cada una de las áreas correspondientes del EGEL". Se define P como el subconjunto de UDAs con membresía a dichas áreas.

Por ejemplo, una UDA que sólo pertenece a un área es seguridad en redes, por lo que su función de pertenencia sería:

μ_{pSSCAE}(seguridad en redes) = 0;
μ_{pNTISC}(seguridad en redes) = 0;
$\mu_{pDHSAAE}$(seguridad en redes) = 0;
μ_{pAHSAE}(seguridad en redes) = 0;
μ_{pRCNE}(seguridad en redes) = 1;

Por otra parte, un caso especial de UDA con función de pertenencia en varias áreas es programación paralela y distribuida:

μ_{pSSCAE}(programación paralela y distribuida) = 0;
μ_{pNTISC}(programación paralela y distribuida) = 0.6;
$\mu_{pDHSAAE}$(programación paralela y distribuida) = 0;
μ_{pAHSAE}(programación paralela y distribuida) = 0;
μ_{pRCNE}(programación paralela y distribuida) = 0.4;

En el caso de seguridad en redes, se tiene una membresía con grado 1 en el área RCNE, mientras que programación paralela y distribuida cuenta con un grado de 0.4

en dicha área, lo cual indica que el 100% del contenido temático de seguridad en redes corresponde a esta área y sólo el 40% del contenido temático de programación paralela y distribuida cubre la misma área.

Al concluir la función de pertenencia, se aplica la siguiente ecuación para cada área:

$$POA = \sum (PUDA) \tag{3}$$

Donde POA: pertenencia obtenida por área.
PUDA: ponderación asignada a la UDA para esa área.

Posteriormente, para cada área se calcula un porcentaje de evaluación, considerando las calificaciones obtenidas por el alumno en cada UDA y la evaluación dada al docente de cada UDA, obteniendo la siguiente ecuación:

$$PA = (CUDA * 0.8 + CDO * 0.2) * PUDA. \tag{4}$$

Donde PA: es el porcentaje de área.
CUDA: calificación de la UDA.
CDO: calificación del docente.

Como se puede ver, a la calificación obtenida por el alumno en cada UDA se le asignó el 80% y a la calificación del docente se le otorgó el 20%. Esta asignación, va en función de que la evaluación docente pudiera no ser objetiva, sin embargo, es un factor a considerar para la apreciación del alumno.

4.2. Estimaciones

Con el análisis previo, es posible obtener las siguientes estimaciones:

- Estimación simple: en esta se considera el valor máximo de POA y los rangos de puntuación del IC, tal y como se muestra en la siguiente ecuación:

$$ES = \sum (PA) / Max (POA) * 6 + 700 \tag{5}$$

Donde ES: es la estimación simple por Área.

- Estimación basada en periodos de egreso: aquí se toma como referencia el periodo de egreso del alumno, suponiendo que por cada año de egreso se le resta 0.025%, con ello se tiene la ecuación:

$$EPE = ES * (1 - (0.025 * NAEG)) \tag{6}$$

Donde EPE: es la estimación por periodo de egreso.
NAEG: número de años de egreso.

- Estimación basada en años de experiencia: se considera la experiencia en el campo laboral, por cada año de experiencia se considera un incremento de 0.375 %, por lo que la ecuación queda de la siguiente manera:

$$EAE = ES * (1 + (0.0375 * NAEX)) \tag{7}$$

Donde EAE: es la estimación por años de experiencia.
NAEX: número de años de experiencia.

4.3. Pronóstico

Considerando las estimaciones anteriores, el siguiente paso es dar un pronóstico del resultado del EGEL. Para ello, se obtiene primero el promedio de las estimaciones descritas anteriormente, aplicando la siguiente ecuación:

$$PAE = (ES + EPE + EAE) / 3 \quad (8)$$

Donde PAE: es el pronóstico del área del EGEL.
La ecuación 8 se debe realizar para cada una de las áreas del EGEL.

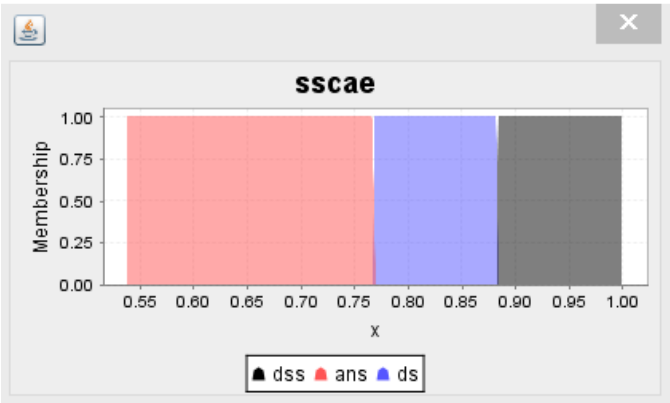


Fig. 5. Fusificación del área Selección de Sistemas Computacionales para Aplicaciones Específicas

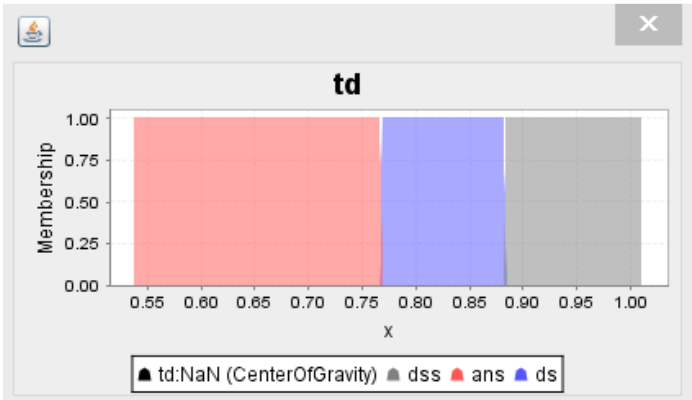


Fig. 6. Fusificación del Testimonio de Desempeño

La fusificación y defusificación se llevó a cabo empleando jfuzzylogic [9]. A cada una de las áreas de evaluación del EGEL para ICO, se le asignaron los mismos intervalos difusos, ANS, DS y DSS (ver fig. 5 y 6). Como ejemplo, se muestra a continuación lo hecho para el área SSCAE.

```
FUZZIFY sscae
TERM ans:= (0.5384, 1) (0.7669, 1) (0.7700, 0);
TERM ds:= (0.7676, 0) (0.7692, 1) (0.8823, 1) (0.8853, 0);
TERM dss := (0.8830, 0) (0.8846, 1) (1, 1);
END_FUZZIFY
```

En la defusificación, se necesitan 25 reglas difusas para obtener el TD del pronóstico difuso del resultado del EGEL, de las cuales como muestra se presentan las 5 reglas para obtener el TDSS:

```
RULE 11: IF ((sscae IS ds OR sscae IS dss) AND (ntisc IS ds OR ntisc IS dss)  AND
          (dhsaae IS ds OR dhsaae IS dss) AND (ahsae IS ds OR ahsae IS dss))
THEN td IS ds;
RULE 12: IF ((sscae IS ds OR sscae IS dss) AND (ntisc IS ds OR ntisc IS dss) AND
           (dhsaae IS ds OR dhsaae IS dss) AND (rcne IS ds OR rcne IS dss))
THEN td IS ds;
RULE 13: IF ((sscae IS ds OR sscae IS dss) AND (ntisc IS ds OR ntisc IS dss) AND
           (ahsae IS ds OR ahsae IS dss) AND (rcne IS ds OR rcne IS dss))
THEN td IS ds;
RULE 14: IF ((sscae IS ds OR sscae IS dss) AND (dhsaae IS ds OR dhsaae IS dss)
           AND (ahsae IS ds OR ahsae IS dss) AND (rcne IS ds OR rcne IS dss))
THEN td IS ds;
RULE 15: IF ((ntisc IS ds OR ntisc IS dss) AND (dhsaae IS ds OR dhsaae IS dss)
           AND (ahsae IS ds OR ahsae IS dss) AND (rcne IS ds OR rcne IS dss))
THEN td IS ds;
```

4.4. Caso de estudio

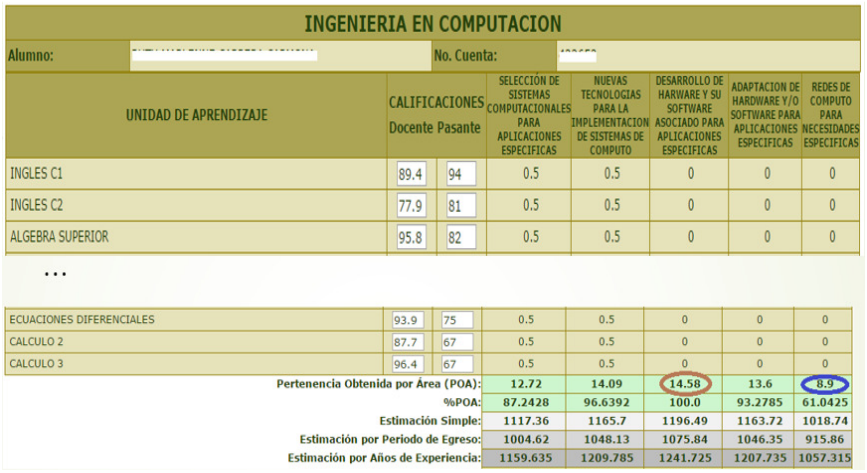
En la última aplicación del EGEL, con fecha del 28 de noviembre de 2014, celebrada en el CU UAEM VM, un sustentante elegido al azar obtuvo los siguientes puntajes con un TD dado por un TDS:

SSCAE = 1039
NTISC = 1134
DHSAAE = 1070
AHSAE = 1047
RCNE = 1001

Ahora bien, al aplicar el modelo de pronóstico difuso aquí propuesto, se introdujeron las calificaciones obtenidas por el alumno en cada una de las UDAs y las calificaciones que dicho alumno otorgó a sus profesores en el proceso de apreciación estudiantil, para obtener las estimaciones y el pronóstico descritos anteriormente.

De acuerdo a las estimaciones mostradas en la fig. 7, se observa que en la carrera de ICO, el área Desarrollo de Hardware y su Software Asociado para Aplicaciones Específicas, es la de mayor presencia y la de menor presencia es el área de Redes de Cómputo para Necesidades Específicas en el plan de estudios, respectivamente. Cabe mencionar, que el EGEL evalúa todas las áreas de manera uniforme y la distribución aquí obtenida no lo es. Dadas las estimaciones obtenidas, el pronóstico para cada una de las áreas es el siguiente:

PAE SSCAE = 1094.1
PAE NTISC = 1117.1
PAE DHSAAE = 1146.6
PAE AHSAE = 1115.2
PAE RCNE = 976.29



INGENIERIA EN COMPUTACION

Alumno: [redacted] No. Cuenta: [redacted]

UNIDAD DE APRENDIZAJE	CALIFICACIONES Docente	Pasante	SELECCIÓN DE SISTEMAS COMPUTACIONALES PARA APLICACIONES ESPECIFICAS	NUEVAS TECNOLOGIAS PARA LA IMPLEMENTACION DE SISTEMAS DE COMPUTO	DESARROLLO DE HARWARE Y SU SOFTWARE ASOCIADO PARA APLICACIONES ESPECIFICAS	ADAPTACION DE HARDWARE Y/O SOFTWARE PARA APLICACIONES ESPECIFICAS	REDES DE COMPUTO PARA NECESIDADES ESPECIFICAS
INGLES C1	89.4	94	0.5	0.5	0	0	0
INGLES C2	77.9	81	0.5	0.5	0	0	0
ALGEBRA SUPERIOR	95.8	82	0.5	0.5	0	0	0
...							
ECUACIONES DIFERENCIALES	93.9	75	0.5	0.5	0	0	0
CALCULO 2	87.7	67	0.5	0.5	0	0	0
CALCULO 3	96.4	67	0.5	0.5	0	0	0
Pertenencia Obtenida por Área (POA):			12.72	14.09	14.58	13.6	8.9
%POA:			87.2428	96.6392	100.0	93.2785	61.0425
Estimación Simple:			1117.36	1165.7	1196.49	1163.72	1018.74
Estimación por Periodo de Egreso:			1004.62	1048.13	1075.84	1046.35	915.86
Estimación por Años de Experiencia:			1159.635	1209.785	1241.725	1207.735	1057.315

Fig. 7. Estimaciones del EGEL para Ingeniería en Computación

Por tanto, es aquí donde pudieran entrar las estrategias a seguir para fortalecer las áreas más desprotegidas. No obstante, para este sustentante el TD pronosticado del EGEL al que el mismo aspiraría es un TDS, el cual corresponde exactamente al TD real. Con esto, realizando un comparativo de la resta de los puntajes reales menos los puntajes del pronóstico se puede obtener la ecuación siguiente:

$$CA = \text{puntaje obtenido} - PAE \tag{9}$$

Donde CA: es el comparativo por área de los puntajes reales con los puntajes del pronóstico de cada área.

CA SSCAE = -55.07
CA NTISC = 16.87
CA DHSAAE = -76.63
CA AHSAE = -68.22
CA RCNE = 24.70

Lo anterior dice, que el pronóstico dado puede alejarse o acercarse en promedio en un valor de ± 31.67, lo cual quiere decir que en cada área el resultado pronosticado puede beneficiar o perjudicar al sustentante en dicho valor. No obstante, se debe contemplar además que por cada área dicho valor puede ser muy diferente.

Calculando el porcentaje del pronóstico de cada área se puede dar la siguiente ecuación:

$$PPA = 100 - (\text{valor absoluto } (100\ (PAE * 100) / \text{valor obtenido por área})) \quad (10)$$

Donde PPA: es el porcentaje del pronóstico del área.
Los valores de los porcentajes del pronóstico obtenidos son:
PPA SSCAE = 94.69
PPA NTISC = 98.51
PPA DHSAAE = 92.83
PPA AHSAE = 93.48
PPA RCNE = 97.53

Finalmente, los valores anteriores dicen que la aproximación del pronóstico generado es considerablemente acertada, ya que en promedio se tuvo un porcentaje de acierto del 95.40.

5. Conclusiones

El análisis y creación del modelo de pronóstico difuso aquí propuesto para el TD del EGEL de los sustentantes de ICO de la UAEM que opten por dicha modalidad de titulación, ha permitido detectar principalmente los siguientes aspectos a considerar:

1. El objetivo es acreditar el EGEL y no necesariamente medir el nivel de conocimiento adquirido.
2. Existen deficiencias notorias en el actual PE, esto va dado con respecto a las áreas que evalúa el EGEL.
3. El porcentaje de UDAs que impactan en el área de RNCE es bajo, por lo que, difícilmente un sustentante logrará alcanzar un TDS, a menos que éste, tenga experiencia en dicha área.
4. No se esperaría que exista un gran porcentaje de resultados reales con un TDSS.

Por lo anterior, si se desea que el EGEL sea una modalidad que cumpla con la meta establecida en el plan rector de la UAEM y que los egresados sean capaces de utilizar lo que han estudiado y aprendido en su licenciatura en situaciones similares a las que se enfrentarán en el ejercicio profesional, se requiere que se establezcan las siguientes medidas:

- Revisar la afinidad de contenidos de las UDAs del PE de ICO, con los reactivos que se aplican en las áreas del EGEL, a fin de adecuar y mejorar los contenidos temáticos y con ello se logre un mejor aprovechamiento de los alumnos. Por ende, un TDS es al menos el rango de puntuación deseado para aquellos que opten por el EGEL, un TDSS sería una ganancia adicional.

- Estructurar un programa intensivo de preparación para los sustentantes que vayan a presentar el EGEL, considerando las áreas de menor fortaleza en la evaluación sin descuidar aquellas con mayor presencia.

Referencias

1. SEP: Lineamientos para la formulación de indicadores educativos. Disponible en http://fs.planeacion.sep.gob.mx/estadistica_e_indicadores/lineamientos_formulacion_de_indicadores.pdf, recuperado en marzo 2015 (2014).
2. ANUIES: Anuario estadístico de educación superior. Disponible en http://www.anuies.mx/iinformacion-y-servicios/informacion-estadistica-de-educacion-superior/anuario-estadistico-de-educacion-superior, recuperado en marzo 2015 (2012-2013)
3. UAEM: Agenda Estadística 2012-2013. Disponible en http://www.uaemex.mx/planeacion/Numeros.html, recuperado en marzo 2015
4. UAEM: Reglamento de evaluación profesional. Disponible en http://www.uaemex.mx/opcevl/, recuperado en marzo 2015
5. CENEVAL: Guía para el sustentante, EGEL para Ingeniería Computacional. Disponible en http://archivos.ceneval.edu.mx/archivos_portal/18791/GuiadelEGEL-ICOMPU.pdf, recuperado en marzo 2015
6. Ponce, P.: Inteligencia artificial, con aplicaciones a la ingeniería. pp. 20–50, Alfaomega, México (2010)
7. Kumar, R.: Soft Computing and its Applications. vol. 1, pp. 213–287, Apple Academic Press (2015)
8. Chen, G., Pham, T., Introduction to Fuzzy Sets, Fuzzy Logic, and Fuzzy Control Systems. CRC Press (2000)
9. Cingolani, P., Alcala-Fdez, J.: jFuzzyLogic: a robust and flexible Fuzzy-Logic inference system language implementation. In: Fuzzy Systems (FUZZ-IEEE), 2012 IEEE International Conference on, IEEE, pp. 1–8 (2013)

Una generalización del clasificador Naive Bayes para usarse en bases de datos con dependencia de variables

Ana E. Ruiz L.[1,2], Christopher R. Stephens S.[3,4], Hugo Flores [1,3]

[1] IIMAS, Universidad Nacional Autónoma de México, México D.F.
[2] Instituto Tecnológico de Minatitlán, Minatitlán, Ver., México
[3] C3 Centro de Ciencias de la Complejidad Universidad Nacional Autónoma de México, México D.F.
[4] Instituto de Ciencias Nucleares, Universidad Nacional Autónoma de México, México D.F.

aeruizlinares@hotmail.com, stephens@nucleares.unam.mx, hugo_fh@yahoo.com

Resumen. A pesar de la suposición que hace sobre la independencia de variables, el Clasificador Naive Bayes es muy utilizado en Minería de Datos y Aprendizaje Automático debido principalmente a su relativa simpleza y robustez mostrados frente a gran cantidad de problemas. Al suponer una independencia de variables, el modelo de NB proporciona un modelo no representativo cuando la base de datos tiene variables dependientes. Ante esta situación, se han propuesto varias aproximaciones que mejoran el desempeño del NB pero requieren mayores recursos y resultan complicados de implementar. Aquí se propone una nueva aproximación que puede ser usada cuando exista dependencia de variables conservando una sencillez de implementación. También se propone una métrica para determinar a priori si utilizar la aproximación más simple del clasificador NB o no. Los resultados obtenidos en cuatro bases de UCI mostraron que el modelo propuesto mejora el desempeño del NB cuando existe dependencia de variables.

Palabras clave: Clasificación · Naive Bayes · dependencia de variables.

1. Introducción

Un modelo muy utilizado en Clasificación debido a su robustez y sencillez de implementación es el Clasificador Naive Bayes (CNB), sin embargo, la suposición que hace de que todos los atributos son condicionalmente independientes dada una clase no siempre se cumple en aplicaciones del mundo real, produciéndose un decremento en su desempeño [1].

Con el fin de atenuar el impacto de la independencia de variables se han propuesto varias aproximaciones al CNB, entre ellos: *Semi-NB Classifiers* [2, 3, 4, 5]; *The Tree Augmented Naive Bayes (TAN)* [6]; *Super Parent TAN* [7,8]; *Improved Naive Bayes (INB)* [9]; *Weighted NB* [10-15]; Taheri et al. en [16] proponen un algoritmo llamado *Attribute Weighted NB (AWNB)* que asigna más de un peso a cada atributo. En [17] se muestra un panorama general del desempeño de varios de estos modelos.

Aunque estas aproximaciones, en términos generales, han presentado mejor desempeño que el NB, utilizan más recursos computacionales y resultan más complejas en el momento de implementarlas. Aquí se propone una aproximación llamada Naive Bayes Generalizado (NBG), con dos variantes: "simétrico" y "no simétrico", que puede ser usado cuando existen dependencias o correlaciones entre dos variables y conserva la sencillez de implementación del NB. Además, se presenta el uso de una herramienta de diagnóstico, denominada épsilon (ε), para averiguar si existen variables correlacionadas en la base de datos a examinar, permitiendo determinar a priori si la aproximación simple del NB es adecuada o es necesario invertir más recursos en la implementación de alguna otra aproximación más sofisticada del método.

2. Naive Bayes Generalizado NBG

El modelo de NBG - a diferencia del NB, que sólo analiza la acción de una variable sobre una clase- permite examinar el impacto sobre la clase de dos variables que actúan de forma sinérgica (fig. 1); cuando en una base de datos, la interacción de dos variables incrementa la probabilidad de clase, se dice que las variables son dependientes o están correlacionadas.

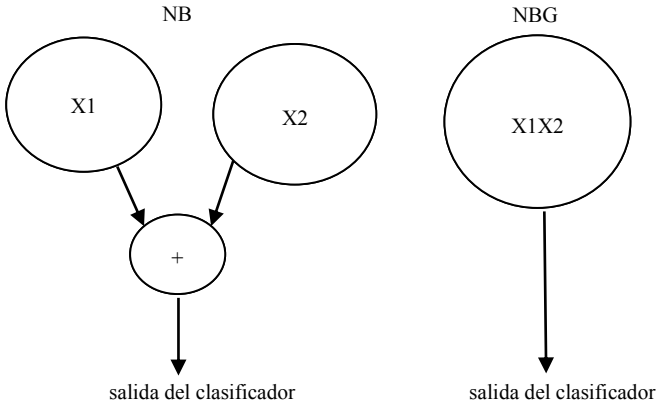


Fig. 1. Diferencia en el comportamiento de un clasificador NB y uno NBG. Se observa en el modelo de la derecha la acción independiente de cada variable, mientras que en el de la derecha se considera la acción conjunta de dos variables.

Al suponer una independencia de variables el modelo de NB proporciona un score de riesgo S_{NB} - función monotónica de la probabilidad condicional $P(C|\boldsymbol{X})$ siendo $\boldsymbol{X}$ un vector con n variables $\boldsymbol{X} = (x_1, x_2, \dots, x_{n)}$ y C la clase - determinado por la ec. 1, donde $P(C)$ es la probabilidad a priori de la clase y $P(\bar{C})$ la probabilidad a priori de la no clase. Con el fin de simplificar el análisis, supóngase que la base de datos sólo tiene dos variables x_1 y x_2, entonces el score de riesgo alcanzado con un CNB se re-

duce a la ec. 2; mientras que el modelo NBG, al contemplar la acción conjunta de dos variables sobre la clase, obtiene un score de riesgo S_{NBG} especificado por la ec.3.

$$S_{NB} = \ln\frac{P(C)}{P(\bar{C})} + \sum_{i=1}^{n} ln\frac{P(X_i|C)}{P(X_i|\bar{C})} \tag{1}$$

$$S_{NB} = \ln\frac{P(C)}{P(\bar{C})} + ln\frac{P(X_1|C)}{P(X_1|\bar{C})} + ln\frac{P(X_2|C)}{P(X_2|\bar{C})} \tag{2}$$

$$S_{NBG} = \ln\frac{P(C)}{P(\bar{C})} + ln\left(\frac{P(X_1X_2|C)}{P(X_1X_2|\bar{C})}\right) \tag{3}$$

2.1 Épsilon ε

Stephens en [18] plantea el uso de ε como una medida sobre el impacto que tiene cierto valor de variable sobre la clase, mencionando que valores ε>2 indican que ese valor de variable es predictiva de la clase, mientras que valores ε<-2 indican que es predictiva para no clase. Aquí se plantea el uso de ec. 4 para determinar si la combinación de $X_1 = i$ con $X_2 = j$ es predictiva de clase y ec. 5 para no clase.

N_C es el número de ocurrencia de clase; $N_{\bar{C}}$ es el número de ocurrencias para no clase; el número de ocurrencias de $X_1 = i$ y $X_2 = j$ con la clase es $P(X_{1i}X_{2j}|C)$ y con la no clase $P(X_{1i}X_{2j}|\bar{C})$; el número de coincidencias de $X_1 = i$ con la clase es $P(X_{1i}|C)$ y $P(X_{1i}|\bar{C})$ con la no clase; finalmente, el número de coincidencias de $X_2 = j$ con la clase es $P(X_{2j}|C)$y con la no clase es $P(X_{2j}|\bar{C})$.

$$\varepsilon_C = \frac{N_C\left(P(X_{1i}X_{2j}|C) - P(X_{1i}|C)P(X_{2j}|C)\right)}{\sqrt{N_C P(X_{1i}|C)P(X_{2j}|C)\left(1 - P(X_{1i}|C)P(X_{2j}|C)\right)}} \tag{4}$$

$$\varepsilon_{\bar{C}} = \frac{N_{\bar{C}}\left(P(X_{1i}X_{2j}|\bar{C}) - P(X_{1i}|\bar{C})P(X_{2j}|\bar{C})\right)}{\sqrt{N_{\bar{C}} P(X_{1i}|\bar{C})P(X_{2j}|\bar{C})\left(1 - P(X_{1i}|\bar{C})P(X_{2j}|\bar{C})\right)}} \tag{5}$$

El numerador de ec. 4 es la diferencia entre el número real de ocurrencias de $X_1 = i$ y $X_2 = j$ con la clase C y el número esperado de acuerdo con la hipótesis nula de que X_1 y X_2 son independientes de C, esto es, la hipótesis nula es la aproximación de Naive Bayes. El numerador de ec. 5 mide esta diferencia con respecto a no clase, $\bar{C}$. Los denominadores de las ec 4 y 5 están asociados con las desviaciones estándar de sus respectivos numeradores.

Ya que tanto ec. 4 como ec. 5 emplean una distribución binomial, valores $|\varepsilon_C| > 2$ y/o $|\varepsilon_{\bar{C}}| > 2$ representarían el intervalo de confianza del 95 % estándar y serían estadísticamente significativos para considerar $X_1 = i$ y $X_2 = j$ correlacionadas para clase y/o no clase. La idea es que si ε_C y/o $\varepsilon_{\bar{C}}$ son significativos: $X_1 = i$ y $X_2 = j$ son variables dependientes y el desempeño del modelo NB se verá reducida, en este caso se recomendaría el uso de alguna otra aproximación al modelo que considera las dependencias de variables, tal como lo hace el NBG.

En [19] utilizan ε como primer paso en la búsqueda de valores de variables que tienen mayor influencia en la predicción de clase. En [20] usan ε para medir la dependencia estadística de un taxón relativa a una hipótesis nula de independencia.

2.2 NBG "simétrico", NBG1

Se denomina simétrico porque se considera que una combinación $X_{1i}X_{2j}$ con un valor significativo de ε_C y/o $\varepsilon_{\bar{C}}$ tiene el mismo impacto en la predictibilidad de la clase y de la no clase. El clasificador que usa esta aproximación calcula primero el score de todas las combinaciones significativamente dependientes. Si la unión de esas dos variables no es estadísticamente significativa, se considera que cada variable actúa de forma independiente sobre la clase y el clasificador calcula el score de esas dos variables usando el modelo NB. Matemáticamente, el comportamiento del clasificador se muestra en la ec. 6.

$$S_{NBG1} = \sum ln\left(\frac{P(X_{\mathrm{lk}}X_{\mathrm{nm}}|C)}{P(X_{\mathrm{lk}}X_{\mathrm{nm}}|\bar{C})}\right) + \sum ln\frac{P(X_i|C)}{P(X_i|\bar{C})} + \ln\frac{P(C)}{P(\bar{C})} \tag{6}$$

2.3 NBG "no simétrico", NBG2

Esta otra variante se denomina no simétrico porque el clasificador considera que una combinación $X_{1i}X_{2j}$ no necesariamente tiene el mismo impacto en la predictibilidad de la clase que de la no clase. El clasificador calcula el score para clase (ec. 7) o el score para no clase (ec. 8) dependiendo si la combinación es considerada dependiente para clase o para no clase. Si la combinación de variables es estadísticamente independiente, utiliza el modelo de NB para calcular el score de esas dos variables por separado. Matemáticamente, el comportamiento del clasificador se muestra en la ec. 9.

$$S_C = \ln P(X_{\mathrm{lk}}X_{\mathrm{mn}}|C) \tag{7}$$

$$S_{\bar{C}} = \ln P(X_{\mathrm{lk}}X_{\mathrm{mn}}|\bar{C}) \tag{8}$$

$$S_{NBG2} = \sum S_C - \sum S_{\bar{C}} + \sum ln\frac{P(X_i|C)}{P(X_i|\bar{C})} + \ln\frac{P(C)}{P(\bar{C})} \tag{9}$$

2.4 Corrección de Laplace

En [21] Provest y Domingos muestran que la estimación de la probabilidad mejora si en vez de calcular la probabilidad basada en frecuencia, se utiliza la Corrección de Laplace para suavizar dicho cálculo (ec. 10).

$$P(X_i|C) = \frac{N_{X_iC}+1}{N_C+k} \tag{10}$$

donde N_{X_iC} es el número de ocurrencias en la base de datos donde aparece el valor de la variable X_i en el subconjunto de registros de la clase, N_C es el total de registros de la base de datos y k es el total de valores que puede tener la clase. En este trabajo se utilizó la corrección de Laplace para el cálculo de todas las probabilidades.

3. Pruebas en base de datos de la UCI

En la tabla 1, se mencionan las bases de datos del repositorio de la Universidad de Irvine (UCI Repository) [22] con los cuales se probaron los modelos anteriores. Antes de generar los modelos de entrenamiento de los tres clasificadores - NB, NBG1 y NBG2- cada una de las bases de datos fueron divididas en un conjunto de entrenamiento y un conjunto de prueba. Para conocer a priori la existencia de variables correlacionadas en las bases se utilizaron los conjuntos de entrenamiento para calcular los valores de ε_C y $\varepsilon_{\bar{C}}$ que se muestran en las figuras 2, 3, 4, 5, 6, 7, 8 y 9; en estas figuras, el eje X es el valor de ε y el eje Y es el número de combinaciones $X_1 = i$ y $X_2 = j$.

Tabla 1. Bases usadas para probar el modelo. Nombre original: nombre bajo el cual se puede encontrar en el repositorio de UCI; Nombre traducido: nombre con el cual se designará en el presente trabajo; No. Reg: número de registros de la base; No. Atrib: número de atributos; P(C): probabilidad de la clase.

Nombre original	Nombre traducido	No. Reg	No. Atrib	P (C)
Breast cancer Wisconsin	Cáncer de mama	699	10	0.34
Mushroom	Hongos	8124	22	0.48
Tic Tac Toe Endgame	Tic Tac Toe	958	9	0.65
Congressional Voting Reords	Votos	435	16	0.61

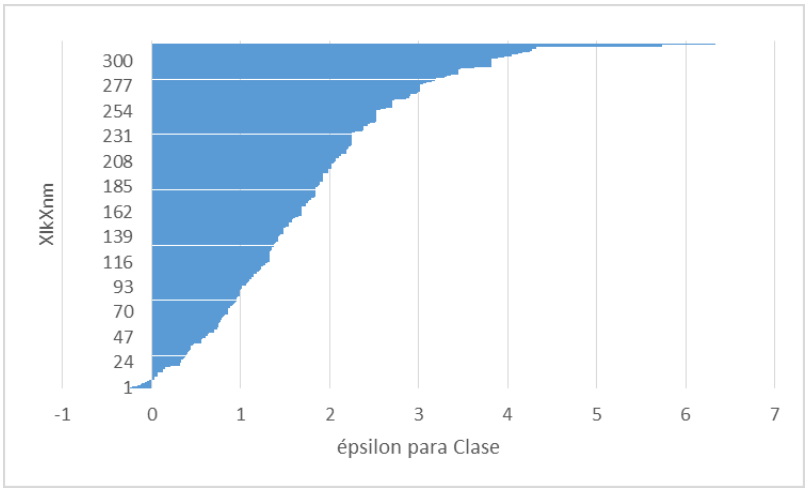


Fig. 2. Gráfico de valores de ε_C para la base de Cáncer.

Después de calcular ε_C y $\varepsilon_{\bar{C}}$ se supo que: en la base de hongos el 83% de las combinaciones de valores de variables son estadísticamente significativas para decir que actúan de forma correlacionada, mientras que en la base de cáncer sólo lo son el 38% de las combinaciones. Además, debido a que en la base de hongos se alcanzan valores más altos de ε_C y $\varepsilon_{\bar{C}}$ que en la de cáncer, se dice que en la base de hongos hay combinaciones de variables que son fuertemente predictivas para clase y para no clase, a diferencia de la base de cáncer, donde se observa en las fig. 2 y fig. 3 que hay más variables dependientes para clase que para no clase.

Lo anterior pudo ser la razón por la cual Pazzani en [23] encontrara mejores resultados con sus algoritmos que manejan dependencia de variables con respecto al NB en la base de hongos y no así en la de cáncer.

En la base de Tic tac toe alrededor de un 21 % de las combinaciones $X_{\mathrm{lk}}X_{\mathrm{nm}}$ indican correlación pero los valores de los indicadores no son muy altos, como se puede apreciar en las fig. 6 y fig. 7. Sucede algo similar en la base de Votos (fig. 8 y fig. 9).

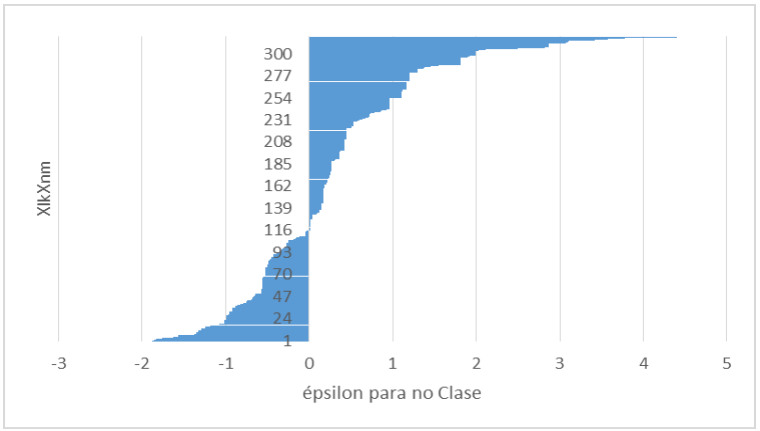


Fig. 3. Gráfico de valores de $\varepsilon_{\bar{C}}$ para la base de Cáncer.

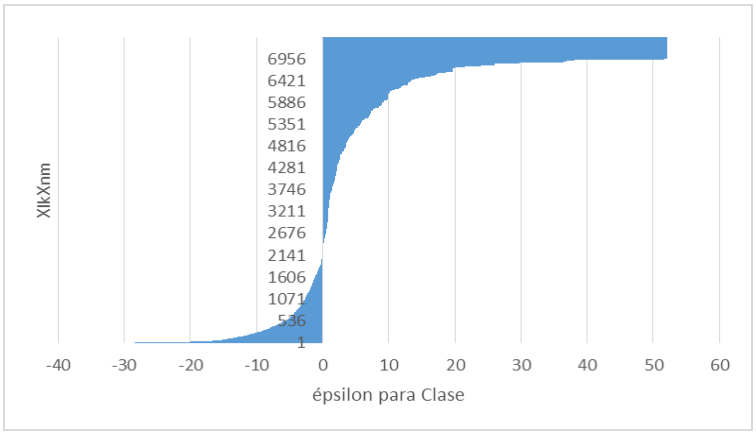


Fig. 4. Gráfico de valores de ε_C para la base de Hongos.

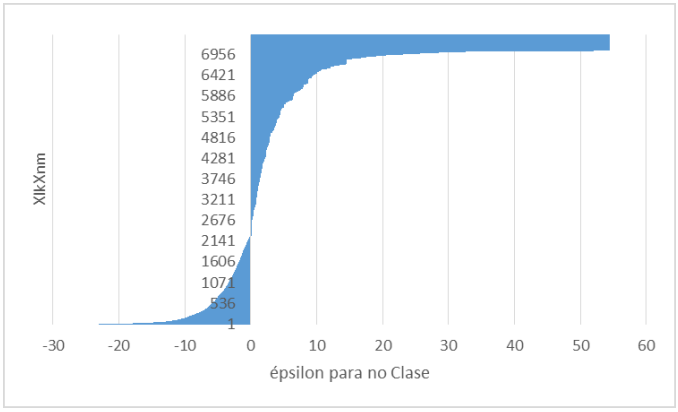


Fig. 5. Gráfico de valores de $\varepsilon_{\bar{C}}$ para la base de Hongos.

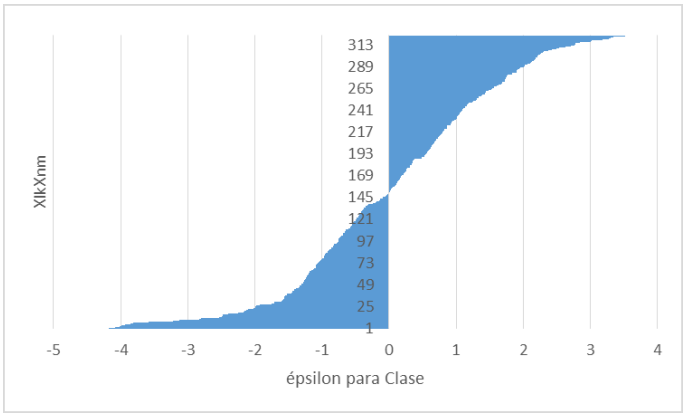


Fig. 6. Gráfico de valores de ε_{C} para la base de Tic tac toe.

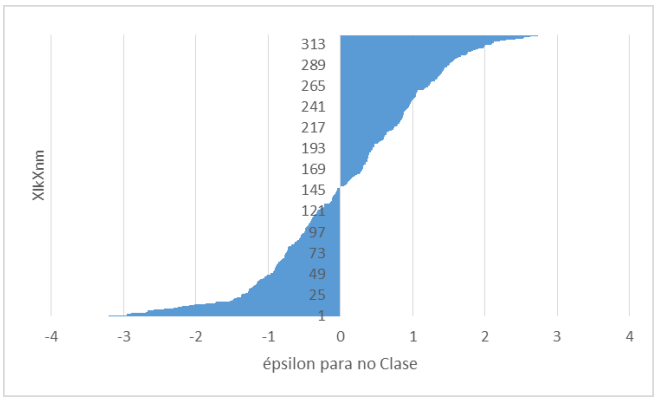


Fig. 7. Gráfico de valores de $\varepsilon_{\bar{C}}$ para la base de Tic tac toe.

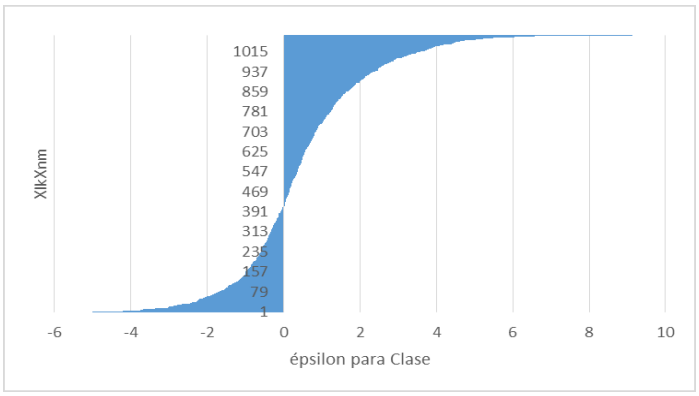


Fig. 8. Gráfico de valores de ε_C para la base de Votos.

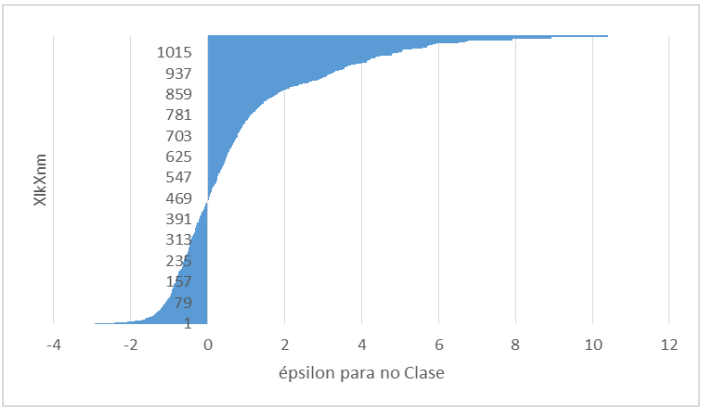


Fig. 9. Gráfico de valores de $\varepsilon_{\bar{C}}$ para la base de Votos.

Tabla 2. Matriz de confusión de la base de Cáncer de mama.

		salida del clasificador					
	CÁNCER	NB		NBG1		NBG2	
		clase	no clase	clase	no clase	clase	no clase
real	clase	75	1	75	1	75	1
	no clase	3	131	4	130	3	131

Tabla 3. Matriz de confusión de la base de Hongos.

		salida del clasificador					
	HONGOS	NB		NBG1		NBG2	
		clase	no clase	clase	no clase	clase	no clase
real	clase	1075	117	1175	17	1140	52
	no clase	5	1240	6	1239	4	1241

Después de correr los tres clasificadores en cada base de dato, se obtuvieron las matrices de confusión (tabla 2, 3, 4, 5) así como el desempeño de cada uno en términos de *sensibilidad*, *especificidad*, *eficiencia* y *error* (tabla 6).

Tabla 4. Matriz de confusión de la base de Tic tac toe.

		salida del clasificador					
	TIC TAC TOE	NB		NBG1		NBG2	
		clase	no clase	clase	no clase	clase	no clase
real	Clase	166	19	153	32	159	26
	no clase	62	41	52	51	48	55

Tabla 5. Matriz de confusión de la base de Votos.

		salida del clasificador					
		NB		NBG1		NBG2	
		clase	no clase	clase	no clase	clase	no clase
real	clase	64	12	64	12	64	12
	no clase	5	49	3	51	2	53

Tabla 6. Desempeño de los tres clasificadores aplicados a la bases de datos.

		SENSIBILIDAD	ESPEC	EFICIENCIA	ERROR
CÁNCER	NB	0.99	0.98	0.98	0.02
	NBG1	0.99	0.97	0.98	0.02
	NBG2	0.99	0.98	0.98	0.02
HONGOS	NB	0.9	1	0.95	0.05
	NBG1	0.99	1	0.99	0.01
	NBG2	0.96	1	0.98	0.02
TIC TAC TOE	NB	0.9	0.4	0.72	0.28
	NBG1	0.83	0.5	0.71	0.29
	NBG2	0.86	0.53	0.74	0.26
VOTOS	NB	0.84	0.91	0.87	0.13
	NBG1	0.84	0.94	0.88	0.12
	NBG2	0.84	0.96	0.89	0.11

Tabla 7. Desempeño promedio de 20 corridas de los tres clasificadores.

		SENSIBILIDAD	ESPEC	EFICIENCIA	ERROR
CÁNCER	NB	0.98	0.97	0.97	0.03
	NBG1	0.99	0.97	0.97	0.03
	NBG2	0.99	0.97	0.98	0.02
HONGOS	NB	0.91	0.99	0.95	0.05
	NBG1	0.99	0.99	0.99	0.01
	NBG2	0.97	1	0.98	0.02
TIC TAC TOE	NB	0.85	0.42	0.71	0.29
	NBG1	0.82	0.5	0.71	0.29
	NBG2	0.82	0.55	0.73	0.27
VOTOS	NB	0.88	0.92	0.9	0.1
	NBG1	0.89	0.93	0.9	0.1
	NBG2	0.89	0.95	0.91	0.09

En la tabla 7 se muestra el desempeño promedio de 20 corridas de los clasificadores. En la tabla 8 y tabla 9 se muestran las Curvas ROC (Receiver Operating Characteristics) y el área bajo la curva ROC conocido como AUC, ya que algunos trabajos [24,25] han mostrado que son una evaluación más discriminante que la razón de error en algoritmos de Aprendizaje Automático con estimaciones de probabilidad de clase. Se observa en dichas tablas que en la base de cáncer los tres clasificadores tienen el mismo valor de AUC, mientras que en las otras tres bases el valor de AUC es mayor en el modelo propuesto que el alcanzado con el CNB. Lo anterior es congruente con los valores obtenidos de ε.

4. Conclusiones y comentarios finales

Los resultados encontrados sugieren que el método propuesto, al tomar en cuenta la acción conjunta de dos valores de variables sobre la predictibilidad de la clase, mejora el desempeño del CNB en bases de datos donde existen variables correlacionadas. El grado de correlación o de dependencia se obtuvo a través de la herramienta de diagnóstico ε. Los valores que se logren con este indicador servirán para decidir si utilizar la aproximación más simple del clasificador NB o buscar otra más sofisticada.

Tabla 8. Curvas ROC y AUC de los tres clasificadores para la base de Cáncer (izquierda) y de Hongos (derecha). En el eje de las X está *(1-Especificidad)* y en el eje Y la *Sensibilidad*.

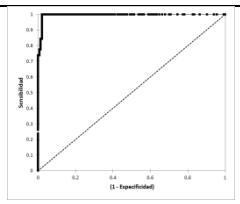
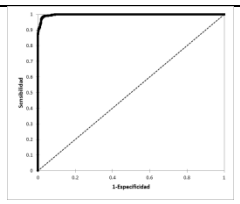
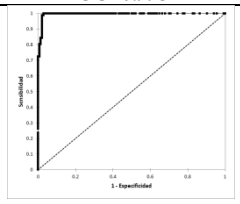
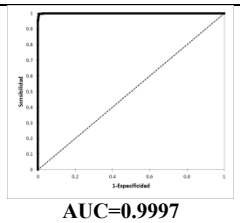
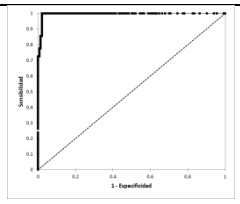
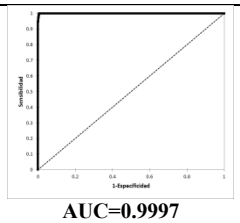
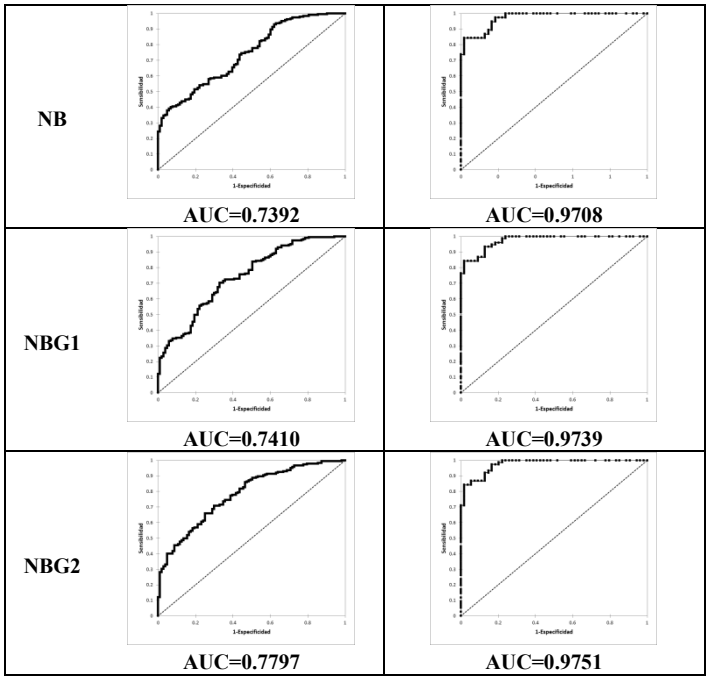
NB	 **AUC=0.9952**	 **AUC=0.9975**
NBG1	 **AUC=0.9952**	 **AUC=0.9997**
NBG2	 **AUC=0.9952**	 **AUC=0.9997**

Tabla 9. Curvas ROC y AUC de los tres clasificadores para la base de Tic tac toe (izquierda) y Votos (derecha). En el eje de las X está *(1-Especificidad)* y en el eje Y la *Sensibilidad*.



Aunque aquí sólo se consideran correlaciones entre un par de valores de variables, se puede buscar la correlación entre más de dos atributos haciendo múltiples pasos. Además, ya que los experimentos se hicieron sobre clasificaciones binarias, se tiene contemplado investigar el comportamiento del NBG con bases de datos multiclases. Resultaría también de gran interés, en un futuro, hacer una comparación en términos de complejidad computacional del método aquí propuesto con otras aproximaciones que también buscan atenuar la estricta restricción de independencia de variables que hace Naive Bayes.

Agradecimientos. Los autores agradecen el apoyo del proyecto PAPIIT-UNAM IN113414 para la realización de este trabajo.

Referencias

1. Jiang L, Wang D, Cai Z, Yan X.: Survey of Improving Naive Bayes for Classification. In: Berlin, Heidelberg: Springer Berlin Heidelberg; pp. 134–45 (2007)
2. Kohavi R: Scaling up the accuracy of naive-Bayes classifiers: a decision-tree hybrid. In: Proceedings of 2nd ACM SIGKDD International, Conference on Knowledge Discovery and Data Mining, pp 202–207 (1996)

3. Langley P, Saga S: Induction of selective Bayesian classifiers. In: Proceedings of tenth conference, uncertainty in artificial intelligence, Morgan Kaufmann, pp 399–406 (1994)
4. Pazzani MJ.: Constructive induction of Cartesian product attributes, ISIS: In-formation, Stat Induction Sci.,pp. 66–77 (1996)
5. Robles V, Larranaga P, Pena J M, Perez M S, Menasalvas E, Herves V.: Learning Semi Naive Bayes Structures by Estimation of Distribution Algorithms. Lecture Notes in Computer Science, vol. 292, pp. 244–58 (2003)
6. Friedman N, Geiger D, Goldszmidt M.: Bayesian network classifiers. Mach Learn, 29, pp. 131–163 (1997)
7. Keogh EJ, Pazzani MJ Learning augmented Bayesian classifiers: A comparison of distribution-based and classification based approaches. In: Proceedings of international workshop on artificial intelligence and statistics, pp. 225–230 (1999)
8. Webb GI, Boughton JR, Wang Z.: Not So Naive Bayes: Aggregating One-Dependence Estimators. Mach Learning, vol. 58, no. 1, pp. 5–24 (2005)
9. Taheri S, Mammadov M, Bagirov A.M.: Improving Naive Bayes classifier using conditional probabilities. In: Proceedings of ninth Australasian data mining conference (AusDM 2011), vol. 125, Ballarat, Australia (2011)
10. Zhang H, Sheng S.: Learning weighted Naive Bayes with accurate ranking. In: Proceedings of the 4th IEEE international conference on data mining, pp. 567–570 (2005)
11. Zhou Y, Huang T.S.: Weighted Bayesian network for visual tracking. In: Proceedings of the 18th international conference on pattern recognition (ICPR'O6) (2006)
12. Jiang L, Zhang H.: Weightily averaged one-dependence estimators. In: Proceedings of the 9th biennial pacific rim international conference on artificial intelligence, Guilin, China, pp. 970–974 (2006)
13. Hall M: A decision tree-based attribute weighting filter for Naive Bayes. Knowledge Based Systems , vol. 20, no. 2, pp.120–126 (2007)
14. Ozsen S, Gunecs S.: Attribute weighting via genetic algorithms for attribute weighted artificial immune system (AWAIS) and its application to heart disease and liver disorders problems. Expert Systems with Applications, vol. 36, no. 1, pp. 386–392 (2009)
15. Wu J, Cai Z.: Attribute weighting via differential evolution algorithm for attribute weighted Naive Bayes (WNB). Journal of Computational Information Systems, vol. 7, no. 5, pp.1672–1679 (2011)
16. Taheri S., Yearwood J., Mammadov M., Seifollahi S.: Attribute weighted Naive Bayes classifier using a local optimization. Neural Computing and Applications, vol. 24, no.5, pp. 995–1002 (2014)
17. Jiang L, Zhang H, Cai Z. A Novel Bayes Model: Hidden Naive Bayes. IEEE Trans Knowledge and Data Engineering, vol. 21, no.10, pp.1361–1371 (2009)
18. Stephens, C.: An introduction to data mining. In: Grover, R. and Vriens, M. (eds), The handbook of marketing research: uses, misuses and future advances. Sage Publ., pp. 445–486 (2006)
19. Stephens, C. R., Waelbroeck, H., and Talley, S.: Predicting healthcare costs using GAs. In: Proceedings of the 2005 workshops on Genetic and evolutionary computation, pp. 159–163, ACM (2005)
20. González-Salazar C., Stephens C.R., Marquet P.A.: Comparing the relative contributions of biotic and abiotic factors as mediators of species' distributions. Ecological Modelling, vol. 248, pp.57–70 (2013)
21. Provost F, Domingos P.: Tree Induction for Probability-Based Ranking. Machine Learning, vol. 52, no. 3, pp.199–215 (2003)
22. Murphy, P. M. , & Aha, D. W.: UCI Repository of machine learning databases. Irvine: University of California, Department of Information & Computer Science [Machine-readable data repository http://archive.ics.uci.edu/ml/datasets.html] (1995)

23. Pazzani, M. J.: Searching for dependencies in Bayesian classifiers. In: D. Fisher & H. Lenz (Eds.), Proceedings of the fifth International Workshop on Artificial Intelligence and Statistics, Ft. Lauderdale, FL (1995)
24. Bradley A.P.: The use of the area under the ROC curve in the evaluation of machine learning algorithms. Pattern Recognition, vol.30, no. 7, pp.1145–59 (1997)
25. Huang J., Ling C.X.: Using AUC and accuracy in evaluating learning algorithms. Knowledge and Data Engineering, IEEE Transactions on, vol.17, no. 3, pp.299–310 (2005)

Semantic Genetic Programming Operators Based on Projections in the Phenotype Space

Mario Graff, Eric Sadit Tellez, Elio Villaseñor, Sabino Miranda-Jiménez

INFOTEC - Centro de Investigación e Innovación en
Tecnologías de la Información y Comunicación,
Cátedras CONACyT,
México

{mario.graff,eric.tellez,
elio.villasenor,sabino.miranda}@infotec.com.mx

Abstract. In the Genetic Programming (GP) community there has been a great interest in developing semantic genetic operators. These type of operators use information of the phenotype to create offspring. The most recent approaches of semantic GP include the GP framework based on the alignment of error space, the geometric semantic genetic operators, and backpropagation genetic operators. Our contribution proposes two semantic operators based on projections in the phenotype space. The proposed operators have the characteristic, by construction, that the offspring's fitness is as at least as good as the fitness of the best parent; using as fitness the euclidean distance. The semantic operators proposed increment the learning capabilities of GP. These operators are compared against a traditional GP and Geometric Semantic GP in the Human oral bioavailability regression problem and 13 classification problems. The results show that a GP system with our novel semantic operators has the best performance in the training phase in all the problems tested.

Keywords: semantic crossover, symbolic regression, geometric semantic genetic programming.

1 Introduction

Genetic Programming (GP) is an evolutionary algorithm that has received a lot of attention lately due to its success in solving hard real-world problems [11]. One of the most promising ideas to improve the performance of GP is to develop semantic genetic operators. The difference between semantic operators and traditional genetic operators is that the former uses the information of the phenotype to generate offspring, and, the later generates an offspring using only the syntax of the individual.

There has been a number of different proposals that fit in the field of semantic operators and systems. Recently, Stefano Ruberto *et al.* [15] introduced the concept of error vector and error space. Briefly, the idea is that the optimal

solution can be constructed from two individuals that are aligned in the vector space. Consequently, the objective of the search procedure can be changed from finding the closest individual to the desired behavior (i.e., the origin in the error space) to finding aligned individuals. This novel type of GP has shown success in solving two complex real-life applications in drug discovery, namely, human oral availability and median lethal dose.

Semantic genetic operators gather the information from the phenotype space by either sampling it or using some properties of the space. Among the operators that sample the space, it is found the work done by Blickle *et al.* [1], who proposed to select as crossing points only those nodes that have an impact on the fitness function. The work done by Nguyen *et al.* [8,16] produces offspring that are semantically different from its parents; this difference is measured by evaluating the individuals in a set of random inputs.

Backpropagation [13] can be used in subtree crossover, subtree mutation, and point mutation to propagate the error —i.e., the information of the phenotype— to the crossover or mutation point. The information propagated can be used to guide the genetic operator, that is, it can be used to either generate a tree that reduces the error propagated or to select, from a second parent, the point that reduces the most the propagated error. Backpropagation was used by Pawlak *et al* [9] to find a subtree that analytically reduces the error produced in the crossover point. Graff *et al.* (see [4,5]) used backpropagation to compute the partial derivative error in the crossover and mutation point. Then this information was used to either select the best crossover point in the second parent or, in the case of point mutation, to select the best function from the function set.

The semantic operators most related to this contribution are the Geometric Semantic Crossover and Mutation proposed by Moraglio *et al.* [7] and the novel implementation of them developed by Vanneschi *et al.* [18] that allows the algorithm to be executed with traditional GP parameters. For a recent review in semantic operators we referred the interest reader to [19].

This contribution presents a novel semantic crossover and mutation, PrXO and PrMut, respectively. These operators have as their principal feature that the fitness (using as fitness the euclidean distance) of the offspring is at least as good as the fitness of the best parent. The offspring produced by PrXO and PrMut is the orthogonal projection of the target behaviour in the parents' plane in the phenotype space. That is, traditionally, the phenotype space is $\Re^{|\mathbb{T}|}$ (see [10]), where $\mathbb{T}$ is the training set. Therefore, each individual in the population is a point in $\Re^{|\mathbb{T}|}$ and the objective is to find a target behaviour $\mathbf{t} \in \Re^{|\mathbb{T}|}$. Under these circumstance the offspring produced by PrXO and PrMut is the orthogonal projection of $\mathbf{t}$ in the plane generated by the linear combinations of the parents. As the result show the use of PrXO and PrMut in a steady state GP system, namely hereafter PrGP, outperform, in the training set, a steady state Geometric Semantic GP and a steady state GP in two classes of problems: Human oral bioavailability problem and 13 classification problems. This is a clear indication of the learning capabilities of PrGP; however, more work is needed to

identify the overfitting.

The rest of the paper is organized as follows. Section 2 presents the semantic genetic operators based on projections in the phenotype space. The problems and the parameters setting used to test the novel semantic operators are described in Section 3. Section 4 presents our results and compared the operators against the Semantic Geometric GP and traditional GP. Finally, the conclusions and some possible directions for future work are given in Section 5.

2 Projection Semantic Genetic Operators

PrGP is a supervised learning algorithm that learns the instances of a training set $\mathbb{T}$ formed by $n \in \mathbb{N}$ pairs of inputs and outputs, i.e., $\mathbb{T} = \{(x_i, y_i) | i = 1 \dots n\}$ where x_i represents the i-th input, and y_i is the associated output. The objective is to find a function f such that $\forall_{(x,y)\in\mathbb{T}}\ f(x) = y$ and that could be evaluated in any element x of the input space.

In general, it is not possible to find a function f that learns perfectly $\mathbb{T}$, consequently, one tries to find a function f that minimize an error function e.g. sum of squared errors $\sum_{(x,y)\in\mathbb{T}}((y - f(x))^2$.

Let us consider a fixed order in $\mathbb{T}$ to define $\mathbf{t} = (y_1, \dots, y_n) \in \mathbb{R}^n$, namely the target vector, which contains all the outputs in $\mathbb{T}$. Let $s(p, x)$ be a function that evaluates the individual p on input x. Using the order in $\mathbb{T}$, it is possible to define $\mathbf{p} = (s(p, x_1), \dots, s(p, x_n))$ that contains the evaluation of individual p in all the inputs x of the training set. In this scenario the fitness (using as fitness function the sum of squared error) of individual p can be computed as $||\mathbf{t} - \mathbf{p}||$ which is the euclidean norm.

A source of inspiration for PrXO and PrMut is the geometric semantic genetic operators proposed by Moraglio *et al.* [7] which are defined as follows:

Geometric Semantic Crossover Let p_1 and p_2 be the first and second parent the offspring produce by these parent is $o = p_1 r + p_2(1 - r)$, where r is a random function or a constant in the range $[0, 1]$. The output of individual o at input x is computed as $s(o, x) = s(p_1, x)s(r, x) + s(p_2, x)(1 - s(r, x))$.

Geometric Semantic Mutation Let p_1 be the individual to be mutated and r_1 and r_2 two random functions, then the offspring produced is $o = p_1 + m(r_1 - r_2)$ where m is the mutation step. Vanneschi *et al.* [18] proposed a variant of this operation, the difference is that r_1 and r_2 are normalized using a sigmoid.

Let us assume that r in the geometric semantic crossover is a constant, then the offspring is just a linear combination of the parents. This combination lies in the line segment intersecting the parents.

This characteristic influenced the development of PrXO and PrMut. That is, it is reasonable to investigate that whether there is a better linear combination between the two parents, and, effects that this modifications has in the converge of the algorithm.

Let us rewrite the geometric crossover and mutation with the constraint that r is a constant. The geometric crossover is computed as $o = \alpha p_1 + \beta p_2$ where $\alpha = r$ and $\beta = 1 - \alpha$; and the geometric mutation is $o = \alpha p_1 + \beta(r_1 - r_2)$ where $\alpha = 1$ and $\beta = m$.

Using this notation it is evident that the geometric operators constraints the values of α and β. Operators PrXO and PrMut removes these restrictions but are not geometric as defined by Moraglio *et al.* [7].

PrXO Let p_1 and p_1 be the first and second parent, then the offspring o is computed as $o = \alpha p_1 + \beta p_2$ where α and β are calculated solving the following equation $A\dot{(}\alpha, \beta)' = \mathbf{t}'$ where $A = (\mathbf{p}'_1, \mathbf{p}'_2)$, $\mathbf{p}_i = (s(p_i, x_1), \ldots, s(p_i, x_n))$ is the evaluation of parent i in all the inputs, and $\mathbf{t}$ is the target vector.

PrMut Let p_1 be the parent to be mutated then the offspring is $o = \alpha p_1 + \beta(r_1 - r_2)$ where r_1 and r_e are two random individuals and α and β are obtained by solving $A\dot{(}\alpha, \beta)' = \mathbf{t}'$ where $A = (\mathbf{p}'_1, (\mathbf{r}_1 and \mathbf{r}_2)')$.

By construction the offspring o generated by PrXO and PrMut is the projection of $\mathbf{t}$ on the plane produced by the linear combination of $\mathbf{p}_1$ and $\mathbf{p}_2$ in the case of crossover; or $\mathbf{p}_1$ and $(\mathbf{r}_1 - \mathbf{r}_2)$ otherwise. Given that $\mathbf{o}$ is the projection of $\mathbf{t}$ then it is the closest point to $\mathbf{t}$ in the plane, consequently if the fitness function is the euclidean distance then the offspring has at least the fitness of the best parent. In the implementation it was decided to discard any individual that has an equal fitness to its parents.

3 Problems and Parameters Settings

PrXO and PrMut are tested on two classes of problems, symbolic regression and classification. The symbolic regression problem is the *Human oral bioavailability*[1] problem previously used in [18]. The problem is to find a function that accurately predicts the percentage of the initial orally submitted drug dose that effectively reaches the system blood circulation after passing through the liver. It consists of 241 inputs, namely molecular descriptor that describe the drug and 359 instances.

The classification problems used are described in Table 1. [2] These problems have been used traditionally as benchmarks for classification methods, these present different characteristics in terms of number input features, size of the training set, and test set, among other.

PrXO and PrMut are compared on the two previous classes of problem against the geometric semantic crossover and mutation; and the traditional

[1] This dataset is available at http://gpbenchmarks.org

[2] These datasets are available in http://theoval.cmp.uea.ac.uk/matlab/benchmarks/

Table 1. Data sets used in the comparison.

Data Set	Input Features	Training set instances	Test set instances
Banana	2	400	4900
Titanic	3	150	2051
Thyroid	5	140	75
Diabetes	8	468	300
Breast-Cancer	9	200	77
Flare-Solar	9	666	400
Heart	13	170	100
Ringnorm	20	400	7000
Twonorm	20	400	7000
German	20	700	300
Image	20	1300	1010
Waveform	21	400	4600
Splice	60	1000	2175

subtree crossover and mutation. In order to make the comparison as fair as possible, each the genetic operators were used in a steady state evolution with tournament selection (tournament size 2). The system with PrXO and PrMut is denominated PrGP (the source code is available for download at https://...) the system with the geometric genetic operators are identified as GSGP, and, the system with the traditional genetic operators is labeled GP. The rest of the parameters used in all the systems are shown on Table 2.

Table 2. Parameters used for PrGP, GSGP and GP on the symbolic regression problem (Human oral bioavailability) and the classification problems

Parameter	Symbolic Regression	Classification
Mutation depth	random $\in [1, 5]$	
Selection	Tournament of size 2	
Population Size	100	1000
Number of Generations	500	50
Function Set ($\mathcal{F}$)	$\{+, -, \times, /\}$	$\{+, -, \times, /, \lvert\cdot\rvert, \exp, \sqrt{}, \sin, \cos, \text{sigmoid}, \text{if}, \max, \min, \ln, (\cdot)^2, \text{argmax}\}$
Crossover rate	50%	90%
Mutation rate	50%	10 %
Max length (only on GP)	1024	$\min(\frac{\lvert\mathbb{T}\rvert}{2}, 256)$

As can be seen, the parameters used in GP are standard having been used previously e.g., [12] and [6]. Nonetheless, there are some parameters that deserve an explanation. The difference in the crossover rate and population size between the regression and classification problems is because GSGP obtained better performance in the bioavailability using these parameter and also these

parameters have been used previously by Vanneschi *et al.* [18]. It was decided to use a different function set $\mathcal{F}$ for each class of problem based on the performance exhibit by GP on classification problem (see [17]) and that results presented in [18] with the bioavailability problem.

The function set $\mathcal{F}$ used in the classification problems is formed by arithmetic functions, transcendental functions and max, min, if and argmax. These four later functions are implemented using arithmetic operators and exp as can be seen in Equations (1)-(4). The if function is a sort of conditional function that selects y or z depending on whether the value of x is 0 or 1, respectively. The argmax returns the index of the subtree that has the highest value.

$$\max(x, y) = \frac{x - y}{1 + e^{-100(x-y)}} + y \tag{1}$$

$$\min(x, y) = \frac{y - x}{1 + e^{-100(x-y)}} + x \tag{2}$$

$$\text{if}(x, y, z) = \frac{y - z}{1 + e^{-100x}} + z \tag{3}$$

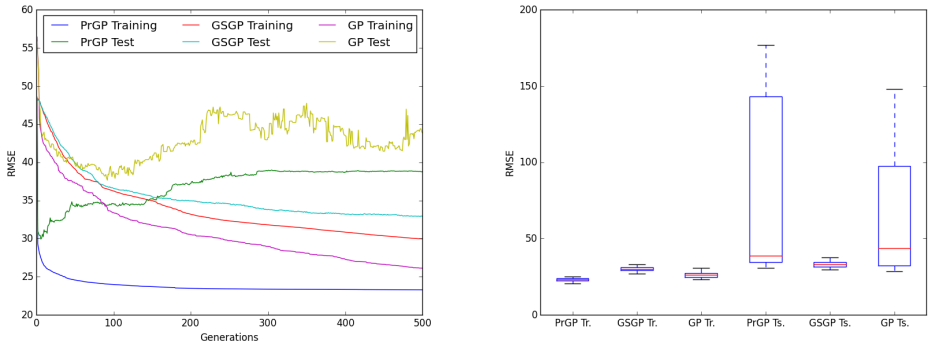
$$\text{argmax}(x) = \sum_i \frac{e^{\beta x_i}}{\sum_j e^{\beta x_j}} i \tag{4}$$

Regarding the length of the individuals it is important to note that PrPG and GSGP do not have a maximum length. The maximum length impose to GP is 1024 for the bioavailability problem and for the classification problems it was used the function suggested in [17]. The maximum length was set as $\min(\frac{|\mathbb{T}|}{2}, 256)$. This value was inspired by the degrees of freedom in a function whose parameters can be linearly identified. That is, in order to identify k parameters it is needed at least $k + 1$ points. Roughly, in a expression with n nodes at least $\frac{n}{2}$ of these nodes are operands and the other half are variables, so assuming that each variable has a coefficient to be identified one needs at least $\frac{n}{2}$ examples.

The last consideration is the procedure used to classify. Each classification problem was treated as a symbolic regression. In order to obtain a label from a continous value, the output of the individual was rounded and the output was limited to be in the range $[0, 1]$ given that all the datasets have only two classes. In addition to this, following the ideas presented on [17], an ensemble of k classifiers is used. That is, each system is initialized k times each of them is executed with different seeds, and for each of them the best individual is kept in a set. Then, the best individuals are used to predict the test set. The class of each object corresponds to the one that receives the major number of votes. Finnaly, the number of examples are balanced to have exactly the same instances for each class. This was performed removing the necessary examples in the training set until all the classes have the same number. In addition to this, the features are normalized to have zero mean and one standard deviation.

4 Results

Figure 1 presents the performance in terms of the root mean square error (RMSE) of the different systems, namely PrGP, GSGP, and GP on the bioavailability problem. This problem contains 359 instances from these instances 30 different training and test set were created using the following procedure. The training set was created by selecting 252 instances and the rest of the instances composed the test set. This process was repeated 30 times.



(a) Generations vs RMSE (b) Boxplot of the RMSE at generation

Fig. 1. Performance in terms of Root Mean Square Error (RMSE) in Human oral bioavailability problem on the training set (Tr.) and test set (Ts.). The boxplot presents the performance at generation 500.

Figure 1 (a) depicts the median of the performance (RMSE) through the generations of the different systems. In the training set it is observed that PrGP obtained the best performance reaching it at around generation 200. A particular characteristic of PrGP is that it learns faster than the other algorithms. GP obtained the second best performance and GSGP was the system with worst performance. The story in the test set is different, there GSGP is the system that got the best performance at the end of the run. Nonetheless, PrGP had the best performance before generation 150. This behaviour is an indication that PrGP presents overfitting, consequently, a procedure to prevent it is necessary.

Figure 1 (b) presents boxplots of the performance in the training and test set for the different systems. From left to right, the first three boxplot presents the performance in the training set and the last three correspond to the test set. The boxplots are created with the performance at generation 500. It is observed that PrGP had the best performance in the training set; and GSGP had the best performance in the test set. It depicts that PrGP and GP are unstable in the test set and some actions must be taken in order to prevent this behaviour.

It is important to note that GSGP in this problem does not present overfitting. This features has been noted by Vanneschi *et al.* [18] and recently by Goncalver *et al* [3]. The source of this feature is the modification performed by Vanneschi *et al.* [18] to original formulation of the geometric semantic mutation.

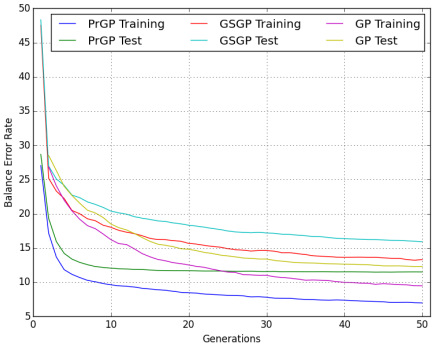
The classification problems consist in 13 different datasets all of these problems have two classes. The performance in all case is the balance error rate (BER). It was decided to split the datasets in those that contained less that 20 features and those with 20 or more features. Figure 2 presents the median performance through the generations in the training and test set. In all the cases it is observed that PrGP obtained the best performance in the training set and in addition to this its convergence rate is higher than the other systems. In the test set, it can be seen that PrGP obtained the best performance in the first generations and then it starts presenting overfitting. The only dataset where PrGP was the second best in the first generations is breast cancer.

Figure 3 presents the performance of the systems in the rest of the classification problems. A similar behaviour is presented on the training set, the exception is in the twonorm dataset, there GP had the best performance in the training set at the end of the run. Nonetheless, PrGP had a higher convergence rate reaching its best performance in the first generations. In the test set PrGP presented overfitting in almost all the datasets the exception is the image dataset. Furthermore, in image, waveform and splice is where PrGP outperforms the other systems in the test set. These dataset are the ones that have the largest number of dimensions and instances in the training set.

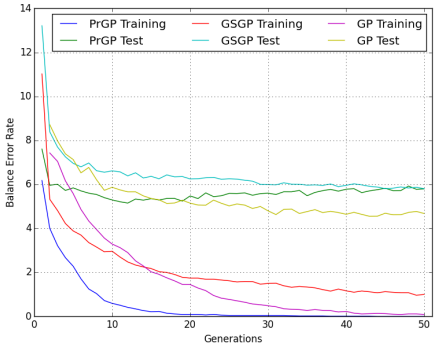
In order to complement the information presented on Figures 2 and 3, Table 3 presents the average performance and its standard deviation of PrGP, GSGP and GP on the training set. From the table, it can be seen that PrGP had the best performance in all the datasets. This is an indication that PrXO and PrMut increase the learning capabilities of genetic programming.

Table 3. Performance in terms of the balance error rate on the training set. The best performance is in boldface

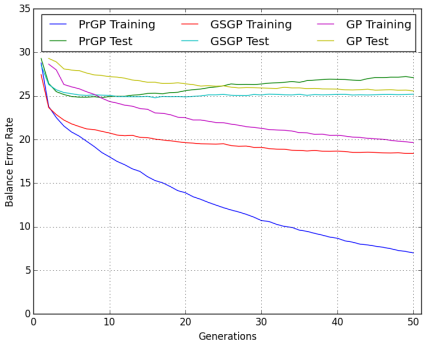
Dataset	PrGP	GSGP	GP
Banana	$\mathbf{6.98 \pm 1.44}$	13.35 ± 2.41	9.47 ± 1.53
Titanic	$\mathbf{26.04 \pm 3.85}$	26.06 ± 3.77	26.17 ± 4.26
Thyroid	$\mathbf{0.00 \pm 0.00}$	1.03 ± 0.90	0.09 ± 0.32
Diabetes	$\mathbf{6.99 \pm 1.30}$	18.37 ± 1.53	19.63 ± 1.75
Breast-Cancer	$\mathbf{4.38 \pm 1.46}$	16.02 ± 2.54	20.52 ± 3.14
Flare-Solar	$\mathbf{29.05 \pm 1.09}$	30.47 ± 1.06	30.31 ± 1.03
Heart	$\mathbf{0.44 \pm 0.49}$	7.30 ± 1.68	8.20 ± 1.95
Ringnorm	$\mathbf{0.18 \pm 0.24}$	2.96 ± 0.83	0.94 ± 0.55
Twonorm	$\mathbf{0.11 \pm 0.17}$	2.25 ± 0.67	0.26 ± 0.26
German	$\mathbf{10.64 \pm 1.20}$	21.51 ± 1.64	25.80 ± 1.71
Image	$\mathbf{4.12 \pm 0.41}$	9.27 ± 0.88	6.64 ± 1.23
Waveform	$\mathbf{0.57 \pm 0.41}$	8.77 ± 1.51	5.66 ± 1.46
Splice	$\mathbf{2.78 \pm 0.52}$	7.95 ± 0.77	9.63 ± 1.46



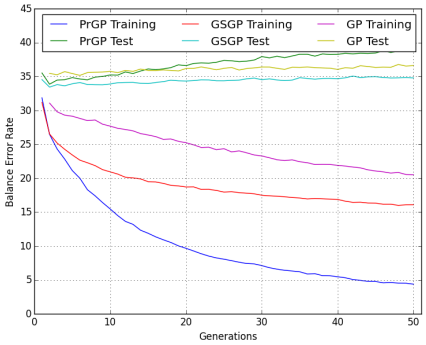
(a) Banana



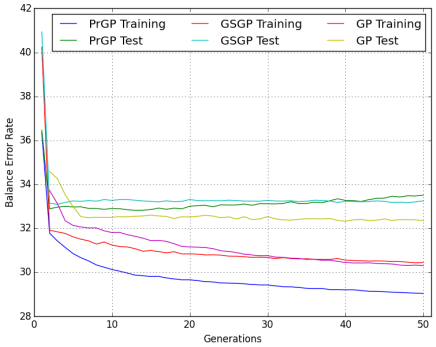
(b) Thyroid



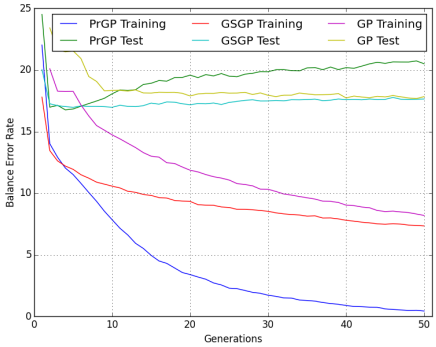
(c) Diabetis



(d) Breast Cancer



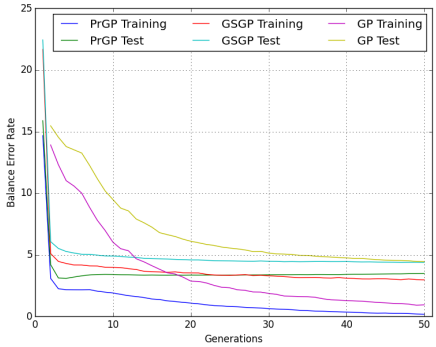
(e) Flare Solar



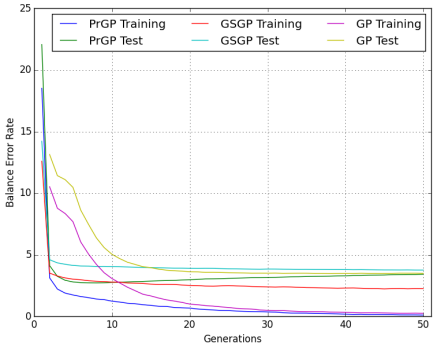
(f) Heart

Fig. 2. Performance in terms of the balance error rate

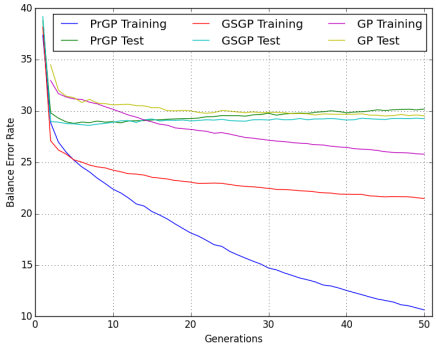
So far, we have only compared the performance of PrGP against other genetic



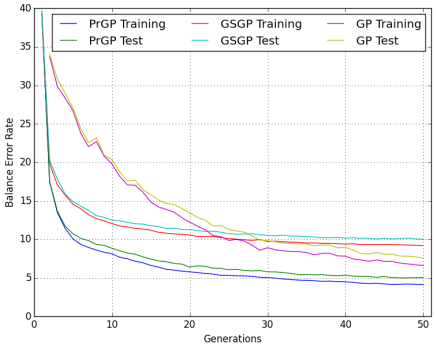
(a) Ringnorm



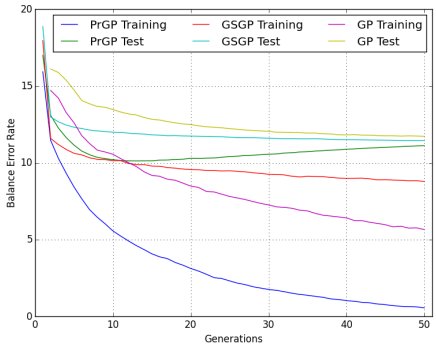
(b) Twonorm



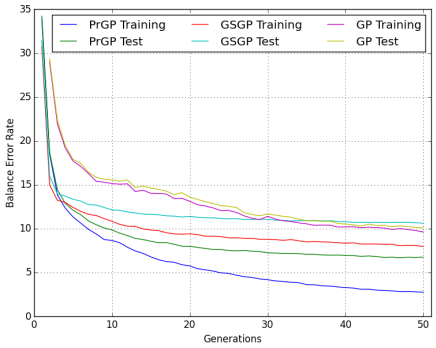
(c) German



(d) Image



(e) Waveform



(f) Splice

Fig. 3. Performance in terms of the balance error rate

programming systems. In the case of classification, it was decided to compare them against state of the art classifiers. The first classifier is an optimized version of support vector machine (SVM) [14] and the second is the Particle Swarm Model Selection (PSMS) [2]. SVM is one of the most used methods in pattern classification due to its proved effectiveness. PSMS uses particle swarm optimization to search for a classification model that maximizes an estimate of classification performance; where a classification model can also include preprocessing, and feature selection techniques. PSMS explores the space of all models that can be built by using a wide variety of methods and returns a very effective classification model. It is important to note that the performance of SVM and PSMS were taken from [14].

Table 4 presents the performance in the test set of PrGP, GSGP, GP, SVM and PSMS. It is observed that SVM had the best performance in four datasets, in second place there is a tie between PrGP and PSMS both having the best performance in three datasets. GP had the best performance in two datasets and GSGP had it in one dataset. Looking at the performance in the datasets with 20 dimensions or more, PrGP had the best performance in three out of 6. In Ringnorm dataset, PrGP presented overfitting it would be more competitive if a stoping criteria would have been used such as kfold-validation. On the other hand, PrGP presented on Image dataset underfitting, this can be corroborated on Figure 3 (f), there it is observed that the performance in the training set is closely followed by the performance on the test set and both are still decreasing at the time the evolution stop.

Table 4. Performance (BER) in the test set of the different genetic programming systems (PrGP, GSGP, and GP) and two state of the art classifiers (SVM and PSMS). The best performance is in boldface.

Dataset	PrGP	GSGP	GP	SVM	PSMS
Banana	11.50 ± 0.53	15.96 ± 1.99	12.30 ± 0.95	46.11 ± 3.64	$\mathbf{10.81 \pm 0.64}$
Titanic	30.76 ± 2.44	30.56 ± 2.62	30.44 ± 2.12	$\mathbf{22.51 \pm 0.16}$	22.81 ± 1.10
Thyroid	5.80 ± 3.13	5.78 ± 3.23	$\mathbf{4.68 \pm 2.54}$	11.60 ± 3.61	4.80 ± 2.82
Diabetes	27.11 ± 2.41	25.21 ± 2.34	25.57 ± 1.95	$\mathbf{23.17 \pm 1.69}$	27.73 ± 1.95
Breast-Cancer	38.78 ± 4.74	34.90 ± 5.04	36.61 ± 5.23	$\mathbf{29.87 \pm 3.77}$	31.95 ± 3.93
Flare-Solar	33.52 ± 1.73	33.27 ± 1.64	$\mathbf{32.35 \pm 1.65}$	32.73 ± 1.63	32.80 ± 1.50
Heart	20.50 ± 3.53	$\mathbf{17.69 \pm 3.47}$	17.84 ± 4.07	17.90 ± 2.85	24.90 ± 10.73
Ringnorm	3.49 ± 0.31	4.39 ± 0.43	4.45 ± 0.67	24.75 ± 0.51	$\mathbf{2.37 \pm 2.20}$
Twonorm	$\mathbf{3.43} \pm 0.26$	3.77 ± 0.40	3.51 ± 0.44	3.57 ± 0.59	7.82 ± 14.88
German	30.21 ± 2.67	29.25 ± 2.51	29.54 ± 2.44	$\mathbf{23.60 \pm 2.22}$	25.80 ± 3.98
Image	5.03 ± 0.61	10.05 ± 0.82	7.57 ± 1.21	15.37 ± 1.01	$\mathbf{3.90 \pm 0.83}$
Waveform	$\mathbf{11.13} \pm 0.60$	11.44 ± 0.67	11.73 ± 0.81	13.45 ± 0.63	12.08 ± 1.23
Splice	$\mathbf{6.78 \pm 0.42}$	10.64 ± 0.65	10.10 ± 1.70	16.37 ± 0.85	12.78 ± 1.92

5 Conclusions

In this contribution we have presented a novel semantic genetic operators, namely PrXO and PrMut. These operators are based on projections in the phenotype space and have as their more prominent characteristic that the fitness of the offspring is at least as good as the fitness of the best parent. These operators have been tested in two classes of problems: symbolic regression and classification problems. The results show that PrGP (GP using PrXO and PrMut) had the best performance in the training set in all the cases tested. Furthermore, it is the system that presented the highest convergence rate which makes possible to reduce the number of generations and still obtained the same performance.

Regarding the generalization ability of PrGP, it was noted that it is needed a procedure to identify the time when PrGP start to overfit. Such a procedure could be to incorporate a kfold validation, or any other appropiate technique. We leave this research avenue for future work. Nonetheless, it is observed that PrGP had the best performance in three out of six of the classification problems that have at least 20 dimensions. This might be a niche of opportunity for PrGP; however, more research is needed to validate this assertion.

References

1. Blickle, T., Thiele, L.: Genetic programming and redundancy. choice 1000, 2 (1994)
2. Escalante, H.J., Montes, M., Sucar, L.E.: Particle swarm model selection. The Journal of Machine Learning Research 10, 405–440 (2009)
3. Goncalves, I., Silva, S., Fonseca, C.M.: On the generalization ability of geometric semantic genetic programming. In: Machado, P., Heywood, M.I., McDermott, J., Castelli, M., Garcia-Sanchez, P., Burelli, P., Risi, S., Sim, K. (eds.) 18th European Conference on Genetic Programming. LNCS, vol. 9025, pp. 41–52. Springer, Copenhagen (8-10 Apr 2015), forthcoming
4. Graff, M., Flores, J.J., Bejar, J.O.: Genetic Programming: Semantic point mutation operator based on the partial derivative error. In: 2014 IEEE International Autumn Meeting on Power, Electronics and Computing (ROPEC). pp. 1–6 (Nov 2014)
5. Graff, M., Graff-Guerrero, A., Cerda-Jacobo, J.: Semantic crossover based on the partial derivative error. In: Nicolau, M., Krawiec, K., Heywood, M.I., Castelli, M., Garci-Sanchez, P., Merelo, J.J., Santos, V.M.R., Sim, K. (eds.) 17th European Conference on Genetic Programming. LNCS, vol. 8599, pp. 37–47. Springer, Granada, Spain (23-25 Apr 2014)
6. Koza, J.R.: Genetic Programming: On the Programming of Computers by Natural Selection. MIT Press, Cambridge, MA, USA (1992)
7. Moraglio, A., Krawiec, K., Johnson, C.G.: Geometric semantic genetic programming. In: Coello, C.A.C., Cutello, V., Deb, K., Forrest, S., Nicosia, G., Pavone, M. (eds.) Parallel Problem Solving from Nature - PPSN XII, pp. 21–31. No. 7491 in Lecture Notes in Computer Science, Springer Berlin Heidelberg (Jan 2012)
8. Nguyen, Q.U., Nguyen, X.H., O'Neill, M.: Semantic aware crossover for genetic programming: The case for real-valued function regression. In: Vanneschi, L., Gustafson, S., Moraglio, A., Falco, I.D., Ebner, M. (eds.) Genetic Programming,

pp. 292–302. No. 5481 in Lecture Notes in Computer Science, Springer Berlin Heidelberg (Jan 2009)
9. Pawlak, T., Wieloch, B., Krawiec, K.: Semantic Backpropagation for Designing Search Operators in Genetic Programming. IEEE Transactions on Evolutionary Computation Early Access Online (2014)
10. Poli, R., Graff, M., McPhee, N.F.: Free lunches for function and program induction. In: Proceedings of the tenth ACM SIGEVO workshop on Foundations of genetic algorithms. pp. 183–194. FOGA '09, ACM, Orlando, Florida, USA (2009), 00011 ACM ID: 1527148
11. Poli, R., Langdon, W.B., McPhee, N.F.: A Field Guide to Genetic Programming. Lulu Enterprises, UK Ltd (Mar 2008)
12. Poli, R., Langdon, W.B., McPhee, N.F.: A field guide to genetic programming. Published via `http://lulu.com` and freely available at `http://www.gp-field-guide.org.uk` (2008), http://www.gp-field-guide.org.uk, (With contributions by J. R. Koza)
13. Rojas, R.: Neutral Networks: A Systematic Introduction. Springer (1996)
14. Rosales-Pérez, A., Escalante, H.J., Gonzalez, J.A., García, C.A.R.: Bias and variance optimization for SVMs model selection. In: FLAIRS Conference. pp. 136–141. Association for the Advancement of Artificial Intelligence, Florida, USA (2013), http://www.aaai.org/ocs/index.php/FLAIRS/FLAIRS13/paper/download/5890/6055
15. Ruberto, S., Vanneschi, L., Castelli, M., Silva, S.: ESAGP – A semantic GP framework based on alignment in the error space. In: Nicolau, M., Krawiec, K., Heywood, M.I., Castelli, M., Garci-Sanchez, P., Merelo, J.J., Santos, V.M.R., Sim, K. (eds.) 17th European Conference on Genetic Programming. LNCS, vol. 8599, pp. 150–161. Springer, Granada, Spain (23-25 Apr 2014)
16. Uy, N.Q., Hoai, N.X., O'Neill, M., McKay, R.I., Galvan-Lopez, E.: Semantically-based crossover in genetic programming: application to real-valued symbolic regression. Genetic Programming and Evolvable Machines 12(2), 91–119 (Jul 2010), http://www.springerlink.com/content/48411662081364h6/
17. Valencia-Ramirez, J.M., Raya, J.A., Cedeno, J.R., Suarez, R.R., Escalante, H.J., Graff, M.: Comparison between Genetic Programming and full model selection on classification problems. In: 2014 IEEE International Autumn Meeting on Power, Electronics and Computing (ROPEC). pp. 1–6 (Nov 2014)
18. Vanneschi, L., Castelli, M., Manzoni, L., Silva, S.: A new implementation of geometric semantic GP and its application to problems in pharmacokinetics. In: Krawiec, K., Moraglio, A., Hu, T., Etaner-Uyar, A.i., Hu, B. (eds.) Genetic Programming, pp. 205–216. No. 7831 in Lecture Notes in Computer Science, Springer Berlin Heidelberg (Jan 2013)
19. Vanneschi, L., Castelli, M., Silva, S.: A survey of semantic methods in genetic programming. Genetic Programming and Evolvable Machines 15(2), 195–214 (Jun 2014), http://link.springer.com/article/10.1007/s10710-013-9210-0

Semantic Crossover Operator for GP based on the Second Partial Derivative of the Error Function

Ranyart R. Suárez[1,2], Mario Graff[2], Juan J. Flores[1]

[1] Division de Estudios de Posgrado,
Facultad de Ingenieria Eléctrica,
Universidad Michoacana de San Nicolás de Hidalgo, Mexico

[2] INFOTEC - Centro de Investigación e Innovación en Tecnologías de la Información y Comunicación,
Cátedras CONACyT,

ranyart@dep.fie.umich.mx, juanf@umich.mx, mario.graff@infotec.com.mx

Abstract. In recent years, a variety of semantic operators have been successfully developed to improve the performance of GP. This work presents a new semantic operator based on the *semantic crossover based on the partial derivative error*. The operator presented here uses the information of the second partial derivative to choose a crossover point in the second parent. The results show an improvement with respect to previous semantic operator.

Keywords: genetic programming, semantic, partial derivative, back-propagation, symbolic regression.

1 Introduction

In the community there has been an increasing interest in the use of semantic operators in Genetic Programming (GP) because these kind of operators use the information provided by the behavior of individuals to produce offprings, i.e. these use the semantics (phenotype) of individuals. On the other hand, traditional genetic operators work by manipulating the syntactic representations (i.e., genotype) of the individuals assuring that the offspring generated from the parents will be syntactically different from their them. Meanwhile, semantic operators assure that the offspring generated is semantically different from them.

From all the approaches proposed in the literature, we can distinguish two classes. The first class is called *geometric operators* (or semi-geometric), these operators are called geometric because the crossover operator, under a metric $d : \mathbb{R}$ two parents p_1 and p_2 produce offspring that lie in the d-segment between the parents. Geometric operators have certain characteristics, for example, they assure that the offspring cannot be less fitted than the less fitted of the parents (*i.e.*, they can not be worse than the worse of the parents). Examples of this

class of operators are described in [4,5,7]. A detailed survey and review of this class of operators can be found in [11,8].

The other class of operators is more general and is simply called semantic operators because the operations to generate new individuals are driven by certain rules or conditions constrained to the semantics of the generated individuals. For example, in [1,2] the offspring is incorporated to the next generation only if it is semantically different from its parents (forcing to have different semantics among the population). These approaches were later extended to other domains [6,10].

This work is based in a previous semantic crossover operator presented by Graff *et al.* [3]. The main idea of their proposal is to compute the partial derivative of the fitness function with respect to the node selected as the crossing point in the first parent, and, with this information, chose the second crossing point in the second parent. This can be accomplished by performing the backpropagation algorithm (see [9]) in GP. The backpropagation algorithm is typically used for training Artificial Neural Networks (ANNs), but its application to GP is straightforward because GP and ANNs have some similarities. In GP, the individuals are represented as trees; the output of the nodes are fed to other nodes, just like the nodes of an ANN.

In [3] the first partial derivative of fitness function *w.r.t.* the crossing point in the first parent is computed in order to know whether the output of the subtree at that particular crossing point has to increase or decrease. With this information, a search is performed in all the subtrees of the second parent; the subtree with the closest output *w.r.t.* the first derivative is selected. For the rest of this paper this methodology will be called GPPDE (Genetic Programming with Partial Derivative Error).

The work presented in this paper is an extension of GPPDE. We propose to compute the second partial derivative of the fitness function *w.r.t.* the crossing point. The idea is that the information of the second derivative complement the information given by the first derivative, consequently, the performance obtained by a system using both derivatives will outperform the results achieved by GPPDE.

The rest of the paper is organized as follows: Section 2 explains how to compute the first and second partial derivatives in GP. Section 3 explains how to interpret the information of partial derivatives in the construction of a crossover operator. Section 4 presents the comparison of different crossover operators, and Section 5 presents the conclusions.

2 Partial Derivatives in GP

GPPDE chooses two parents and one crossing point in one of the parents. Let us assume that this crossing point in the first parent is node v, and the error function is E. The semantic operator requieres to compute $\frac{\partial E}{\partial v}$, which is the partial derivative of the error function with respect to the node selected as crossing point. Using backpropagation, the error can be propagated (using a

supervised learning approach, where target values are known), until it reaches the desired node (v).

The advantage of computing the derivative of the error function is that the first derivative gives information about the error surface. For example, if $\frac{\partial E}{\partial v}$ has positive sign, it means that output in node v is greater than the value needed to minimize the error function, E. With this information, the crossing point in the second father is selected by comparing the sign of $\frac{\partial E}{\partial v}$ with the output of all subtrees in the parent, choosing the subtree that has similar sign output.

The main idea of this work is similar to GPPDE, that is, to propose a semantic crossover method for GP, based on the derivative of the error function. The semantic crossover method presented in this work uses the information provided by the second derivative of the error function to select the crossing point in the second parent (GPPDE uses the first derivative). We have called this method GPPDE2 (Genetic Programming with 2nd. order Partial Derivative error).

There are some conditions that need to be satisfied in order to compute the second derivative: 1) the error function has to be derivable and the second derivative of this function must exist and 2) the functions contained in the function set have to be derivable. The error function used in this work is the quadratic error (Equation (1)), where y is the desired output of the program and $\hat{y}$ is the current output. Note that E has first and second order derivatives with respect to the output of the program these can be seen in Equations (2) and (3).

$$E = (y - \hat{y})^2 \tag{1}$$

$$\frac{\partial E}{\partial \hat{y}} = -2(y - \hat{y}) \tag{2}$$

$$\frac{\partial^2 E}{\partial \hat{y}^2} = 2 \tag{3}$$

Before describing how the information of the first and second derivatives of E is used, we need to describe how the derivatives are computed in GP using the backpropagation algorithm.

2.1 Computing Partial Derivatives with the Backpropagation Algorithm

Backpropagation algorithm consists of two steps: 1) the forward step and 2) the backpropagation step. Consider the small program shown in Figure 1, and let us compute the first partial derivative of the programs' output with respect to node x, i.e., $\frac{\partial g(f(x))}{\partial x}$. To do so, we need to compute the output of each node and the first derivative of the function contained in the node as well; this is the forward step. Figure 2 shows the information stored in each node in the program.

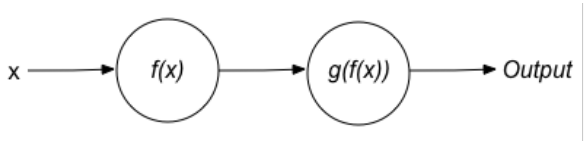


Fig. 1. GP program

Mathematically, the first and second derivatives of $g(f(x))$ with respect to x are: $\frac{\partial g(f(x))}{\partial x} = g'(f(x))f'(x)$ and $\frac{\partial^2 g(f(x))}{\partial x^2} = g''(f(x))f'(x)^2 + g'(f(x))f''(x)$. Note that all the terms needed to compute the first and second derivatives are stored in the nodes after applying the forward step. The backpropagation step consists in traversing the program backwards from the root to the selected node (in this case node x) and computing the product between the information stored in the nodes.

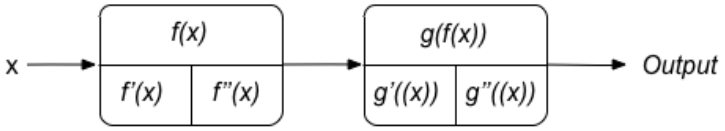


Fig. 2. Information stored in each node

Figure 3 shows how the backpropagation step is performed in order to compute the terms $\frac{\partial g(f(x))}{\partial x}$ and $\frac{\partial^2 g(f(x))}{\partial x^2}$. In Figure 3 (a), the backpropagation step is shown to compute the first derivative. The backpropagation step computes the products of the derivatives stored in the nodes; in this example, the products between input 1 and $g'(f(x))$, and $g'(f(x))$ and $f'(x)$ are computed. The result is $g'(f(x))f'(x)$, which is indeed the term $\frac{\partial g(f(x))}{\partial x}$. This is basically the process used by GPPDE to compute the first derivative of a function with respect to some node.

In this work we extend the procedure mentioned above to compute the second derivative. Whereas backpropagation in GPPDE *sees* one node at a time, GPPDE2 requires to see two nodes at a time (in this example, nodes f and g). With the information stored in these two nodes, two products are computed: 1) the product between the second derivative of the function in the first node and the square first derivative of the function in the second node, and, 2) the product between the first derivative of the function in the first node between the second derivative of the function in the second node. Figure 3 b) depicts the backpropagation step used by GPPDE2.

The following example allows to depict more clearly the process used to compute the second derivative. Consider the program shown in Figure 4, this program corresponds to $\hat{y} = 1.5x^2 - 0.7x + 1.2$. Suppose the program is chosen as

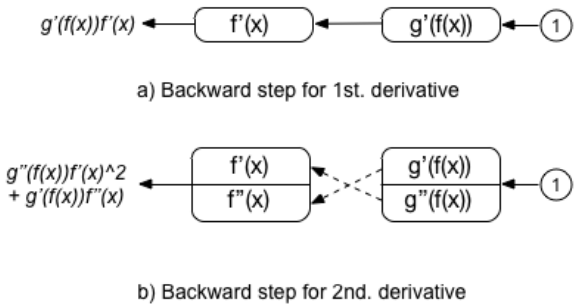


Fig. 3. Backpropagation step to compute the first and second derivatives

the first parent in a crossover operation and the crossing point is node v, i.e., the second constant -0.7. Let us assume that the inputs are $x = [-1, -0.5, 0, 0.5, 1]$, the backpropagation, in the forward step, computes the output of the program, i.e., $\hat{y}(x) = [3.4, 1.925, 1.2, 1.225, 2]$, and, also, stores the first and second derivatives of each function as in Figure 2. In this example the training set is $T = [\{-1, 1.9\}, \{-0.5, 1.175\}, \{0, 1.2\}, \{0.5, 1.975\}, \{1, 3.5\}]$, therefore the error is $e = (y(x) - \hat{y}(x)) = [-1.5, -0.75, 0, 0.75, 1.5]$.

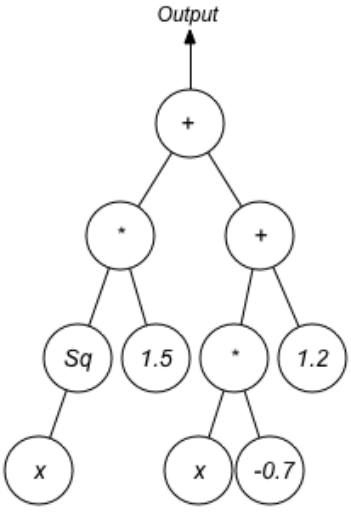


Fig. 4. Simple GP program

In the backward step the first elements that are needed are the first and second order derivatives of the fitness function. In this example, the fitness

function used is $E = (y(x) - \hat{y}(x))^2$, the first and second order derivative of E $w.r.t.$ the program's output are $\frac{\partial E}{\partial \hat{y}(x)} = -2(y(x) - \hat{y}(x)) = -2e$ and $\frac{\partial^2 E}{\partial \hat{y}(x)^2} = 2$, respectively. Figure 5 shows backward step used in GPPDE2. Note that in the figure, the root node is the function E, which takes two arguments y and $\hat{y}(x)$. Additionally, the target $y(x)$ and the part $1.5x^2$ of $\hat{y}(x)$ are simplified and the nodes involved in the process have been labeled $(E, 0, 1, 2)$.

The steps depicted in Figure 5 are the following:

(a) Node E correspond to the fitness function and node 0 is the root of the program. Following the example in Figure 3, the function in node E is g and the function in node 0 is f. GPPDE computes the term $\frac{\partial g(f(x))}{\partial x} = -2e * 1 = -2e$ and GPPDE2 computes the term $\frac{\partial^2 g(f(x))}{\partial x^2} = 2 * 1^2 + 0 * -2e = 2$, these terms are propagated to node 0.

(b) Now, node 0 corresponds to g and node 1 corresponds to f. The terms computed are: $\frac{\partial g(f(x))}{\partial x} = -2e * 1 = -2e$ and $\frac{\partial^2 g(f(x))}{\partial x^2} = 2 * 1^2 + 0 * -2e = 2$, these terms are propagated to node 1.

(c) Node 1 is g and node 2 is f function, the terms computed are: $\frac{\partial g(f(x))}{\partial x} = -2e * x = -2ex$ and $\frac{\partial^2 g(f(x))}{\partial x^2} = 2 * x^2 + 0 * -2e = 2x^2$, these terms are propagated to node 2.

d) The process stops because the propagated terms have finally reached the parent node of v.

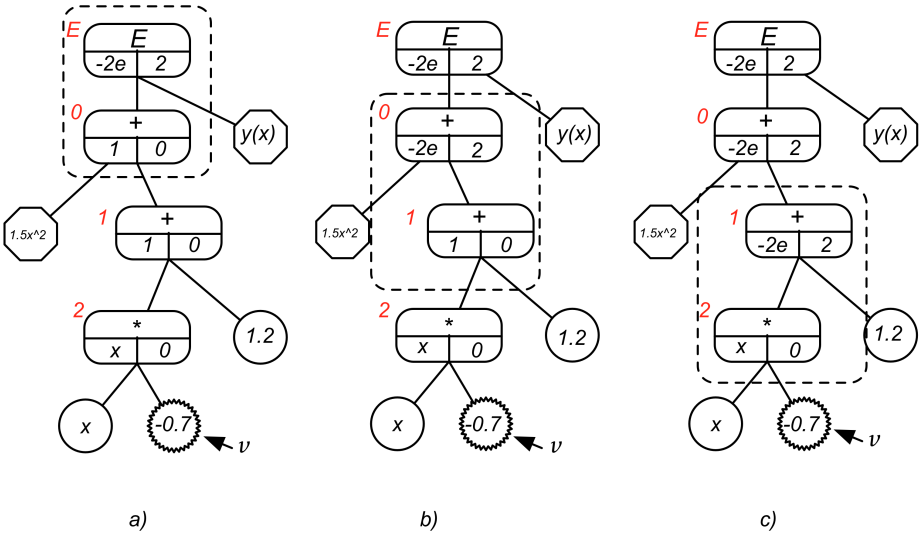


Fig. 5. Computing of partial second derivative of E function w.r.t. node v

When node v is reached, the propagated terms are: $-2ex$ and $2x^2$. These

values are indeed the first and second partial derivatives of $E = (y - ax^2 - vx - c)^2$ *w.r.t.* to v. Equations 4 and 5 show the derivatives.

$$\frac{\partial E}{\partial v} = 2(y - ax^2 - vx - c)(-x) = -2ex \tag{4}$$

$$\frac{\partial^2 E}{\partial v^2} = \frac{\partial}{\partial v} 2(y - ax^2 - vx - c)(-x) = \frac{\partial}{\partial v} 2vx^2 = 2x^2 \tag{5}$$

Note that the first and second derivatives stored in each node are vectors not scalars. Besides, every time that the values are propagated, these values replace the stored values in the next node.

3 Semantic Crossover Operator Using Partial Derivatives

GPPDE2 has computed the second partial derivative of the fitness function *w.r.t.* the node selected as the crossing point. Since the fitness function is quadratic, we decided to perform the newton method at the crossing point. The newton method (Equation 6) minimizes the error contribution to function E provided by the function that receives node v as an input.

$$X^{n+1} = X^n - f'(X^n) * [f''(X^n)]^{-1} \tag{6}$$

To demonstrate how the newton method works, let us perform the first iteration in the previous example with the information:

- The samples x = [-1, -0.5, 0, 0.5, 1]
- The error $e = [-1.5, -0.75, 0, 0.75, 1.5]$
- Output of node v, $X^n = [-0.7, -0.7, -0.7, -0.7, -0.7]$
- First partial derivative $\frac{\partial E}{\partial v} = -2ex = [-3, -0.75, 0, -0.75, -3]$
- Second partial derivative $\frac{\partial^2 E}{\partial v^2} = 2x^2 = [2, 0.5, 0, 0.5, 2]$

Since the output of node v is constant, let us take the sum of the first and second partial derivatives (-7.5 and 5). The first iteration of the newton method yields $X^1 = -0.7 - \frac{-7.5}{5} = -0.7 + 1.5 = 0.8$. This iteration of the newton method can be interpreted as the desired output from node v. So, the constant -0.7 has to be changed to 0.8. In this case, the new individual would be
$\hat{y}(x) = 1.5x^2 + 0.8 + 1.2$ which is indeed the target function for the example ($y(x)$).

In practice, the crossover point can be any node (function, variable or constant). Therefore, instead of taking the sum of partial derivatives, all the operations are done point to point between vectors, and, then, with X^1, i.e. the first iteration of the newton step, the crossover operator performs a search in all the possible sub-trees of the second parent choosing the node whose output has a minimum euclidean distance with X^1. Equation 7 shows the crossover operator, where X^1 is the first iteration of newton step, X_i^j represents the i^{th} element of X^1, S^j is the output of the j^{th} sub-tree in the second parent and S_i^j is the i^{th} element of S^j.

$$\arg\min_{j} \sum_{i}^{N} \sqrt{(X_i^1 - S_i^j)^2} \tag{7}$$

4 Results

One of the main applications of GP is symbolic regression. The problem of symbolic regression consists in finding a mathematical model that best fits certain data. In this contribution, the testbed consists in 1,100 symbolic regression problems used previously to test GPPDE in [3]. Each of the problems contained in this set was built in the following way: 1) Two polynomials $W(x)$ and $Q(x)$ were generated randomly choosing their degree in the range $[2, 8]$ with random real coefficients in the range $[-10, 10]$, 2) A rational function $y(x) = \frac{W(x)}{Q(x)}$ is used to create the data. Each function $y(x)$ was sampled 21 times uniformly distributed in the range $[-1, 1]$. The goal here is use GP to create a model that recreates the rational functions by fitting the sampled data.

Given that GPPDE2 uses more information than GPPDE, it is expected that GPPDE2 exhibits better performance than GPPDE. The first derivative used in GPPDE gives information about the direction in which the error will decrease, in other words, the first derivative tell GPPDE if the output in the subtree chosen for crossover has to be smaller or bigger. On the other hand, the second derivative computed in GPPDE2, provides the step size needed to minimize the error contribution for the subtree chosen for crossover.

Both GPPDE2 and GPPDE were run 20 times with the parameters shown in Table 1. This means that each of the 1100 problems were run 20 times and 20 different performances were obtained. Then, these measures were sorted and the median was chosen to be the final measure. We decided to choose the median because for both algorithms, in some problems, there was noise.

Table 1. GP Parameters

Parameter	Value
Population Size	1000
Number of Generations	50
Function Set ($\mathcal{F}$)	$\{+, -, \times, /\}$
Terminal Set ($\mathcal{T}$)	$\mathcal{T} = \{x \in \mathcal{R}\}$
Crossover rate	90%
Mutation rate	10%
Mutation depth	random $\in [1, 5]$
Selection	Tournament of size 2

Table 2 presents the results obtained by the GP systems tested. Additionally to GPPDE2 and GPPDE, we decided to add a third column that represents traditional Genetic Programming (GP). GP was added to the comparison in

order to demonstrate the differences in performance by performing traditional crossover and the crossover with partial derivatives.

All of the three GP systems see the problem as a minimization problem. This means that a performance closer to zero is better. The performance is given by the squared error between the target and the program's output. From Table 2 we can see that GPPDE2 won in 556 of 1100 problems (50.55%) whereas GPPDE won in 544 problems (49.45%) and GP did not have the best performance in any problem. There is no important difference between the number of problems won by GPPDE2 and GPPDE; however, there exists differences in performance. The mean of medians for GPPDE2 is 0.2905 and it was a better overall result than 0.5287 of GPPDE (a fitness closer to zero is better). Moreover, the standard deviation of GPPDE2 (1.9985) was smaller than the GPPDE's measure (5.9473) indicating more consistent results.

Table 2. Results of GP Systems

	GPPDE2	GPPDE	GP
# of problems won	556	544	0
Mean of Medians	0.2905	0.5287	4.1425
Std. Deviation	1.9985	5.9473	24.6912

From these results it can be inferred that GPPDE2 obtained better results than GPPDE. Given the number of problems that each algorithm won and the small difference in these results, is reasonably to question if GPPDE2 is indeed performing better than GPPDE. To answer this, we decided to measure the difference in performance when some algorithm outperforms the other. We measure the squared difference of performance between GPPDE2 and GPPDE when one of the algorithms won, Table 3 presents this comparison. From the table, it can be seen that when GPPDE2 has a better performance than GPPDE, the margin between the two algorithms is about 1.5670, this margin is three times bigger than the inverse case (when GPPDE has better performance than GPPDE2) with 0.5108.

Table 3. Difference in performance for GPPDE2 and GPPDE

Method	Difference in performance
GPPDE2	1.5670
GPPDE	0.5108

5 Conclusions

In this work, a new semantic crossover operator called GPPDE2 has been presented. This new method is and improved version of GPPDE, a previous crossover

semantic method. The idea behind both algorithms is to compute the partial derivatives of the error function with respect to some node (the node is selected by the cross point) in the first parent. With this information, choose the second cross point in the second parent. The main difference is that GPPDE computes the first derivative and GPPDE2 computes the second derivative. Results show that the crossover operator that uses the information of the second derivative of the error function in order to choose the crossing point in the second parent gives better results. Both semantic operators gave better results than using the traditional crossover operator (both crossing points randomly chosen) for the tested problems.

The results presented in this work open the possibility of further analysis in order to explore the capabilities of GPPDE2, for example, increasing the number of functions contained in the function set and analyze if this changes improve the current performance.

References

1. Beadle, L., Johnson, C.: Semantically driven crossover in genetic programming. In: IEEE Congress on Evolutionary Computation, 2008. CEC 2008. (IEEE World Congress on Computational Intelligence). pp. 111–116 (2008)
2. Beadle, L., Johnson, C.: Semantically driven mutation in genetic programming. In: IEEE Congress on Evolutionary Computation, 2009. CEC '09. pp. 1336–1342 (2009)
3. Graff, M., Graff-Guerrero, A., Cerda-Jacobo, J.: Semantic crossover based on the partial derivative error. In: Genetic Programming, pp. 37–47. Springer (2014)
4. Krawiec, K., Lichocki, P.: Approximating geometric crossover in semantic space. In: Proceedings of the 11th Annual conference on Genetic and evolutionary computation. p. 987994. GECCO '09, ACM, New York, NY, USA (2009), http://doi.acm.org/10.1145/1569901.1570036
5. Moraglio, A., Krawiec, K., Johnson, C.G.: Geometric semantic genetic programming. In: Parallel Problem Solving from Nature-PPSN XII, pp. 21–31. Springer (2012)
6. Nguyen, Q.U., Nguyen, X.H., O?Neill, M.: Semantic aware crossover for genetic programming: the case for real-valued function regression. In: Genetic Programming, pp. 292–302. Springer (2009)
7. Pawlak, T., Wieloch, B., Krawiec, K.: Semantic Backpropagation for Designing Search Operators in Genetic Programming. IEEE Transactions on Evolutionary Computation Early Access Online (2014), 00000
8. Pawlak, T.P., Wieloch, B., Krawiec, K.: Review and comparative analysis of geometric semantic crossovers. Genetic Programming and Evolvable Machines pp. 1–36 (2014)
9. Rojas, R.: Neural Networks: A Systematic Introduction. Springer, 1 edn. (Jul 1996)
10. Uy, N.Q., Hoai, N.X., ONeill, M., McKay, R.I., Galvn-Lpez, E.: Semantically-based crossover in genetic programming: application to real-valued symbolic regression. Genetic Programming and Evolvable Machines 12(2), 91–119 (Jul 2010), http://www.springerlink.com/content/48411662081364h6/
11. Vanneschi, L., Castelli, M., Silva, S.: A survey of semantic methods in genetic programming. Genetic Programming and Evolvable Machines 15(2), 195–214 (Jun 2014), http://link.springer.com/article/10.1007/s10710-013-9210-0

Sistema de recomendación de música basado en aprendizaje semi-supervisado

J. Roberto Alvarado-García, Janet V. Hernández-García, Esaú Villatoro-Tello, Gabriela Ramírez-de-la-Rosa y Christian Sánchez-Sánchez

Departamento de Tecnologías de la Información,
División de Ciencias de la Comunicación y Diseño,
Universidad Autónoma Metropolitana Unidad Cuajimalpa, México D.F.

{2113066338,2113066463}@alumnos.cua.uam.mx
{evillatoro,gramirez,csanchez}@correo.cua.uam.mx

Resumen. Actualmente los sistemas de recomendación son cada vez más utilizados por usuario y empresas que buscan por más y mejores contenidos digitales en Internet. Idealmente, los sistemas de recomendación deben aprender los gustos y preferencias de sus usuarios con la intención de facilitarles el proceso de búsqueda. En este trabajo presentamos un método de recomendación musical que es capaz de aprender y de adaptarse a los gustos de sus usuarios sin la necesidad de tener información previa del perfil del usuario. Para la realización de nuestros experimentos utilizamos un subconjunto de datos extraído de la base de datos musical Gracenote. Los resultados obtenidos muestran que con un conjunto reducido de características es posible construir de forma efectiva un modelo de recomendación. Agregado a esto, se muestra que con pocos datos etiquetados es posible obtener resultados aceptables en el problema de recomendación de música.

Palabras clave: sistemas de recomendación, recomendación de música, selección de atributos, aprendizaje semi-supervisado, aprendizaje automático.

1. Introducción

A medida que el Internet se ha vuelto una fuente importante de información, tanto para usuarios expertos como para usuarios inexpertos, también se ha convertido en un canal importante para la distribución de contenidos digitales muy diversos, por ejemplo: música, vídeo, imágenes, etc. En particular, la búsqueda de música representa una tarea tediosa y a menudo difícil para los usuarios. Una de las razones por las que esto sucede es debido a que los usuarios buscan por contenidos que satisfagan sus gustos personales, razón principal por la cual los sistemas de recomendación se vuelven aplicaciones indispensables [8].

Las ventajas de un sistema de recomendación no solo aplican al usuario que las utiliza; las empresas dedicadas a la venta de música por Internet pueden obtener atractivas ventajas de este tipo de sistemas. Así por ejemplo, un sistema

de recomendación puede ser utilizado para ofrecer mayor diversidad de productos (*i.e.,* contenidos musicales) relacionados a los gustos de sus consumidores y en consecuencia incrementar el número de sus ventas. En este sentido, el entender las necesidades del usuario se vuelve un factor fundamental para lograr dichos objetivos [9]. Actualmente, este esquema de negocio es seguido por empresas como Netflix[1], donde el proporcionar a sus clientes contenidos apropiados a sus preferencias les ha permitido volverse un ejemplo exitoso de la utilidad de los sistemas de recomendación que aprenden a partir de los gustos de los usuarios.

Un sistema de recomendación de música (Music Recommendation System o MRS por sus siglas en Inglés) es un sistema con la capacidad de proveer a sus usuarios suscritos recomendaciones musicales tomando en cuenta los gustos y preferencias del usuario o grupo de usuarios [3,2]. Entre los enfoques normalmente empleados por los sistemas de recomendación podemos mencionar seis principales [9], los cuales son: *i)* Sistemas Basados en Contenido, los cuales se apoyan en la valoración dada por el usuario; *ii)* Filtrado Colaborativo, basado en las valoraciones que ha realizado una población y realizando un consenso de las mismas; *iii)* Filtrado Demográfico, basado en las características del usuario como su sexo, país, lenguaje o edad; *iv)* Basados en Comunidad, utilizada en las redes sociales se apoya en la valoración de las amistades del usuario para realizar recomendaciones; *v)* Basados en Conocimiento, son sistemas que utilizan la información proporcionada por el usuario para otorgar una clasificación, y *vi)* Filtrado Híbrido, que combina dos o más de los tipos de sistemas de recomendación mencionados. En la sección 2 se dan más detalles de este tipo de enfoques.

En este trabajo se propone una herramienta para la recomendación de contenidos musicales basada en atributos de contenido. El sistema propuesto aprende los gustos musicales del usuario conforme éste lo utiliza al aplicar técnicas de aprendizaje de máquina[2]. Para la realización de nuestros experimentos utilizamos instancias obtenidas de la base de datos Gracenote[3]. Los resultados obtenidos muestran que: *1)* con una pequeña cantidad de atributos es posible hacer recomendaciones pertinentes a un conjunto de usuarios, y *2)* el método propuesto converge hacia un desempeño aceptable después de pocas interacciones con el mismo.

El resto del documento se encuentra organizado de la siguiente manera, en la sección 2 se describen algunos de los enfoques previamente utilizados así como ejemplos de algunos sistemas de recomendación actualmente activos en Internet. En la sección 3 se describe la configuración del método propuesto. La sección 4 se describen los datos empleados para la realización de este trabajo, así como los objetivos y resultados obtenidos de nuestros experimentos. Finalmente en la sección 5 se mencionan nuestras principales conclusiones y algunas ideas de trabajo futuro.

[1] Empresa que distribuye contenidos visuales (`https://www.netflix.com`)
[2] También conocido como Aprendizaje Automático por algunos autores [7].
[3] `http://www.gracenote.com`

2. Trabajo relacionado

El objetivo general de esta sección es proveer al lector de los antecedentes necesarios para conocer el funcionamiento de los sistemas de recomendación de música. En primer lugar se comentan las técnicas comúnmente empleadas para proponer sistemas de recomendación, y en segundo lugar se de da una breve descripción de los sistemas de recomendación de música que actualmente se encuentran activos en Internet.

2.1. Métodos de recomendación

Como se mencionó en la sección anterior, según [9] existen seis esquemas distintos empleados en los sistemas de recomendación. Sin embargo, en el problema de recomendación musical se habla de dos enfoques principales, *i.e.*, sistemas basados en contenido y sistemas basados en filtrado colaborativo.

En los sistemas basados en contenido, los datos (*i.e.*, canciones) que han sido vistos y calificados en algún momento previo, son empleados para construir/enriquecer un perfil de usuario; en otras palabras, tratan de aprender los gustos del usuario a partir del contenido que él ha visitado. Estos seudo-perfiles son empleados posteriormente para encontrar y recomendar más contenidos. Para lograrlo, el esquema tradicional busca aquellas canciones con una similitud alta con aquellos contenidos que el usuario ha visto antes. Ejemplos de este tipo de sistemas son los descritos en [1,3]. Un elemento clave en este tipo de sistemas es la adecuada representación de los contenidos así como la existencia y veracidad de un perfil del usuario, ambos aspectos impactarán de manera directa en la efectividad de las recomendaciones hechas por el sistema.

Como alternativa al cálculo de similitudes entre los contenidos musicales y los perfiles de usuario, los sistemas basados en filtrado colaborativo miden similitudes entre perfiles de usuarios. La idea intuitiva de este tipo de técnicas es que usuarios con gustos similares pueden ser objeto de recomendaciones similares. De esta forma, usuarios con perfiles similares son vistos como un grupo (*i.e.,* un meta-perfil) de usuarios que comparten ciertas características. El objetivo principal del filtrado colaborativo es entonces proporcionar de forma masiva recomendaciones a usuarios con el mismo meta-perfil. Ejemplos de este tipo de sistemas son [2,10]. Bajo el enfoque de filtrado colaborativo, los sistemas dependen de la efectiva construcción de los meta-perfiles de usuarios además de que son propensos a hacer recomendaciones poco efectivas debido a la diversidad que puede ser producto de la naturaleza de los grupos de usuarios considerados.

2.2. Aplicaciones de recomendación

Entre los sistemas de recomendación de música más conocidos a nivel mundial esta Pandora[4]. Pandora nace de un proyecto llamado Music Genome Project, el

[4] `http://www.pandora.com/`

cual clasifica la música en base a su melodía, armonía, ritmo, instrumentación, orquestación, arreglos, letra y otros elementos, en total utilizan 450 características. Estas características son analizadas para realizar recomendaciones completamente personalizadas a cada usuario suscrito en la plataforma. La gran ventaja de Pandora es su enorme base de conocimientos mediante la cual obtiene detalles muy específicos de las canciones que contiene. Como principales desventajas están la disponibilidad restringida a unos cuantos países y el costo que implica usarla.

Una aplicación similar a Pandora es Last.fm[5], un servicio de código abierto fundado en 2002. Last.fm es una red social, una radio vía Internet y además un sistema de recomendación de música que construye perfiles y estadísticas sobre gustos musicales, basándose en los datos enviados por los usuarios registrados. Usa un esquema de filtrado colaborativo para sugerir listados de reproducción a usuarios con el mismo meta-perfil. Entre las ventajas de Last.fm están el uso de etiquetas referidas al género, humor o características del artista que permiten al usuario escuchar canciones con etiquetas que le podrían interesar. Otra ventaja es su complemento *Audioscrobbler* que puede construir un perfil de usuario con la biblioteca de música de su computadora personal. Por último Last.fm crea sus listas de acuerdo al número de reproducciones de una canción y no al número de ventas, ofreciendo una estadística más real de la música escuchada. Como desventajas encontramos que al igual que Pandora, Last.fm ofrece suscripciones bajo el pago de una cuota. Además, el esquema de filtrado colaborativo no ofrece una recomendación a nivel de usuario sino a nivel comunidad.

Otro ejemplo de aplicación es Moodagent[6]. Moodagent, a diferencia de Last.fm y Pandora, emplea como parte de sus características para representar las canciones a las emociones contenidas en las mismas. Esto en combinación con atributos perceptuales (atributos extraídos de la señal acústica) permiten a Moodagent hacer recomendaciones a sus usuarios. Una de las principales desventajas de Moodagent es que requiere de un gran conjunto de datos etiquetados para identificar adecuadamente las preferencias de sus usuarios, agregado a esto, Moodagent no implementa ninguna estrategia que le permita ajustarse al perfil de usuario conforme éste utiliza la aplicación.

El método propuesto en este trabajo retoma algunas de las ideas de los sistemas de recomendación basados en contenido. La diferencia principal del método propuesto es que éste no requiere de tener información del perfil de usuario para proponer recomendaciones. En su lugar proponemos una arquitectura que permite a un algoritmo de aprendizaje automático ajustarse a los gustos y preferencias del usuario conforme éste va utilizando la aplicación. La ventaja principal del esquema propuesto es la posibilidad de tener recomendaciones 100 % personalizadas.

[5] http://www.lastfm.es
[6] http://www.moodagent.com

3. Método propuesto

La figura 1 muestra de forma esquemática la arquitectura del sistema propuesto. Nótese que un elemento importante dentro del sistema es el *usuario*, pues es él quien a través del Módulo de Retroalimentación permitirá a los algoritmos de aprendizaje conocer sus gustos y preferencias musicales. Es importante mencionar que bajo este esquema, la primera vez que el usuario utiliza la aplicación del MRS, ésta proporcionará una selección aleatoria (y lo más diversificada posible) de contenidos musicales al usuario. El Módulo de Retroalimentación estará encargado de identificar aquellos contenidos que el usuario haya marcado como favoritos. Idealmente la retroalimentación que el usuario proporciona al MRS no debe ser hecha de forma explícita. El sistema podría configurarse para detectar cuándo el usuario escucha contenidos completos una o varias veces y de esta forma proporcionar una retroalimentación de forma indirecta. Así entonces, los objetos identificados como favoritos serán enviados al conjunto de datos etiquetados, y posteriormente tras haber pasado por el proceso de Extracción de Atributos apoyar a la Construcción del Clasificador. El resultado de estos procesos será un modelo más adecuado a las preferencias musicales del usuario. Posteriormente dicho modelo es empleado para clasificar y recomendar nuevo contenido al usuario, el cual previo a mostrarse, atraviesa por un proceso de Ranking, el cual permite identificar aquellas recomendaciones en las que se tiene mayor confianza respecto a su clasificación. Así entonces, se espera que el MRS converja a un modelo adecuado a los gustos y preferencias del usuario conforme éste utilice el sistema.

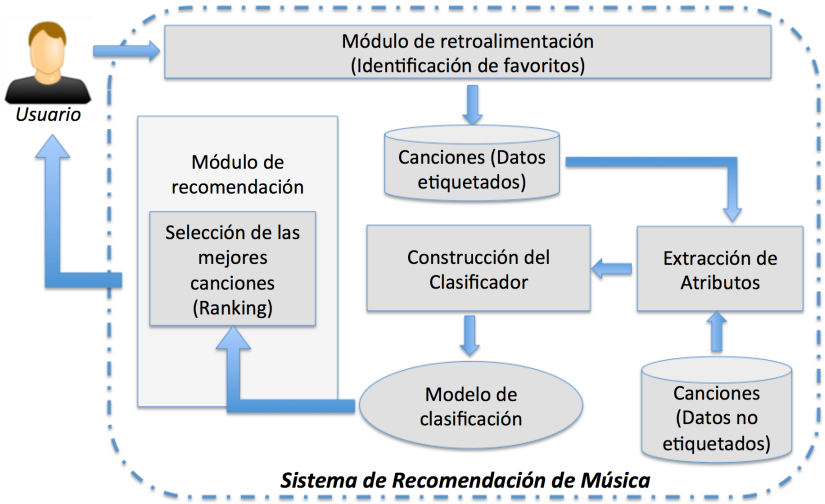


Fig. 1. Arquitectura del modelo de clasificación 100 % personalizado y re-entrenado conforme el usuario provee retroalimentación al sistema.

En resumen, abordamos el problema de recomendación de contenidos musicales como un problema de clasificación binaria, donde a cada canción se le asigna la etiqueta "recomendable" o "no-recomendable". El sistema propuesto aplica técnicas de aprendizaje semi-supervisado para ir afinando y mejorando el modelo de clasificación conforme más datos etiquetados se van obteniendo. El aprendizaje semi-supervisado aprovecha la información que es posible obtener tanto de los datos no etiquetados como de los pocos datos etiquetados disponibles. Es importante mencionar que en el aprendizaje semi-supervisado ambos conjuntos de datos (*i.e.,* no-etiquetados y etiquetados) deben provenir del mismo dominio, además de que son técnicas recomendadas cuando los datos etiquetados son escasos [11].

3.1. Conjunto de Datos

Debido a la falta de un corpus adecuado a nuestros propósitos, nos dimos a la tarea de la construcción y etiquetado de un conjunto de datos que nos permitirá evaluar la pertinencia de nuestro método propuesto. Para esto obtuvimos datos del sitio en línea Gracenote, una filial de Tribune Media Company, empresa encargada de distribución de contenido por medios digitales.

Gracenote cuenta actualmente con más de 100 millones de canciones, las cuales están etiquetadas con gran diversidad de meta-datos. En el nivel más alto de organización se identifican seis grandes categorías de estos meta-datos, los cuales son: *i)* género de la canción, *ii)* humor, *iii)* año de grabación, *iv)* velocidad de la canción, *v)* origen o región más asociada al artista y *vi)* sexo del cantante. De entre estas categorías seleccionamos ocho sub-tipos de atributos para la representación de los datos, los cuales se describen en la tabla 1:

Tabla 1. Tabla que describe los atributos seleccionados para la representación de los contenidos musicales contenidos en la base de datos de Gracenote.

Nombre	**Valores**	**Descripción**
ID canción	N/A	Identificador de canción
Título	N/A	Título de la canción
Autor	N/A	Autor de la canción
Año de lanzamiento	8 generaciones	Periodo de tiempo al que pertenece la canción
Genero	59 géneros	Divide la música en diferentes estilos de acuerdo a sus letras, instrumentos y ritmo
Lenguaje	11 idiomas y 1 sin letra	Ubica el lenguaje de las canciones
Tempo	7 velocidades	El tempo equivale a la velocidad de la canción
Estado de ánimo	25 emociones	Los tipos de emoción que tiene una canción

Es importante mencionar que los primeros tres atributos, aunque se utilizan en la interfaz gráfica de nuestro sistema, no son empleados por el proceso de aprendizaje al momento de construir el modelo de clasificación. Agregado a esto, los cinco atributos restantes, *i.e.,* año de lanzamiento, género, lenguaje, tempo y estado de ánimo, fueron utilizados como atributos *nominales* en el proceso de entrenamiento y clasificación de las instancias.

El conjunto total de datos empleados en la fase experimental fue de 120 canciones, de las cuales 60 fueron marcadas como positivas (recomendables) y 60 como negativas (no-recomendables). Para el proceso de etiquetado de los datos se tomaron en cuenta las valoraciones, hechas a mano, de dos usuarios con perfiles similares.

3.2. Algoritmos de aprendizaje

Dado que nuestra propuesta para identificar contenidos musicales recomendables para un usuario no depende de ningún algoritmo de aprendizaje en particular, podemos emplear prácticamente cualquier clasificador para enfrentar el problema. Para los experimentos realizados seleccionamos dos diferentes algoritmos de aprendizaje, los cuales son algoritmos representativos dentro de la gran variedad de algoritmos de aprendizaje disponibles actualmente en el campo de aprendizaje computacional [4,6]. Específicamente, consideramos los siguientes:

- **Naïve Bayes(NB).** Método probabilístico que asume la independencia de los atributos entre las diferentes clases del conjunto de entrenamiento.
- **J48.** Un algoritmo que permite generar un árbol de decisión, el cual selecciona los atributos más discriminativos basándose en su medida de entropía.

En nuestros experimentos se empleó la implementación de Weka [5] de cada uno de estos algoritmos empleando los parámetros por defecto. Es importante mencionar que para todos los experimentos se aplicó como estrategia de validación la técnica de validación cruzada a diez pliegues.

3.3. Evaluación

Para evaluar el método propuesto se utilizaron las medidas tradicionales para evaluación de sistemas de clasificación, tales como *precisión*, *recuerdo* y *medida* F [7]. La precisión (P) es la proporción de instancias clasificadas correctamente en una clase c_i con respecto a la cantidad de instancias clasificadas en esa misma clase. El recuerdo (R), la proporción de instancias clasificadas correctamente en una clase c_i con respecto a la cantidad de instancias que realmente pertenecen a esa clase. Así, la precisión se puede ver como una medida de la corrección del sistema, mientras que el recuerdo da una medida de cobertura o completitud.

Normalmente se emplea la medida F para describir el comportamiento de la clasificación, la cual se define como:

$$F = \frac{(1+\beta^2)P * R}{\beta^2(P+R)} \quad (1)$$

donde β representa la media armónica entre la precisión y el recuerdo. La función de β es la de controlar la importancia relativa entre las medidas de precisión y recuerdo. Es común asignar un valor de 1 indicando igual importancia a ambas medidas.

4. Experimentos y resultados

Como se ha mencionado en secciones anteriores, el objetivo principal de este trabajo fue determinar hasta qué punto es posible construir un sistema de recomendación de música, siguiendo un enfoque tradicional de clasificación automática, capaz de determinar cuándo una canción es probablemente atractiva para un usuario en particular. Para lograr dicho objetivo se propusieron dos conjuntos de experimentos, mismos que se describen a continuación:

Experimento 1. Evaluar el impacto que tiene emplear los atributos seleccionados como forma de representación de las canciones, *i.e.,* atributos nominales, en el proceso de recomendación de música. En este experimento se plantea un esquema completamente supervisado. La hipótesis principal de este experimento sugiere que con pocos atributos y muchos datos etiquetados es posible construir un modelo que permita identificar de manera eficiente los gustos de un usuario.

Experimento 2. Evaluar el comportamiento del sistema propuesto en un escenario lo más real posible, es decir, considerando la retroalimentación proporcionada por el usuario. En otras palabras, se simula un esquema de aprendizaje semi-supervisado. La hipótesis principal de este experimento sugiere que tras pocas iteraciones, el sistema de recomendación puede lograr un desempeño comparable al obtenido en el Experimento 1.

Los resultados del Experimento 1 se muestran en la tabla 2 y tabla 3. Como se mencionó antes, se evaluó el desempeño de dos algoritmos de clasificación (J48 y Naïve Bayes) en la tarea de identificar canciones recomendables para un usuario en particular. Los resultados se reportan en términos de precisión (P), recuerdo (R) y la medida F (*F-measure*). Agregado a esto, se evaluó la pertinencia de cada uno de los atributos seleccionados de forma individual. Así entonces, cada fila de las tablas 2 y 3 representa el desempeño obtenido por el clasificador respectivo utilizando sólo el atributo correspondiente. La última fila de ambas tablas (*i.e., Todos*) se refiere al desempeño del clasificados empleando los cinco atributos como forma de representación de las canciones.

Los resultados mostrados en la tabla 2 reflejan el desempeño obtenido por los algoritmos de clasificación en el problema de clasificación binaria, es decir, identificando la clase "recomendable" y "no-recomendable". Como es posible observar, ambos métodos de clasificación, *i.e.,* J48 y Naïve Bayes obtienen un desempeño

Tabla 2. Resultados obtenidos en la tarea de recomendación de música empleando un esquema de aprendizaje supervisado. Para todos los experimentos se empleó una estrategia de validación cruzada de diez pliegues.

Atributos empleados	**Algoritmos de clasificación/Medidas de Evaluación**					
	J48			**Naïve Bayes**		
	P	*R*	*F*	*P*	*R*	*F*
Genero	0.884	0.875	0.874	**0.884**	**0.875**	**0.874**
Año	0.659	0.658	0.658	0.659	0.658	0.658
Lenguaje	0.636	0.608	0.587	0.636	0.608	0.587
Edo. ánimo	0.742	0.742	0.742	0.742	0.742	0.742
Tempo	0.617	0.617	0.617	0.617	0.617	0.617
Todos	**0.884**	**0.875**	**0.874**	0.859	0.858	0.858

comparable. Esto indica, hasta cierto punto, que los atributos empleados son apropiados para la tarea de clasificación abordada y que los resultados no dependen del algoritmo de clasificación.

Nótese que emplear sólo el atributo de *Género* para entrenar los algoritmos de clasificación permite obtener muy buenos resultados de clasificación. Es igualmente valioso hacer notar que el resto de los atributos, de forma independiente, aportan información valiosa al clasificador, logrando valores de F que varían de 0.62 a 0.73. En general, de estos experimentos podemos concluir que los árboles de decisión (J48) obtienen los mejores resultados de clasificación, y al mismo tiempo, también podemos decir que solo el atributo de *género* es el más importante en el proceso de clasificación.

Tabla 3. Resultados obtenidos identificando sólo a los elementos de la clase positiva (*i.e.,* música "recomendada"). Para todos los experimentos se aplicó una técnica de validación cruzada de diez pliegues.

Atributos empleados	**Algoritmos de clasificación/Medidas de Evaluación**					
	J48			**Naïve Bayes**		
	P	*R*	*F*	*P*	*R*	*F*
Genero	0.941	0.800	0.865	0.941	0.800	0.865
Año	0.667	0.633	0.650	0.667	0.633	0.650
Lenguaje	0.575	0.833	0.680	0.575	0.833	0.680
Edo. ánimo	0.746	0.733	0.739	0.746	0.733	0.739
Tempo	0.609	0.650	0.629	0.609	0.650	0.629
Todos	**0.941**	0.800	**0.865**	0.841	**0.883**	0.862

Los resultados de la tabla 3 muestran el desempeño de los algoritmos empleados al momento de identificar solo los elementos de la clase positiva, es decir, clasificando las instancias "recomendables". El objetivo de esta tabla de resultados es mostrar qué configuración es capaz de obtener el mejor desempeño en la identificación de las canciones recomendables. La razón de hacer esto se debe a que en un escenario real, un usuario preferirá tener pocas recomendaciones pero muy precisas; en otras palabras, un sistema que tenga una baja presencia de falsos positivos. Así entonces, el método que permite tener menor tasa de falsos positivos son los árboles de decisión, pues logran superar en 10 puntos porcentuales al método de Naïve Bayes.

Con la intención de identificar la pertinencia de cada uno de los atributos empleados durante los experimentos se hizo un análisis de ganancia de información. Los resultados de este análisis se muestran en la tabla 4. Del análisis de ganancia de información se puede concluir que todos los atributos tienen información valiosa. Debido a que no hay atributos que cumplan $IG \leq 0$, concluimos que todos los atributos son pertinentes en el problema de clasificación abordado y en consecuencia es preferible conservarlos.

Tabla 4. Análisis de Ganancia de Información sobre los cinco atributos empleados para representar las canciones de nuestra base de datos.

Atributo	**IG**
Genero	0.8275
Edo. ánimo	0.5131
Año	0.1841
Lenguaje	0.1259
Tempo	0.0657

Finalmente, la figura 2 muestra los resultados obtenidos del Experimento 2. De acuerdo a la descripción que se dio de la arquitectura del sistema en la sección 3 (figura 1), el usuario juega un papel importante debido a la retroalimentación (directa o indirecta) que puede proporcionar al sistema respecto de sus gustos. Bajo esta configuración, el sistema propuesto deberá ser capaz de entrenar un modelo de clasificación para recomendación de música conforme el usuario utiliza la aplicación. Como se mencionó antes, esta configuración se asemeja a un esquema de aprendizaje semi-supervisado, es decir, se tienen pocos elementos etiquetados al principio y conforme se va utilizando el clasificador, *i.e.,* después de varias iteraciones, se incrementa la colección de datos etiquetados con los cuales el modelo de clasificación puede ser re-entrenado con la finalidad de adecuarse más al concepto que se quiere aprender, en este caso los gustos musicales del usuario.

En la figura 2 se simuló la retroalimentación que podría proporcionar un usuario al momento de interactuar con el sistema. Así entonces, en el eje horizontal se indica la cantidad de ejemplos etiquetados con los que se entrena y

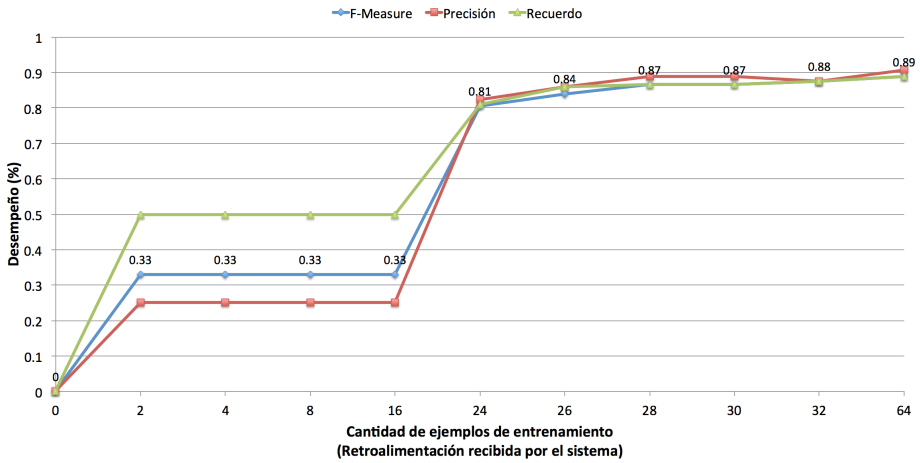


Fig. 2. Resultados simulando el proceso de retroalimentación.

genera el modelo de recomendación (*i.e.,* 2, 4, 8, 16, 24, ...). Los resultados se reportan en términos de P, R y F-*measure*. Es importante mencionar que en la figura 2 se reporta el promedio de haber hecho estos experimentos empleando cinco diferentes muestreos de los datos, esto como forma de validación de los experimentos realizados.

Los resultados obtenidos de este experimento muestran que con 2 ejemplos etiquetados el sistema logra alcanzar un desempeño de $F = 0.33$, y tan solo después de haber proporcionado retroalimentación sobre 24 ejemplos, el sistema de recomendación logra obtener ya un $F = 0.81$. Es importante hacer notar que a partir de este punto el modelo de recomendación tiende a estabilizarse, alcanzando en una $F = 0.89$ con 64 instancias etiquetadas.

En conclusión, estos experimentos reafirman la pertinencia del método propuesto, y muestran cómo el enfoque semi-supervisado es capaz de obtener resultados comparables contra un enfoque completamente supervisado.

5. Conclusiones y trabajo futuro

En este trabajo hemos presentado un método de recomendación de música que aprende y se adapta a los gustos del usuario. El método propuesto se abordó como un problema de clasificación binaria y por tanto emplea técnicas de aprendizaje automático para la construcción del modelo de recomendación.

Entre los objetivos que se plantearon al inicio del trabajo fue por un lado, evaluar la pertinencia de una serie de atributos que son fácilmente extraíbles de la base de datos musicales Gracenote. Y por otro lado, evaluar la efectividad del método bajo un enfoque semi-supervisado, es decir, con muy pocos datos etiquetados.

Los resultados obtenidos mostraron que los atributos proporcionados por la base de Gracenote son apropiados para el problema abordado. Fue interesante observar que bajo un esquema 100 % supervisado, el atributo *Genero* proporciona información muy valiosa al momento de construir el modelo de recomendación. Sin embargo una serie de experimentos adicionales, mostraron que todos los atributos empleados poseen valores altos de ganancia de información, razón por la cual se decidió conservarlos en los experimentos posteriores. Agregado a esto, dado que ambos algoritmos de aprendizaje empleados son capaces de enfrentar la ausencia de un atributos (*i.e.,* datos faltantes), si en un determinado momento una canción extraída de la base de datos carece del atributo *Genero*, el modelo será capaz de asignar una categoría si cuenta con algo de información en el resto de los atributos.

Por otro lado, un segundo bloque de experimentos mostraron que con pocos datos etiquetados es posible construir un modelo de recomendación adaptado a los gustos del usuario. Bajo un enfoque semi-supervisado se pudo observar que con apenas 2 datos etiquetados el método logra una $F = 0.33$, y con solo 24 datos el método alcanza un desempeño comparable al obtenido cuando se emplea todo el conjunto de datos etiquetados disponible (*i.e.,* $F = 0.81$). Estos resultados son alentadores, pues indican que el método propuesto es apropiado para identificar "rápidamente" los gustos de los usuarios y además de que puede adaptarse a los mismos conforme el usuario va proporcionando retroalimentación.

Como trabajo futuro inmediato se propone incluir alguna serie de atributos perceptuales, es decir, atributos extraídos de la señal acústica. Ejemplos de este tipo de atributos son la densidad del *pitch*, atributos asociados a la frecuencia y amplitud de la señal acústica, valores de entropía, etc. Nuestra intuición es que dichos atributos pueden enriquecer favorablemente el modelo de recomendación propuesto.

Agradecimientos. Agradecemos al Departamento de Tecnologías de la Información de la Universidad Autonoma Metrópolitana Unidad Cuajimalpa y al proyecto CONACYT número CB2010/153315 por el apoyo otorgado para la realización de este trabajo.

Referencias

1. Basu, C., Hirsh, H., Cohen, W.: Recommendation as classification: Using social and content-based information in recommendation. In: Proceedings of the Fifteenth National Conference on Artificial Intelligence. pp. 714–720. AAAI Press (1998)
2. Bu, J., Tan, S., Chen, C., Wang, C., Wu, H., Zhang, L., He, X.: Music recommendation by unified hypergraph: Combining social media information and music content. In: Proceedings of the International Conference on Multimedia. pp. 391–400. MM '10, ACM, New York, NY, USA (2010), http://doi.acm.org/10.1145/1873951.1874005
3. Chen, H.C., Chen, A.L.P.: A music recommendation system based on music data grouping and user interests. In: Proceedings of the Tenth International Conference

on Information and Knowledge Management. pp. 231–238. CIKM '01, ACM, New York, NY, USA (2001), http://doi.acm.org/10.1145/502585.502625
4. Duda, R., Hart, P.: Pattern classification and scene analysis. Wiley (1996), http://www.ica.luz.ve/~enava/redesn/ebooks/DHS/Versi%F3nPS/DHSChap4.ps
5. Garner, S.R.: Weka: The waikato environment for knowledge analysis. In: Proc. of the New Zealand Computer Science Research Students Conference. pp. 57–64 (1995)
6. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer (2009)
7. Mitchell, T.: Machine Learning. McGraw-Hill (1997)
8. Park, H.S., Yoo, J.O., B., C.S.: A context-aware music recomendation system using fuzzy bayesian networks with utility theory. Fuzzy Systems and Knowledge Discovery 4223, 970–979 (2006)
9. Ricci, F., Rokach, L., Shapira, B., Kantor, P.B.: Recommender Systems Handbook. Springer (2011)
10. Su, J.H., Yeh, H.H., Yu, P.S., Tseng, V.S.: Music recommendation using content and context information mining. IEEE Intelligent Systems 25(1), 16–26 (Jan 2010), http://dx.doi.org/10.1109/MIS.2010.23
11. Zhu, X., Goldberg, A.B.: Introduction to Semi-Spuervised Learning. Morgan and Claypool (2009)

Evolución Diferencial con perturbaciones Gaussianas

M. A. Sotelo-Figueroa[1], Arturo Hernández-Aguirre[2], Andrés Espinal[3] y J. A. Soria-Alcaraz[1]

[1] Universidad de Guanajuato, División de Ciencias Económico Administrativas, Departamento de Estudios Organizacionales, Guanajuato, Gto, México

[2] Centro de Investigación en Matemáticas, Departamento de Ciencias de la Computación, Guanajuato, Gto, México

[3] Tecnologico Nacional de México, Instituto Tecnológico de León, Departamento de Estudios de Posgrado e Investigación, León, Gto, México
masotelo@ugto.mx, artha@cimat.mx, andres.espinal@itleon.edu.mx, soajorgea@gmail.com

Resumen. La Evolución Diferencial es una metaheurística poblacional que ha sido ampliamente utilizada para la Optimización de problemas de Caja Negra. La muta es el operador principal de búsqueda en Evolución Diferencial, del cual hay diferentes esquemas reportados en el estado del arte; sin embargo dichos esquemas carecen de mecanismos para realizar una intensificación los cuales pueden permitir una mejor búsqueda y evitar óptimos locales. El presente artículo propone una perturbación Gaussiana con el objetivo de mejorar el desempeño de dos esquemas de muta bien conocidos y ampliamente utilizados en el estado del arte. Se realizaron pruebas en un conjunto de instancias de pruebas del CEC2013, los resultados obtenidos se compararon mediante la prueba no paramétrica de Friedman para determinar si el uso de la perturbación propuesta mejora el rendimientos de los esquemas originales.

Palabras clave: differential evolution, blackBox optimization, Gaussian perturbation.

1. Introducción

La Evolución Diferencial (ED) [21] es un algoritmo poblacional que ha sido ampliamente estudiado y básicamente se pueden dos líneas de investigación[5]: la búsqueda de nuevos esquemas de mutación con la finalidad de poder mejorar su rendimiento [2] [16] [1] [9] [12], y la otra es el estudio de los parámetros para obtener mejores resultados in [6] [19] [3] [14] [4].

Para hacer que la ED obtenga mejores resultados se ha hibridado con algunas otras técnicas como Búsqueda Local [18] u Optimización por Cumulo de Partículas [24]. También se han tratado de proponer nuevos esquemas de mutación como una Mutación Gaussiana [16]. Dichas propuestas tratan de mejorar el comportamiento de la ED sin embargo hacen que la ED se vuelva más compleja de lo que originalmente era.

Los problemas de caja negra [22] [8][11] [10] son problemas los cuales no se conoce a priori la función objetivo con la que se esta trabajando, motivo por el cual no se puede utilizar algoritmos basados en gradiente.

En el presente articulo se propone una perturbación Gaussiana la cual se puede utilizar con cualquier esquema de mutación para mejor el desempeño de la ED. Para probarla se utilizaron las 28 funciones propuestas en la sesión especial de Optimización Real de Parámetros del CEC2013[4]. Para discernir los resultados se aplico la prueba no paramétrica de Friedman.

El articulo se estructura como sigue: en la Sección 2 se presenta el algoritmo Evolución Diferencial y la perturbación Gaussiana que se propone, en la Sección 3 se explica el experimento que se realizo. Los resultados del experimento se muestran en la Sección 4. La Sección 5 muestra las conclusiones y el trabajo futuro.

2. Evolución diferencial

La Evolución Diferencial (ED) [21] fue desarrollada por R. Storm y K. Price en 1996. Es un algoritmo evolutivo el cual se basa en vectores y puede ser considerado como desarrollo adicional al Algoritmo Genético. La ED es un método de búsqueda estocástico con tendencias de auto organización sin necesidad de hacer uso la información de la derivada [23].

Como en los Algoritmos Genéticos [13], los parámetros a optimizar se representan como un vector d-dimensional y se aplican una serie de operadores sobre dichos vectores. Sin embargo, a diferencia del Algoritmo Genético, la ED aplica dicho operador sobre cada componente del vector.

Para un problema de optimización $d - dimensional$ con d parámetros, se genera una población inicial de n vectores donde se tienen x_i donde $i = 1, 2, \ldots, n$. Para cada solución x_i de cualquier generación t se puede usar la notación mostrada en la Ecuación 1 que consiste de $d - componentes$ en un espacio de búsqueda $d - dimensional$. Este vector puede ser considerado como el cromosoma o el genoma.

$$x_i^t = (x_1^t, x_2^t, \ldots, x_{d,i}^t) \tag{1}$$

La ED contiene tres pasos principales: mutación, cruza y selección. La mutación es llevada acabo por un Esquema de Mutación. Para cada vector x_i en cualquier tiempo o generación t, se generan tres números diferentes de manera aleatoria x_p, x_q y x_r en el tiempo t y se genera un vector de prueba mediante un Esquema de Mutación, como se ve en la Ecuación 2. El parámetro $F \in [0, 2]$ es denominado como peso diferencial.

$$v_i^{t+1} = x_p^t + F(x_p^t - x_r^t) \tag{2}$$

La cruza es controlada por la probabilidad $Cr \in [0, 1]$, y puede ser binomial o exponencial. El esquema de cruza binomial se aplica a cada uno de los componentes del vector mediante la generación de un número aleatorio uniformemente distribuido $r_i \in [0, 1]$ y se manipula como se muestra en la Ecuación 3 y se puede decidir en cada componente su se intercambia con el vector de prueba o no.

[4] http://www.ntu.edu.sg/home/EPNSugan/index_files/CEC2013/CEC2013.htm

$$u_{j,i}^{t+1} = \begin{cases} v_{j,i} & \text{si } r_i \leq C_r \\ x_{j,i} & \text{en otro caso} \end{cases} j = 1, 2, ..., d \tag{3}$$

En el esquema de cruza exponencial un segmento del vector de prueba es seleccionado empezando de la posición $k \in [0, d-1]$ y tamaño $L \in [1, d]$ de manera aleatoria. Este esquema se puede ver en la Ecuación 4.

$$u_{j,i}^{t+1} = \begin{cases} v_{j,i}^t \; v_{j,i}^t \text{ for } j = k, \ldots, k-L \in [1, d] \\ x_{j,i}^t \qquad\qquad \text{en otro caso} \end{cases} \tag{4}$$

La selección utilizada por la ED es en esencia la misma que se usa en el Algoritmo Genético. Dependiendo de lo que se esta haciendo se selecciona el valor más grande o el más pequeño. Dicha selección se puede ver en la Ecuación 5.

$$x_i^{t+1} = \begin{cases} u_i^{t+1} \text{ if } f(u_i^{t+1}) \leq f(x_i^t) \\ x_i^t \qquad \text{en otro caso} \end{cases} \tag{5}$$

Los tres pasos de la ED se muestran en el Algoritmo 1. Cabe señalar que el uso de J es para asegurar que $v_i^{t+1} \neq x_i^t$ lo cual incrementa la eficiencia de la exploración. La eficiencia de la ED es controlada por dos parámetros: el peso diferencial F y la probabilidad de cruza C_r.

Algorithm 1 Algoritmo de Evolución Diferencial

Require: F peso diferencial, C_r probabilidad de cruza, n tamaño de la población.

1: Iniciar la población Inicial.
2: **while** no se cumpla el criterio de paro **do**
3: **for** $i = 1$ to n **do**
4: Para cada x_i se seleccionan de manera aleatoria 3 diferentes vectores x_p, x_r y x_r.
5: Generar un nuevo vector v mediante algún esquema de mutación 2.
6: Generar aleatoriamente un Índice $J_r \in \{1, 2, \ldots, d\}$.
7: Generar aleatoriamente un número $r_i \in [0, 1]$.
8: **for** $j = 1$ a n **do**
9: Para cada parámetro $v_{j,i}$ (j-esimo componente de v_i), actualizarlo como:
10: $u_{j,i}^{t+1} = \begin{cases} v_{j,i}^{t+1} & \text{si } r_i \leq C_r \text{ o } j = J_r \\ x_{j,i}^t & \text{si } r_i > C_r \text{ o } j \neq J_r \end{cases}$
11: **end for**
12: Seleccionar y actualizar la solución usando 5.
13: **end for**
14: Actualizar el contador $t = t + 1$
15: **end while**

En [17] se pueden ver 5 estrategias para mutar el vector v:

- DE/rand/1: $V_i = X_{r1} + F(X_{r2} - X_{r3})$.
- DE/best/1: $V_i = X_{best} + F(X_{r1} - X_{r2})$.
- DE/curren to to best/1: $V_i = X_i + F(X_{best} - X_i) + F(X_{r1} - X_{r2})$.

- DE/best/2: $V_i = X_{best} + F(X_{r1} - X_{r2}) + F(X_{r3} - X_{r4})$.
- DE/rand/2: $V_i = X_{r1} + F(X_{r2} - X_{r3}) + F(X_{r4} - X_{r5})$.

donde: r_1, r_2, r_3, r_4 son números enteros aleatoriamente generados y mutuamente diferentes, también deben de ser diferentes del vector de prueba i y del índice del mejor vector X_{best}.

2.1. Perturbación gaussiana

Con la propuesta de esta Perturbación Gaussiana se busca que la ED evolucione de manera normal bajo un determinado esquema de mutación y en caso de que el vector no se pueda mejorar mediante dichos pasos entonces se aplica una perturbación a cada dimensión tratando de que el vector actual pueda salir de un mínimo local. Esto se hace si la Ecuación 5 no se aplico y se quedo con el vector original. Lo que se hace es que para elemento del vector se le aplica una perturbación Gaussiana mediante la suma de un número generado aleatoriamente mediante una distribución Normal con media cero y una varianza de uno.

La Perturbación Gaussiana esta basada en un esquema de mutación Gaussiana[16] , la cual genera un vector de prueba usando una normal con media en un individuo, como se ve en la Ecuación 6, y con una varianza basada en la diferencia de dos elementos aleatorios. Este esquema de mutación esta utilizando el esquema Rand1 para hacer la generación del nuevo individuo.

$$u_i^{t+1} \sim N\left(\mu_i, \sigma_i^2\right) \tag{6}$$

$$\mu = x_{r1}^t \tag{7}$$

$$\sigma = \mid x_{r2} - x_{r3} \mid \tag{8}$$

Algorithm 2 Perturbación Gaussiana

```
1: if No se actualizo el Individuo usando la Ecuación 5 then
2:     for j = 1 to d do
3:         r ∈ N(0, 1)
4:         u_{j,i}^{t+1} = x_{j,i}^t + r
5:     end for
6:     Seleccionar y actualizar la solución usando 5.
7: end if
```

3. Experimentos

Se utilizaron las 28 Funciones de la Tabla 1, las cuales fueron tomadas de la sesión especial de optimización de parámetros reales del CEC2013. Dichas funciones están para 2, 10, 30 y 50 dimensiones. Se ejecutaron 51 pruebas independientes para determinar

la Mediana y la Desviación Estándar de cada una de las funciones. El máximo número de Llamadas a Función esta en función de la dimensión con la que se este trabajando y es de 10000*D; así pues para 2D=20000, 10D=100000, 30D=300000 y 50D=500000. El espacio de búsqueda para cada función es de $[-100, 100]^D$, la población se debe de inicializar de manera aleatoria uniformemente distribuida y se debe de utilizar el mismo método de optimización para todas las funciones.

Los parámetros utilizados por la ED fueron $F = 0.9$ y $Cr = 0.8$. Dichos parámetros se obtuvieron mediante un proceso de optimización de parámetros basado en Covering Arrays (CA) [20], los CAs se generaron mediante el Covering Array Library (CAS) [15] del National Institute of Standards and Technology (NIST) [5].

Para comparar los resultados obtenidos mediante los diferentes esquemas de mutación de ED se utilizo la prueba no paramétrica de Friedman [7] con un nivel de significancia del 99 % o un p-Valor menor que 0.1.

Tabla 1. Funciones del CEC2013 utilizadas como instancias de prueba

	No.	Función	$f_i^* = f_i(x^*)$
Funciones Unimodales	1	Sphere Function	-1400
	2	Rotated High Conditioned Elliptic Function	-1300
	3	Rotated Bent Cigar Function	-1200
	4	Rotated Discus Function	-1100
	5	Different Powers Function	-1000
Funciones Básicas Multimodales	6	Rotated Rosenbrock's Function	-900
	7	Rotated Schaffers F7 Function	-800
	8	Rotated Ackley's Function	-700
	9	Rotated Weierstrass Function	-600
	10	Rotated Griewank's Function	-500
	11	Rastrigin's Function	-400
	12	Rotated Rastrigin's Function	-300
	13	Non-Continuous Rotated Rastrigin's Function	-200
	14	Schwefel's Function	-100
	15	Rotated Schwefel's Function	100
	16	Rotated Katsuura Function	200
	17	Lunacek Bi_Rastrigin Function	300
	18	Rotated Lunacek Bi_Rastrigin Function	400
	19	Expanded Griewank's plus Rosenbrock's Function	500
	20	Expanded Scaffer's F6 Function	600
Funciones Compuestas	21	Composition Function 1	700
	22	Composition Function 2	800
	23	Composition Function 3	900
	24	Composition Function 4	1000
	25	Composition Function 5	1100
	26	Composition Function 6	1200
	27	Composition Function 7	1300
	28	Composition Function 8	1400

[5] http://csrc.nist.gov/groups/SNS/acts/index.html

4. Resultados

En las Tablas 2, 3, 4 y 5 podemos ver los resultados obtenidos para cada una de las diferentes dimensiones probadas. Los resultados que se muestran son la Mediana y la Desviación Estándar que se obtuvieron al aplicar cada uno de los esquemas de mutación a las diferentes función de prueba.

Los resultados para 2 dimensiones, Tabla 2, se puede observar que las Medias reportadas son prácticamente los óptimos de cada una de las funciones de prueba, y que la Desviación Estándar tiende a cero, lo que implica que en cada uno de las 51 pruebas independientes que se realizaron se llego casi al mismo resultado. Para el resto de las dimensiones, los resultados obtenidos con los diferentes esquemas de mutación para cada función objetivo varían.

Los valores de la prueba no paramétrica de Friedman aplicada a cada una de las dimensiones se puede ver en la Tabla 6, en el caso de la dimensión 2 es la única que el p-valor no es menor de 0.1 con lo cual no se puede determinar estadísticamente si hay algún esquema de mutación que tenga un desempeño diferente a los demás. Las otras dimensiones tienen p-Valores menores que 0.1 por lo que se realizo un procedimiento post-hoc [7] para determinar cual de los esquemas de mutación dieron mejores resultados.

La Tabla 7 contiene los resultados del procedimiento post-hoc de las dimensiones 20, 30 y 50. Dicha tabla contiene la posición en cada dimensión que ocupa cada uno de los esquemas usados, se muestran los resultados de manera ascendente mostrando con el número más pequeño aquel esquema que obtuvo el mejor desempeño.

5. Conclusiones y trabajos futuros

En base a los resultaos obtenidos en la Sección 4 podemos concluir lo siguiente:

- La Evolución Diferencial permite realizar la optimización de problemas de caja negra usando diferentes esquemas de mutación.
- Entre los diferentes esquemas de mutación encontrados en el estado del arte y probados en el presente articulo, el esquema Best1 es el que permite obtener mejores resultados.
- La perturbación Gaussiana propuesta permite ser incorporada a los diferentes esquemas de mutación, permitiendo en un espacio de búsqueda mayor mejorar los resultados que los esquemas por si solos obtienen.

Dentro de los trabajos futuros podemos considerar los siguientes:

- Usar números aleatorios con otro tipo de distribución que nos permitan intensificar y diversificar.
- Aplicar la Evolución Diferencial con la perturbación Gaussiana a otro tipo de problemas de optimización.

Agradecimientos. Los autores quieren agradecer a la Universidad de Guanajuato (UG) y al Instituto Tecnologico de León (ITL) por el apoyo brindado para poder realizar la presente Investigación.

Tabla 2. Resultados obtenidos en 2 Dimensiones

Función	Rand1		Rand1 Perturbación		Best1		Best1 Perturbación	
	Mediana	Desviación Estándar	Mediana	Desviación Estándar	Mediana	Desviación Estándar	Mediana	Desviación Estándar
1	-1400	0	-1400	0	-1400	0	-1400	0
2	-1300	0	-1300	0	-1300	0	-1300	0
3	-1200	0	-1200	0	-1200	0	-1200	0
4	-1100	0	-1100	0	-1100	0	-1100	0
5	-1000	0	-1000	0	-1000	0	-1000	0
6	-900	0	-900	0	-900	0	-900	0
7	-800	0	-800	14.500E-8	-800	0	-800	0
8	-689.3377	7.7096	-700	6.0700E-8	-700	2.6699	-700	0
9	-599.9790	10.162E-2	-600	1.2300E-4	-600	0	-600	0
10	-500	10.2545E-4	-500	16.9942E-4	-499.9926	94.6512E-4	-499.9926	80.4591E-4
11	-400	0	-400	0	-400	19.3129E-2	-400	0
12	-300	0	-300	0	-300	13.7949E-2	-300	0
13	-200	27.5899E-2	-200	38.6259E-2	-200	59.1737E-2	-200	96.2092E-2
14	-100	7.34525E-2	-100	6.05952E-2	-99.6878	5.3373	-99.6878	17.6587E-2
15	100	9.28302E-2	100	11.9007E-2	100.3122	6.5817	100.3122	5.6430
16	200.2055	10.580E-2	200.2837	11.9015E-2	200.4341	27.6091E-2	200.1446	19.4136E-2
17	300.0100	64.8657E-2	300.0264	48.4489E-2	300.0900	1.0324	302.0225	96.2327E-2
18	400	74.5786E-2	400	65.6815E-2	400	1.0307	402.0386	1.0155
19	500	0	500	0	500	53.0422E-4	500	30.3457E-4
20	600	82.4342E-4	600.0031	82.0231E-4	600.0194	84.6878E-4	600.0031	96.6389E-4
21	700	13.8648	700	0	700	27.7297	700	0
22	800	0	800	13.200E-14	800	6.1883	800	5.8100
23	900	0	900	4.3371	900	34.3032	900	20
24	1000	0	1000	74.0571E-2	1000	14.0500	1000	3.7075
25	1100	29.7368	1100	6.3700E-14	1100	48.1511	1100	0
26	1200	1.1224E-2	1200	2.78572E-2	1200.0810	20.1335E-2	1200.0810	1.0026
27	1400	50.0206	1300.4080	54.7587E-2	1400	40.9605	1300.8487	2.8579
28	1400	0	1400	0	1400	32.2190	1400	0

Tabla 3. Resultados obtenidos en 10 Dimensiones

Función	Rand1		Rand1 Perturbación		Best1		Best1 Perturbación	
	Mediana	Desviación Estándar	Mediana	Desviación Estándar	Mediana	Desviación Estándar	Mediana	Desviación Estándar
1	-1400	0	-1400	6.7800E-8	-1400	0	-1400	0
2	416306.31	214802.26	341737.85	135754.22	-1299.2456	24.5095	-923.3450	5172.0868
3	343002.02	636163.88	2541979.10	113E5	-1199.9424	2.2167	-1193.1031	72.7863
4	984.7043	1371.7921	-555.0403	211.2600	-1099.9988	44.3797E-4	-1098.9986	1.5002
5	-1000	1.3100E-10	-999.9999	45.500E-6	-1000	0	-1000	8.5700E-14
6	-899.9925	33.0102E-4	-899.5852	22.9179E-2	-900	93.8019E-2	-899.9999	55.2727E-2
7	-778.6989	11.0512	-787.3262	4.1405	-799.9973	2.2449	-799.8426	38.8166E-2
8	-679.6476	6.26167E-2	-679.6215	7.52653E-2	-679.5992	7.76678E-2	-679.6267	7.92253E-2
9	-590.9883	62.9856E-2	-592.2454	1.8715	-590.9685	87.1897E-2	-598.9312	1.8933
10	-499.3358	8.52432E-2	-499.1029	11.3437E-2	-499.8499	11.910E-2	-499.7758	20.1074E-2
11	-388.7206	2.7602	-380.2701	3.7180	-396.0202	2.3142	-397.0151	2.0714
12	-263.5439	7.3158	-257.2486	5.1130	-287.0655	7.7023	-285.0756	8.4215
13	-160.4450	5.1433	-158.1615	5.6212	-180.1075	9.9533	-169.6112	11.3929
14	1350.3154	267.3899	1030.1151	178.3200	168.5423	166.5590	87.0873	141.5081
15	2090.5090	156.9597	1112.2961	125.7929	1983.9856	244.4828	479.3273	200.3958
16	201.1530	18.5114E-2	201.1135	21.4722E-2	201.2751	24.7795E-2	201.1070	22.5713E-2
17	333.2465	5.1286	338.0571	4.2009	316.9115	3.2064	316.2581	3.1796
18	424.5464	3.0416	431.4449	4.6895	414.5922	2.7948	414.2326	2.3850
19	502.8092	46.5163E-2	503.1865	51.4913E-2	500.8897	42.7528E-2	501.0482	87.3303E-2
20	604.0924	16.3105E-2	602.4839	18.7561E-2	603.8289	32.8581E-2	602.3309	35.6801E-2
21	1100.1939	27.7566	1100.1939	34.500E-10	1100.1939	27.7566	1100.1939	68.200E-14
22	2546.6971	209.5217	2241.9218	185.8858	1233.4577	168.8049	1079.3355	279.1669
23	3053.8101	195.3891	2544.0314	158.6696	2912.9330	242.4062	2032.3326	490.0057
24	1224.6044	1.8565	1223.6982	1.4481	1223.2766	3.9215	1222.6930	15.3119
25	1323.9070	1.5264	1322.5013	1.4241	1323.0140	5.8763	1321.3607	3.1662
26	1427.0967	35.1082	1400.2457	9.10547E-2	1400.0157	44.4678	1400.0159	29.6483
27	1847.9867	17.0230	1839.8457	16.2134	1832.5907	24.1561	1836.7544	52.7770
28	1700	27.7297	1700.0121	48.6305	1700	47.0588	1700	93.5813

Tabla 4. Resultados obtenidos en 30 Dimensiones

Función	Rand1		Rand1 Perturbación		Best1		Best1 Perturbación	
	Mediana	Desviación Estándar	Mediana	Desviación Estándar	Mediana	Desviación Estándar	Mediana	Desviación Estándar
1	-1346.4059	38.0542	-1118.0943	141.0085	-1400	0	-1400	41.000E-8
2	112E7	277E6	979E5	167E5	659E5	703E5	332E5	148E5
3	2070E9	2000E9	1970E7	393E7	155E7	2520E7	716E6	139E7
4	342908.65	246484.92	29231.02	3264.3626	183170.40	105788.71	20698.94	3932.5735
5	-670.2021	138.4160	-653.5791	98.0488	-1000	18.300E-10	-999.9996	6.6800E-4
6	-852.3908	26.7010	-809.5061	36.7907	-882.7099	1.3664	-877.1137	1.6291
7	1042.5698	652.6231	-666.9936	12.1286	-450.9967	226.2352	-712.5903	30.1951
8	-679.0687	5.3800E-2	-679.0628	5.2371E-2	-679.0441	4.62247E-2	-679.0565	4.50616E-2
9	-560.2177	1.0239	-560.3796	1.0978	-560.1388	1.0592	-560.1474	1.0517
10	689.1837	741.5460	108.5105	192.1903	-499.9822	1.22565E-2	-498.7924	67.4287E-2
11	-188.9647	27.2281	-165.6650	19.8788	-363.1566	10.6429	-363.7414	11.2915
12	-12.5828	18.9520	-13.3075	17.5848	-71.7929	15.5653	-49.5101	22.9677
13	89.1128	18.3198	91.0993	15.6000	31.5563	18.9024	51.7046	16.4528
14	6747.4326	714.8719	6736.8623	494.1199	2251.4205	487.9502	1645.5606	522.4326
15	8999.9815	343	7167.6465	334.7258	8727.1068	347.4994	6979.3171	1200.1692
16	202.5787	22.6687E-2	202.4857	26.0314E-2	202.5192	23.6755E-2	202.4469	29.5087E-2
17	578.3695	24.3320	579.8762	21.2909	378.2476	12.8320	387.3654	11.1742
18	665.9283	25.7968	674.5000	23.2997	468.5144	8.9153	469.2101	10.4613
19	539.5813	25.4176	544.5058	21.1697	503.2208	2.5988	507.9325	4.7273
20	613.9282	11.1009E-2	611.6082	20.923E-2	613.7171	14.437E-2	611.4694	27.4063E-2
21	1237.5016	75.9275	1139.7762	87.4000	1000	57.6963	1000.0135	52.6740
22	8725.8409	772.3042	7951.9067	390.4807	3109.2959	504.2832	2531.5257	471.7557
23	10153.5262	395.2123	8545.5701	301.9254	9700.0509	331.1385	8517.9607	335.5761
24	1308.5288	2.5034	1302.5127	3.5268	1304.2682	3.2614	1301.5633	3.6273
25	1405.1394	3.1100	1399.0560	2.7703	1400.3797	3.1719	1399.8855	3.0722
26	1606.5142	17.9361	1530.9653	63.2531	1604.2630	9.8524	1405.7701	93.8747
27	2680.5745	34.1000	2616.3925	28.5592	2642.5637	30.4504	2603.9495	24.8560
28	2478.2177	185.2082	2491.3137	123.3221	1700	203.7669	1700.0346	64.4479

Tabla 5. Resultados obtenidos en 50 Dimensiones

Función	Rand1		Rand1 Perturbación		Best1		Best1 Perturbación	
	Mediana	Desviación Estándar	Mediana	Desviación Estándar	Mediana	Desviación Estándar	Mediana	Desviación Estándar
1	-1040.8523	235	-335.3356	585.4279	-1400	1.2100E-12	-1400	38.200E-6
2	321E7	876E6	305E6	483E5	20E7	185E6	854E5	539E5
3	20900E9	13800E9	4610E7	595E7	311E9	478E9	991E7	870E7
4	609558	295433.54	49059.94	4829.3853	404703.52	148341.93	41253.35	4874.5962
5	114.6304	474	-549.7202	111.7359	-1000	61.400E-8	-999.9907	90.9358E-4
6	-774.3136	41.1148	-684.7121	64.4253	-855.3971	8.0002	-853.5313	15.1197
7	2242.5014	1088.6112	-659.7467	9.2511	-52.6781	337.6897	-693.0075	19.0840
8	-678.8529	3.96501E-2	-678.8578	2.9287E-2	-678.8664	2.86874E-2	-678.8707	3.49506E-2
9	-526.6454	1.2908	-536.6595	6.3188	-526.7943	1.3887	-561.7574	13.9882
10	2234.6962	1097.0866	702.5670	283.5973	-499.1533	43.5963E-2	-496.3909	2.3646
11	12.2708	32.9327	36.3994	34.0209	-301.3726	24.2598	-286.8243	31.0356
12	233.1595	35.5541	243.6663	21.3201	154.9661	28.4237	187.8832	38.8346
13	329.0811	32.4769	353.1823	27.9270	257.3082	32.3157	315.7222	39.7084
14	11686.1406	1050.4285	12317.1506	784.0558	4961.4007	808.5805	3952.7790	735.6795
15	16504.6334	385.4880	14364.0033	378.4235	16120.3115	437.0822	14190.8173	365.0883
16	203.4003	24.1954E-2	203.4057	27.1238E-2	203.2891	30.5194E-2	203.3514	27.1652E-2
17	827.5058	38.5225	829.2380	38.4022	474.9279	24.1299	491.3891	26.3886
18	902.9463	48.8572	904.9828	45.0742	542.3676	25.1382	556.5838	27.9382
19	1340.5712	4400.3963	745.9946	402.1813	507.6820	2.4078	518.5668	9.2166
20	623.7504	16.6811E-2	620.9047	26.7632E-2	623.5780	21.0339E-2	620.8574	32.5134E-2
21	1569.8515	159.7738	1424.6311	219.5310	1000	53.3391	1100.0115	49.1729
22	14161.6903	1319.6516	14501.5546	865.3156	6366.1895	876.7257	5077.7514	784.6218
23	17421.5151	377.4880	15651.9869	323.7425	17214.1185	376.9616	15535.7750	457.3987
24	1397.6106	3.2987	1386.4351	3.9145	1392.4623	4.1267	1388.2413	3.5376
25	1490.2184	4.6573	1484.0797	2.8575	1486.8253	4.1214	1482.6281	3.3060
26	1698.1012	3.8452	1683.9892	22.5093	1693.3772	4.0488	1686.4911	42.1806
27	3560.2109	41.1272	3470.2224	36.1716	3503.1063	41.4514	3447.8082	41.3196
28	2377.4740	1672.4561	2243.1992	117.8650	1800	1277.5309	1800.0504	745.3048

Tabla 6. Valores y p-Valores obtenidos por la prueba no paramétrica de Friedman

	2D	10D	30D	50D
Valor	2.9678	35.7535	35.9464	45.6428
p-Valor	0.3966	8.4452E-8	7.6891E-8	7.2075E-10

Tabla 7. Clasificación de algoritmos mediante el procedimiento Post-hoc descrito en [7]

Algoritmo	10D	30D	50D
Best1 Perturbación	1.5893	1.5536	1.5357
Best1	1.9821	2.1250	1.9286
Rand1 Perturbación	3.1429	2.8214	2.8929
Rand1	3.2857	3.5000	3.6429

Referencias

1. Al-dabbagh, R., Botzheim, J., Al-dabbagh, M.: Comparative analysis of a modified differential evolution algorithm based on bacterial mutation scheme. In: Differential Evolution (SDE), 2014 IEEE Symposium on. pp. 1–8 (Dec 2014)
2. Bhowmik, P., Das, S., Konar, A., Das, S., Nagar, A.: A new differential evolution with improved mutation strategy. In: Evolutionary Computation (CEC), 2010 IEEE Congress on. pp. 1–8 (July 2010)
3. Brest, J., Greiner, S., Boskovic, B., Mernik, M., Zumer, V.: Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems. Evolutionary Computation, IEEE Transactions on 10(6), 646–657 (Dec 2006)
4. Brest, J., Maučec, M.: Self-adaptive differential evolution algorithm using population size reduction and three strategies. Soft Computing 15(11), 2157–2174 (2011)
5. Das, S., Suganthan, P.: Differential evolution: A survey of the state-of-the-art. Evolutionary Computation, IEEE Transactions on 15(1), 4–31 (Feb 2011)
6. Das, S., Konar, A., Chakraborty, U.K.: Two improved differential evolution schemes for faster global search. In: Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation. pp. 991–998. GECCO '05, ACM, New York, NY, USA (2005)
7. Derrac, J., García, S., Molina, S., Herrera, F.: A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. Swarm and Evolutionary Computation pp. 3–18 (2011)
8. Droste, S., Jansen, T., Wegener, I.: Upper and lower bounds for randomized search heuristics in black-box optimization. Theory of Computing Systems 39(4), 525–544 (2006)
9. Einarsson, G., Runarsson, T., Stefansson, G.: A competitive coevolution scheme inspired by de. In: Differential Evolution (SDE), 2014 IEEE Symposium on. pp. 1–8 (Dec 2014)
10. El-Abd, M.: Black-box optimization benchmarking for noiseless function testbed using artificial bee colony algorithm. In: Proceedings of the 12th Annual Conference Companion on Genetic and Evolutionary Computation. pp. 1719–1724. GECCO '10, ACM, New York, NY, USA (2010)
11. El-Abd, M., Kamel, M.S.: Black-box optimization benchmarking for noiseless function testbed using particle swarm optimization. In: Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers. pp. 2269–2274. GECCO '09, ACM, New York, NY, USA (2009)
12. Fan, Q., Yan, X.: Self-adaptive differential evolution algorithm with zoning evolution of control parameters and adaptive mutation strategies. Cybernetics, IEEE Transactions on PP(99), 1–1 (2015)
13. Holland, J.: Adaptation in natural and artificial systems. University of Michigan Press (1975)
14. Jin, W., Gao, L., Ge, Y., Zhang, Y.: An improved self-adapting differential evolution algorithm. In: Computer Design and Applications (ICCDA), 2010 International Conference on. vol. 3, pp. V3–341–V3–344 (June 2010)
15. Kacker, R.N., Kuhn, D.R., Lei, Y., Lawrence, J.F.: Combinatorial testing for software: An adaptation of design of experiments. Measurement 46(9), 3745 – 3752 (2013)
16. Li, D., Chen, J., Xin, B.: A novel differential evolution algorithm with gaussian mutation that balances exploration and exploitation. In: Differential Evolution (SDE), 2013 IEEE Symposium on. pp. 18–24 (April 2013)
17. Luke, S.: Essentials of Metaheuristics. Lulu (2009)
18. Noman, N., Iba, H.: Accelerating differential evolution using an adaptive local search. Evolutionary Computation, IEEE Transactions on 12(1), 107–125 (Feb 2008)
19. Omran, M., Salman, A., Engelbrecht, A.: Self-adaptive differential evolution. In: Hao, Y., Liu, J., Wang, Y., Cheung, Y.m., Yin, H., Jiao, L., Ma, J., Jiao, Y.C. (eds.) Computational

Intelligence and Security, Lecture Notes in Computer Science, vol. 3801, pp. 192–199. Springer Berlin Heidelberg (2005)
20. Rodriguez-Cristerna, A., Torres-Jiménez, J., Rivera-Islas, I., Hernandez-Morales, C., Romero-Monsivais, H., Jose-Garcia, A.: A mutation-selection algorithm for the problem of minimum brauer chains. In: Batyrshin, I., Sidorov, G. (eds.) Advances in Soft Computing, Lecture Notes in Computer Science, vol. 7095, pp. 107–118. Springer Berlin Heidelberg (2011)
21. Storn, R., Price, K.: Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. J. of Global Optimization 11, 341–359 (December 1997)
22. Wang, G., Goodman, E., Punch, W.: Toward the optimization of a class of black box optimization algorithms. In: Tools with Artificial Intelligence, 1997. Proceedings., Ninth IEEE International Conference on. pp. 348–356 (Nov 1997)
23. Yang, X.S.: Nature Inspired Metaheuristic Algorithms. Luniver Press, 2da edn. (2008)
24. Zavala, A., Aguirre, A., Diharce, E.: Particle evolutionary swarm optimization algorithm (peso). In: Computer Science, 2005. ENC 2005. Sixth Mexican International Conference on. pp. 282–289 (2005)

Algoritmo evolutivo paralelo para aplicaciones en tomografía sísmica

Eustolia Carreón[1], José F. Ramírez[1], Miguel O. Arias[2], Edmundo Bonilla[1], Roberto Morales[1]

[1] Instituto Tecnológico de Apizaco, Apizaco, Tlaxcala, México

[2] Instituto Nacional de Astrofísica, Óptica y Electrónica, Tonantzintla, Puebla, México

{euce_79,edbonn,moralescaporal}@hotmail.com, federico_ramirez@yahoo.com.mx, ariasmo@inaoep.mx

Resumen. En este trabajo se realiza la paralelización sobre una Unidad de Procesamiento de Gráficos de la función de evaluación de una Evolución Diferencial (ED) que tiene como objetivo generar un modelo inicial de velocidades sísmicas en un volumen de la corteza terrestre. La función de evaluación incluye un algoritmo que traza los rayos sísmicos generados a partir de 7 fuentes de energía (shotpoints) hacia cientos de receptores (geófonos). En la etapa de paralelización, el cálculo del tiempo residual y el trazo de la trayectoria del rayo sísmico es asignado a una unidad mínima de ejecución en la GPU (hilo), que para la aplicación realizada en este trabajo fueron 4,440 ejecutándose simultáneamente en paralelo. Al ejecutar el algoritmo secuencial, el tiempo estimado fue de 8 segundos aproximadamente, mientras que en la versión paralela fue de 0.1 segundo. Los datos usados en este trabajo se obtuvieron de un experimento realizado en el campo volcánico El Potrillo, ubicado en el sur de Nuevo México, a cargo del Departamento de Ciencias Computacionales y del Departamento de Ciencias Geológicas de la Universidad de Texas, en El Paso.

Palabras clave: evolución diferencial, algoritmo de cobertura de rayos, CUDA, cómputo paralelo, unidad de procesamiento de gráficos.

1. Introducción

La tomografía sísmica es una técnica de imagen que procesa observaciones del movimiento de la tierra, recolectadas con sismómetros, para mejorar los modelos estructurales, y ha sido uno de los medios más efectivos para obtener imágenes del interior del planeta en las últimas décadas [1]. Un estudio de tomografía sísmica permite obtener un modelo de velocidades de la estructura de la corteza terrestre bajo cierta región. El conocimiento de estas velocidades tanto en longitud, amplitud y pro-

fundidad del volumen permite la localización de diferentes elementos que se encuentran bajo la tierra, por ejemplo: tipos de rocas, fluidos, gases, material de desecho, etc., debido a que cada uno de ellos presenta diferente velocidad de transmisión de ondas sísmicas de acuerdo a su densidad [2]. Para ello se usan diversas técnicas de búsqueda, en este caso en particular se utilizan los Algoritmos Evolutivos (AE), ya que tienen la capacidad de buscar en espacios grandes y no son dependientes de una solución inicial aproximada a la óptima como los métodos tradicionales basados en el gradiente; sin embargo, conlleva grandes costos computacionales, ya que generalmente este tipo de algoritmos generan poblaciones de soluciones candidatas, repitiendo el proceso muchas veces [3] y por lo tanto ralentiza la ejecución por horas o días, dependiendo de la cantidad de los datos que se usen. Debido a que muchas aplicaciones actualmente requieren mayor poder de cómputo del que una computadora secuencial es capaz de ofrecer, el cómputo paralelo ofrece la distribución del trabajo entre diferentes unidades de procesamiento, resultando mayor poder de cómputo y rendimiento del que se puede obtener mediante un sistema tradicional de un solo procesador [4]. Con el desarrollo de herramientas de programación de Unidades de Procesamiento Gráfico (GPUs, por sus siglas en inglés), varios algoritmos han sido adaptados a este hardware satisfactoriamente y la plataforma de computación híbrida GPU - CPU, ha alcanzado aceleraciones importantes, comparada con las implementaciones sobre CPUs únicamente [5]. En los últimos años se han implementado los AE en GPUs y se ha visto que son capaces de mejorar decenas de veces el desempeño mediante el uso de este hardware, sobre todo cuando se utilizan tamaños grandes de población [6].
En este trabajo se paraleliza, sobre una GPU, la función de evaluación de una ED, para generar un modelo inicial de velocidades sísmicas en un volumen de la corteza terrestre.

En la sección 2 se mencionan los módulos de un Algoritmo de Tomografía Sísmica (ATS). En la sección 3 se explica la ED usada y en la sección 4 se muestra el modelo paralelo. En la sección 5 se presentan resultados experimentales obtenidos y se hace una comparación entre el desempeño de las versiones secuencial y paralela. En la sección 6 se mencionan las conclusiones y los trabajos futuros.

2. Generación de frente de onda y cobertura de rayos

Diversos procesos constituyen un ATS [2], de los cuales en esta investigación sólo se consideran: la simulación de un frente de onda, la cual genera, en función de un conjunto de velocidades iniciales, los tiempos de llegada de las ondas sísmicas a partir de las fuentes de energía, hacia cada uno de los vértices de un modelo discreto del volumen de la corteza terrestre en espacios de 1 km^3 (Fig. 1); y el proceso de cobertura de rayos sísmicos, el cual se encarga de hacer el trazo de los rayos sísmicos siguiendo su trayectoria de forma inversa; es decir, desde el origen de los geófonos (destino del frente de onda generado) hacia cada una de las fuentes de energía (origen del frente de onda generado); considerando únicamente los que llegan a las fuentes y excluyendo los que se salen del volumen de estudio.

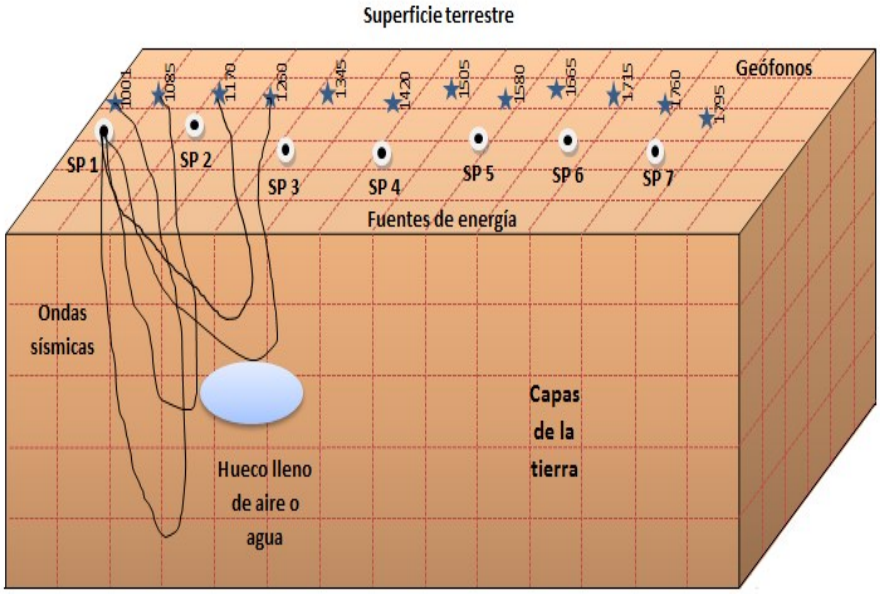


Fig. 1. Modelo discreto de la corteza terrestre

2.1 Representación de individuos

En el algoritmo, cada individuo de la población representa un conjunto de profundidades y velocidades en diferentes capas de la tierra, para este caso se tomaron 8 puntos de la profundidad debido a que anteriormente, con el uso de Estrategias Evolutivas, ha dado mejores resultados en comparación con otros valores [2]; sin embargo, este número puede variar de acuerdo a la elección de los usuarios. En la Fig. 3 se representa un individuo, el cual al ser visto como un vector, está dividido en dos partes. En la primera, los valores 1 y 69 ocupan las posiciones 0 y 7, respectivamente, correspondientes al mínimo y máximo de la profundidad en kms, y las 6 posiciones intermedias son datos de tipo entero, diferentes entre sí, y que deben ser ordenados de menor a mayor. En la segunda parte del individuo, se asignan las velocidades en un rango de 3 a 8 km/s; que cumplen con las mismas restricciones de la primera pero son valores reales. Las velocidades para las 61 capas restantes de las 69 en total se calculan mediante interpolación lineal.

La representación de cada individuo muestra que a cada valor de profundidad le corresponde una velocidad, debido a que la finalidad fue generar un modelo de capas planas, donde cualquier punto dentro de toda la superficie de una capa en determinado valor de profundidad tiene la misma velocidad; y aunque se sabe que en la realidad no es así, es usado de esta manera para obtener un modelo inicial de velocidades y realizar posteriormente una simulación de frente de onda.

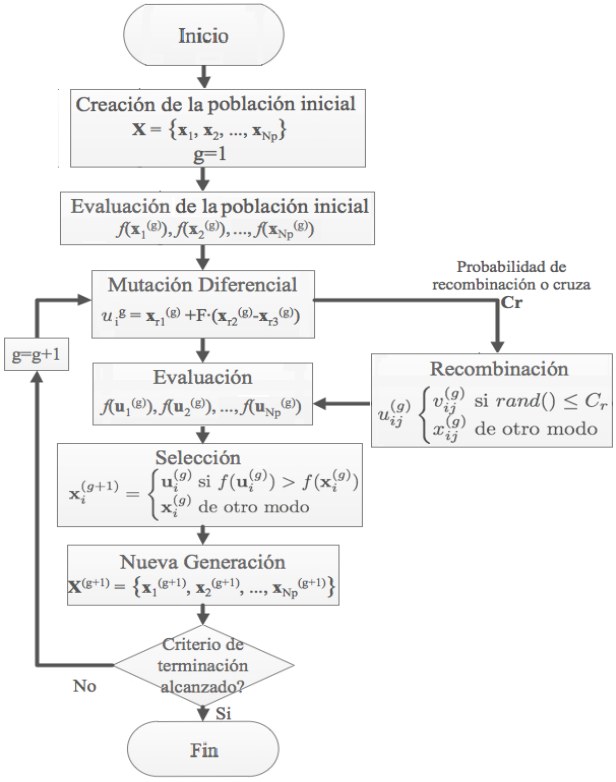


Fig. 2. Diagrama de flujo de una ED

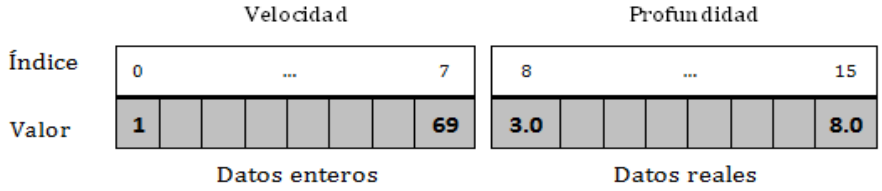


Fig. 3. Representación de un individuo

2.2 Mutación

El operador de mutación proporciona diversidad a la población a través de la introducción de nuevas soluciones, evitando la convergencia prematura en el algoritmo; es decir, la posibilidad de una rápida obtención no necesariamente del óptimo global. Para superar este problema es necesario conservar la diversidad en las generaciones, mediante un parámetro denominado factor de escala (F), elegido por el usuario para controlar la amplificación de la diferencia entre 2 individuos, así como para evitar el

estancamiento de la búsqueda, permitiendo la exploración de otras áreas en el espacio de búsqueda [8]. Su valor es tomado del rango de 0 a 1 [7]; de 0 a hasta 1.2 [10]; o bien es establecido a 0.5 [11]. Para este trabajo se asignó a F un valor aleatorio entre 0 y 1. La mutación se implementó mediante la siguiente ecuación

$$\vec{u_j} = \vec{x_{r1}} + F.(\vec{x_{r2}} - \vec{x_{r3}}) \tag{1}$$

Donde $\vec{u_j}$ es el vector mutante de la población y r1, r2, r3 son vectores de la población representados por valores enteros desde 0 hasta el número total de individuos, los cuales son diferentes entre sí y diferentes del índice en funcionamiento. En este trabajo el 80% del total de la población fue generado mediante este operador.

2.3 Recombinación o Cruza

La recombinación o cruza es una de las operaciones evolutivas implicadas en la generación de nuevos individuos. Esta operación es llevada a cabo con la participación de dos o más padres, quienes heredan rasgos o características a sus descendientes mediante una mezcla de información que se da de manera aleatoria. El parámetro más importante en esta operación es la tasa de recombinación (CR), cuyo valor define el porcentaje de la población generada con este operador, y en contraste con los Algoritmos Genéticos debe ser pequeño. Para este trabajo se usó el método de cruza binomial y se le asignó a CR un valor de 0.2; se eligieron 2 padres al azar y se generó 1 hijo mediante la ecuación 2.

$$Hijo_i = \begin{cases} Padre1_i \; Si \; rand \in (0,1) \geq 0.5 \\ Padre2_i \qquad En \; caso \; contrario \end{cases} \tag{2}$$

2.4 Selección

Cada vez que un individuo x_i' fue generado, se evaluó su modelo de velocidades para determinar si formaría parte de la siguiente generación o no. Debido a que el problema en cuestión es de minimización, si el valor de la evaluación $f(x_i')$ era menor que el de su padre $f(x_i)$ este individuo pasaba a la siguiente generación; de lo contrario, el padre era elegido $f(x_i)$.

2.5 Función de evaluación

La función de evaluación de este algoritmo consistió en medir los tiempos residuales (dt, que es la diferencia entre el tiempo de llegada del rayo sísmico a cada geófono calculado y el tiempo observado). Para obtener el tiempo de llegada de cada uno de los geófonos se realizó una simulación de la propagación del frente de onda dentro del volumen de estudio, empleando el modelo de [12]. Se leyeron los datos de las ubica-

ciones de las 7 fuentes de energía, de los cientos de receptores distribuidos y de los tiempos de llegada de la onda sísmica, en base a un experimento de campo realizado por un geólogo. Para calcular el tiempo de llegada dentro de la celda donde se encontraba el geófono, se usó una interpolación trilineal, utilizando los tiempos de llegada del frente de onda a los ocho vértices de la celda. Estos dos procesos son parte del ATS propuesto por [12,13]. De acuerdo a los valores iniciales de la velocidad y la profundidad (representadas en cada uno de los individuos), se generó mediante el método de Vidale [12] un modelo inicial de tiempos de llegada del frente de onda y a continuación se trazaron las trayectorias de los rayos sísmicos, siguiendo la dirección del gradiente de frente de onda en cada celda, desde el receptor a la fuente. Después de que se generaron los tiempos de viaje, se calculó el tiempo residual mediante la diferencia del tiempo observado y el calculado. Una vez realizadas estas operaciones para cada rayo que atravesaba el modelo desde el geófono hasta la fuente, se repitió el proceso para el resto de las fuentes y de los receptores y se obtuvo el valor cuadrático medio (RMS, del inglés, Root Mean Square), ver la Fig. 4.

```
Leer GeofonosxFuente, tiempos [414414]
NoFuente=1;
while NoFuente ≤ TotalFuentes do
    Leer posicionNoFuente
    NoGeofono=1;
      while NoGeofono≤TotalGeofonos do
        Leer posicionNoGeofono, t∘
        Hallar la celda que contiene NoFuente
        Obtener tc  (Interpolación con tiempos [8])
        dt=posicionNoGeofono - posicionNoFuente
        nCeldas=0;
        rayo=posicionNoGeofono;
        while rayo≠posicionNoFuente do
            Calcular gradiente del rayo en la celda
            Encontrar el paso para cambiar de celda
            nCeldas=nCeldas + 1;
              if rayo=posicionNoFuente then
                Obtener longitud del rayo
                  for i=1 hasta nCeldas do
                    Obtener trayectoria del rayo
                  end for
              end if
          end while
          NoGeofono=NoGeofono + 1;
      end while
      NoFuente=NoFuente + 1;
end while
Regresa dt
```

Fig. 4. Algoritmo de cobertura de rayos

El pseudocódigo que se presenta en la Fig. 4 muestra el funcionamiento del algoritmo de cobertura de rayos sísmicos, el cual se ejecuta después de haber calculado los tiempos de llegada, mediante la simulación del frente de onda.

3. Paralelización del algoritmo de cobertura de rayos e integración de la ED en la GPU

El algoritmo de cobertura de rayos calcula el tiempo de llegada del frente de onda desde las fuentes de energía hacia cada uno de los geófonos localizados en una capa cercana a la superficie terrestre a evaluar (tc), y halla la diferencia entre éste y el tiempo observado en el experimento (to), además verifica que, según el modelo de velocidades propuesto por la ED, cada uno de los rayos sísmicos llegue hasta los geófonos, cruzando un conjunto de celdas internas del modelo. En el algoritmo secuencial, el tiempo de llegada al geófono, así como el cálculo de la trayectoria entre el geófono y el shotpoint es realizado geófono por geófono, ralentizando el proceso de cálculo, debido a que éste debe terminar antes de empezar con el siguiente.

En este trabajo se hizo la distribución del cálculo de dt, y la trayectoria de los rayos sísmicos, de manera que cada geófono se procesara sobre un hilo diferente en una GPU NVIDIA GeForce GT 430, de 96 núcleos. Se creó la mínima cantidad de hilos posible dentro de cada bloque con el propósito de que se llevara a cabo una ejecución efectivamente paralela, debido a que en la arquitectura SIMT (Single Instruction, Multiple Thread) se ejecutan los hilos en grupos paralelos de 32 llamados warps [15]. El modelo paralelo usado en este trabajo de investigación se muestra en la Fig.5, donde cada hilo realiza el proceso que se muestra en el algoritmo de la Fig. 4.

La información correspondiente a las ubicaciones de las fuentes de energía y de los receptores, así como la de los tiempos de llegada fue transferida desde la CPU a la GPU, con la finalidad de tener acceso a ésta de manera más rápida.

En la CUDA [16] los hilos se encuentran dentro de bloques. El número de bloques para este trabajo fue calculado mediante la ecuación 3:

$$\text{nBloques} = \left[\frac{\text{N}}{\text{NMAXHILOS}}\right] \tag{3}$$

Donde N representa a los 4440 geófonos que fueron evaluados en total, ya que si bien es cierto que el número de receptores distribuidos en el volumen fue de 793, no todos pueden ser considerados en los 7 shotpoints y tampoco tuvieron la misma información con respecto a las fuentes de energía, debido a sus distintas ubicaciones. NMAXHILOS identifica al número de hilos por bloque, que puede ser desde 32 hasta 512 o 1024, dependiendo del modelo de la GPU. En este trabajo se usó NMAXHILOS= 64, para un mejor rendimiento.

Como puede notarse en la ecuación, el número bloques es el entero inmediato superior al resultado de la división debido a que, como pueden variar el número de receptores y también el número de hilos por bloque, es posible que el número de bloques resultante de esta operación no sea suficiente para ejecutar los hilos que sean necesarios, por lo que realizando el cálculo de esta manera no se dará este caso; por el

contrario, es posible que sobren, pero cuando suceda a estos se les asigna un valor de 0 para ser considerados en el proceso pero sin afectar el resultado final.

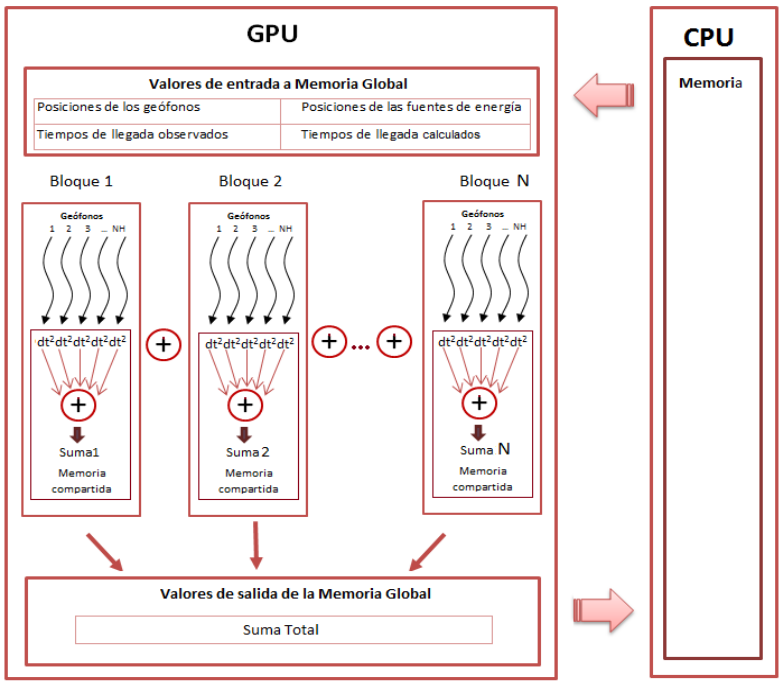


Fig. 5. Modelo de paralelización para algoritmo de cobertura de rayos

La función que ejecuta un hilo (kernel) incluye: el valor total de geófonos; total de fuentes de energía número; número de shotpoint que puede ser omitido en el experimento, pero es tomado en cuenta asignando 0 a los tiempos de los hilos que evalúen los geófonos de este número de fuente, para no afectar el proceso paralelo y el resultado final; las ubicaciones de las 7 fuentes de energía, en amplitud, longitud y profundidad del volumen de estudio; los identificadores de los geófonos; los tiempos de llegada de las ondas sísmicas, desde las 7 fuentes de energía hacia cada uno de los geófonos observados por el geólogo; los tiempos de llegada desde cada una de las fuentes de energía hacia todos los puntos del modelo 3D, calculados en la simulación del frente de onda; el resultado de la suma de los tiempos residuales al cuadrado (dt^2) de todos los geófonos; las posiciones del primero y último geófono que se evaluaron en cada fuente (ver tabla 1), debido a que los 4440 receptores fueron almacenados dentro de un vector unidimensional, además de que cada fuente genera un frente de onda diferente; es decir, 7 * 414 414 tiempos de llegada como se muestra en la Fig. 6, y son transferidos a la GPU. Cuando el algoritmo calcula el gradiente dentro de una celda, necesita los 8 tiempos de llegada de los vértices de esa celda; es por ello que el

kernel debe saber a qué fuente pertenece para realizar el cálculo de la posición en dicho tiempo.

Tabla 1. Posiciones de inicio y fin de geófonos, para cada fuente de energía

Inicio	0	488	1105	1839	2436	3124	3792
Fin	487	1104	1838	2435	3123	3791	4439

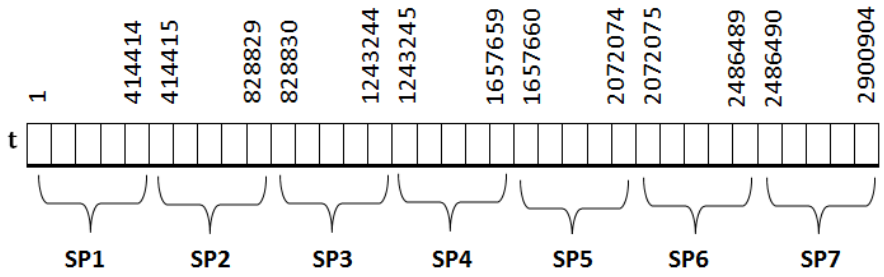


Fig. 6. Acceso al vector de tiempos

En este trabajo de investigación la memoria compartida fue usada para realizar la suma de los tiempos residuales de los geófonos en un bloque; por lo que se usó un arreglo de tipo *shared* dentro del kernel, cuyo tamaño estuvo en función del número de hilos por bloque que decidieron usarse (NMAXHILOS); en este caso se tomó el 64 debido a que tuvo un mejor desempeño en comparación con 16, 32, 128, 256 y 512.

Es importante mencionar que fue necesario un identificador del geófono dentro de cada bloque, y este se calculó con la ecuación 4:

$$I = NMAXHILOS \times blockIdx.x + threadIdx.x \quad (4)$$

Donde blockIdx.x es el identificador del bloque y threadIdx.x es el identificador del hilo dentro de cada bloque.

Pudo haber identificadores de geófonos que sobrepasaran el número total de receptores a evaluar, en consecuencia éstos no debían procesar ningún dato. Si este era el caso, al igual que se hizo con la fuente de energía omitida, automáticamente se le asignó el valor de 0 al resultado de dt, que almacenó el tiempo residual cuadrático.

Para la evaluación de cada hilo y debido a que la información de cada geófono era distinta, de acuerdo al número de fuente; se debió calcular, en base a I y a SP, el valor que se usara para desplazamiento en el vector que contiene los 414 414 tiempos de llegada de todas las fuentes de energía; con la finalidad de acceder al valor correspondiente para el hilo en ejecución. Este proceso se realizó verificando si I estaba dentro de los rangos contenidos entre el primer valor y el último de cada SP. Una vez que

todos los hilos ejecutaron sus procesos para encontrar el tiempo residual, se aseguró que todos terminaran para continuar con los demás cálculos.
Posteriormente, se realizó la suma de los valores de dt^2 de todos los hilos en cada bloque, y se almacenaron en sum. El proceso lo llevó a cabo sólo uno de los hilos y en este trabajo fue el threadIdx.x==0. La ecuación 5 se usó para llevar a cabo este proceso.

$$sum_b = \sum_{i=1}^{NMAXHILOS-1} dt^2 \quad (5)$$

Debido a que se hizo uso de datos almacenados en memoria compartida, el acceso a ellos fue mucho más rápido en comparación con la memoria global.
Para finalizar la función kernel, se hizo la suma total de los tiempos residuales pero de todos los bloques, haciendo uso de la ecuación 6:

$$\text{temp2} = \sum_{b=0}^{nBloques} sum_b \quad (6)$$

Donde nBloques es el número total de bloques, sum_b contiene la suma de los hilos en cada bloque, y $temp^2$ la suma de todos los bloques.

El kernel fue ejecutado de manera paralela en cada hilo de CUDA, devolviendo la suma total de los valores de dt^2.

4. Resultados

El algoritmo paralelo fue ejecutado 5 veces con poblaciones de 10, 20, 30, 40 y 50 individuos en cada una, sobre una tarjeta NVIDIA GeForce GT 430, la cual tiene 96 núcleos CUDA.

Las versiones secuencial y paralela fueron evaluadas de acuerdo al tiempo que tardaron en realizar el cálculo de la suma total de la diferencia al cuadrado de los tiempos de llegada a cada uno de los geófonos, con la restricción de que se generara un rayo sísmico entre la fuente y cada receptor.

La medición de tiempos en una versión secuencial se hace generalmente en la CPU; sin embargo, para efectos de igualdad en cuanto a las unidades de una medida únicamente, el algoritmo secuencial se ejecutó en un hilo dentro de un bloque sobre la GPU; no obstante, debido a que en CUDA la GPU tiene un tiempo máximo de cómputo por proceso, la tarjeta no soportó la ejecución de la versión secuencial para los 4 440 geófonos puesto que el tiempo requerido era mayor al permisible, por lo cual sólo fue posible registrar el tiempo de 1 070 de ellos antes de que la computadora terminara el proceso. Mediante el método regresión lineal se buscó un modelo en 2D que aproximara los valores de los receptores registrados para posteriormente calcular la tendencia de estos datos para 4 440 geófonos.

En la Fig. 7 puede apreciarse que, el comportamiento del tiempo tiende a incrementarse a medida que el número de geófonos aumenta. Según el cálculo realizado en este modelo, para 4 440 geófonos el tiempo estimado es de 8.9196 segundos.

En la Fig. 8 se graficaron los tiempos de ejecución del algoritmo de cobertura de rayos, en horas, en función del tamaño de población por cada versión. Puede obser-

varse que, el tiempo en la versión secuencial incrementa de manera constante, en contraste con la versión paralela donde se aprecia que el valor de la pendiente disminuye mientras el tamaño de la población crece, por lo que se deduce que si el tamaño de la población sigue incrementándose las líneas de la gráfica se separarán mucho más, remarcando la diferencia entre ambas versiones, donde la versión paralela superará en gran medida a la versión secuencial.

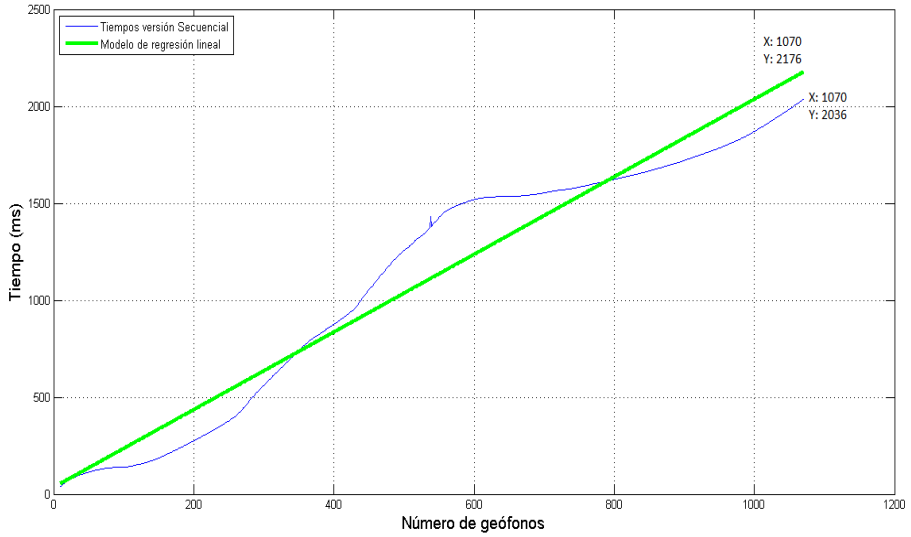


Fig. 7. Modelo de Regresión Lineal para la medición de tiempos en la versión secuencial

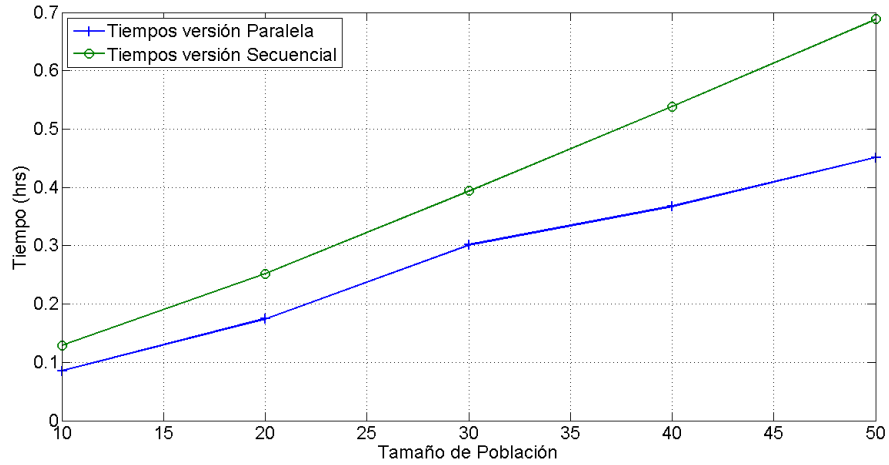


Fig. 8. Comparación de tiempos de la Versión Secuencial y Paralela

En la Fig. 9 se aprecia la gráfica con los mínimos valores de aptitud por tamaño de población. Como puede observarse, las aptitudes que corresponden al tamaño pobla-

ción 40 son, de cierta manera, las mejores en comparación con el resto de las poblaciones.

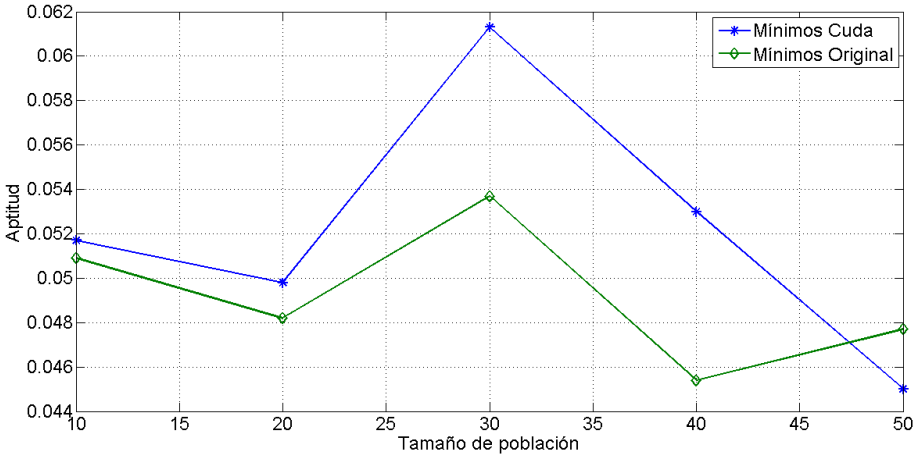


Fig. 9. Aptitud mínima por tamaño de población

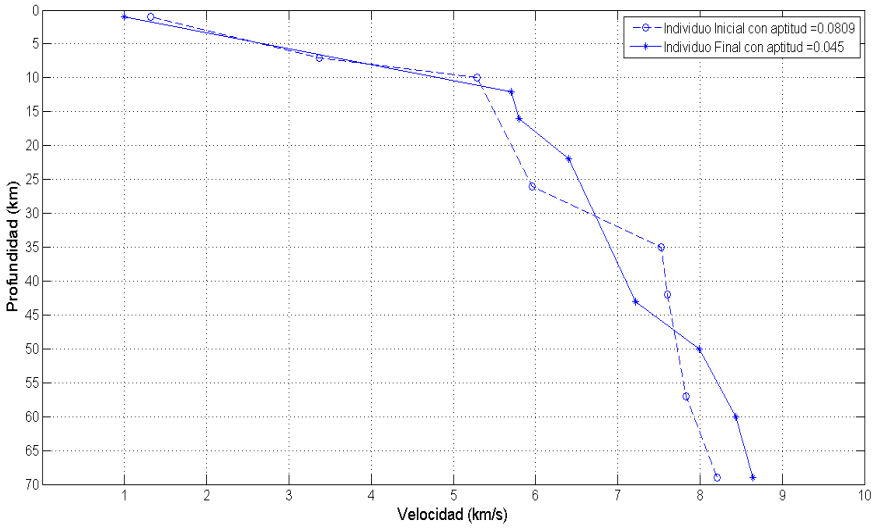


Fig. 10. Modelo de velocidades inicial con tamaño de población 40

En general puede notarse que, de acuerdo al tamaño de la población, la aptitud mejora en ambas versiones, a través de 10 generaciones y tiende a ser similar al final de cada evaluación; esto debido a que los procesos llevados a cabo son iguales en su forma secuencial y paralela.

La Fig. 10 muestra el modelo generado por la ED e indica la velocidad a la que se propagan las ondas sísmicas en cada una de las capas de los 69 km de profundidad en

el volumen de estudio. Este modelo fue obtenido por el individuo inicial y el individuo final de la versión en CUDA, con un tamaño de población de 40 individuos.

5. Conclusiones y trabajos futuros

En este trabajo de investigación se presentó la implementación de una Evolución Diferencial, donde la función objetivo es un algoritmo de cobertura de rayos sísmicos, cuya ejecución se llevó a cabo en una GPU. Si bien es cierto que la paralelización de los algoritmos evolutivos puede aplicarse en todo el proceso (generación de población inicial, operadores de selección, mutación y cruza, y la función de evaluación) la mayoría de los trabajos, incluyendo este, se enfocan únicamente a la paralelización de la función de evaluación debido a que es la que consume mayor tiempo de cómputo.

El modelo paralelo diseñado en este trabajo genera 7 frentes de onda (uno por cada fuente de energía), de tal manera que los 4 440 geófonos acceden simultáneamente a los tiempos de llegada. Esta solución mostró un mejor desempeño en comparación con la versión secuencial original al distribuir la evaluación de los geófonos de manera simultánea, en lugar en hacerlo uno detrás de otro.

La parte de mayor relevancia en este trabajo es el diseño del modelo paralelo, donde se asignó la evaluación de un geófono en cada hilo de CUDA y se formaron grupos de la mínima cantidad posible de hilos por cada bloque ya que la ejecución de manera simultánea de éstos dentro de un bloque se da en conjuntos de 32 hilos; y aunque este valor representa la agrupación de hilos más pequeña, en este caso se usaron 64 ya que este número mostró un mejor desempeño para la reducción del tiempo de ejecución del algoritmo de cobertura de rayos, al ser procesado sobre la GPU.

El desarrollo de este trabajo muestra la solución a uno de los módulos que conforman un ATS; sin embargo existen otros procesos tales como la generación del frente de onda y el suavizado de la propagación de rayos sísmicos, que serán paralelizados más adelante.

Otra propuesta es usar otros algoritmos de búsqueda u optimización como las estrategias evolutivas, o los algoritmos bioinspirados.

Por la parte de tomografía sísmica, se planea diseñar otro modelo paralelo para aprovechar la memoria compartida, el cual no realice el trazo de los rayos siguiendo la trayectoria de la onda celda por celda, sino que cada una de las celdas haga un monitoreo de los rayos que pasan por ellas, de tal manera que un hilo represente una celda y los 8 tiempos de llegada, del shotpoint a sus vértices, estén almacenados en la memoria compartida.

Referencias

1. Lee, E.-J., Huang, H., Dennis, J. M., Chen, P., and Wang, L.: An optimized parallel algorithm for seismic tomography. Computers Geosciences (2013)
2. Ramírez Cruz, J., Fuentes, O., Romero, R., and Velasco, A.: A Hybrid Algorithm for Crustal Velocity Modeling. In: Batyrshin, I. and Mendoza, M., edi-

tors, Advances in Computational Intelligence, volume 7630 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, pp. 329–337 (2013)
3. Oiso, Masashi and Matsumura, Yoshiyuki and Yasuda, Oshiyuki and Ohkura, Kazuhiro,:Implementing genetic algorithms to CUDA environment using data parallelization. Technical Gazette, Hrcak Portal of scientific journals of Croatia,, vol.18, pp. 511– 517 (2011)
4. Kirk, D. B. and Hwu, W.-m. W. : Programming Massively Parallel Processors: A Hands-on Approach. Morgan Kaufmann Publishers Inc., 1st edition, (2010).
5. Mu, D., Chen, P., and Wang, L.: Accelerating the discontinuous Galerkin method for seismic wave propagation simulations using multiple GPUS with CUDA and MPI. Earthquake Science, vol. 26, no. 6, pp. 377–393 (2013)
6. Chitty, D. M., Malvern, Q., and Ps, W.: A data parallel approach to genetic programming using programmable graphics hardware. In: GECCO 07: Proceedings of the 9th annual conference on Genetic and evolutionary computation, ACM Press, pp. 1566–1573 (2007)
7. González, S. J. D.: Implementación de un algoritmo de evolución diferencial paralelo basado en unidades de procesamiento gráfico. Master's thesis, Universidad Michoacana de San Nicolás de Hidalgo (2011)
8. Sun, C., Zhou, H., and Chen, L.: Improved differential evolution algorithms. In: Computer Science and Automation Engineering (CSAE), 2012 IEEE International Conference on, vol. 3, pp. 142–145 (2012)
9. Gao-yang, L. and Ming-guang, L.: The summary of differential evolution algorithm and its improvements. In: Advanced Computer Theory and Engineering (ICACTE), 2010 3rd International Conference on, vol. 3 (2010)
10. Bujok, P. and Tvrdik, J.: Parallel migration models applied to competitive differential evolution. In: Symbolic and Numeric Algorithms for Scientific Computing (SYNASC), 13th International Symposium on (2011)
11. Ao, Y. and Chi, H.: Experimental study on differential evolution strategies. In: GCIS '09, WRI Global Congress on Intelligent Systems (2009)
12. Vidale, J. E.: Finite-difference calculation of travel times in three dimensions. Geophysics, vol. 55, no. 5, pp. 521–526 (1990)
13. J. A. Hole: Nonlinear high-resolution three-dimensional seismic travel time tomography. Journal of Geophysical Research, vol. 97, no. 85, pp. 6553–6562 (1992)
14. Engelbrecht, A.: Computational Intelligence: An Introduction. Wiley, pp. 242–244 (2007)
15. NVIDIA Corporation: NVIDIA CUDA C Programming Guide (2014)
16. Sanders, J. and Kandrot, E.: CUDA by Example: An Introduction to General-Purpose GPU Programming. Addison-Wesley Professional (2010)

Un sistema recomendador móvil de rutinas de ejercicio basado en el perfil del usuario

Jaime Guzmán-Luna[1], Ingrid-Durley Torres[2], Sebastián Vallejo[2]

[1] Universidad Nacional de Colombia, Medellín, Colombia

[2] Institución Universitaria Salazar y Herrera, Medellín, Colombia

jaguzman@unal.edu.co, i.torres@iush.edu.co, j.vallejo@iush.edu.co

Resumen. Este artículo, describe la investigación y el desarrollo de un sistema recomendador móvil que usando técnicas de Inteligencia Artificial (IA), permite sugerir rutinas de ejercicio, orientadas a fortalecer la calidad de vida del usuario basándose en su perfil antropomórfico y patológico.

Palabras clave: rutinas de ejercicio, recomendador, dispositivo móvil, perfil antropomórfico, patología.

1. Introducción

Los gimnasios públicos al aire libre, surgen como respuesta al agitado ritmo de vida actual y tienen como objetivo ayudar a conformar sociedades en las que se tenga consciencia acerca del riesgo de una vida sedentaria sobre la salud, a su vez que facilitan adquirir hábitos saludables. Dichos gimnasios, por la flexibilidad de espacio y tiempo, carecen de un asesor experto que oriente a los asistentes en su actividad física y la conveniencia o limitante de uso de las mismas máquinas o de los ejercicios sobre ellas. La diversidad usuarios en edades, peso y la presencia de enfermedades, exigen una clara orientación para los asistentes, cuando acuden a realizar el ejercicio físico, ya que no todas las máquinas resultan convenientes a todos los usuarios, o no todos los ejercicios sobre una misma máquina están recomendados para todos los asistentes.

Ante este panorama, el problema de mejorar la calidad de vida de los usuarios que asisten a los gimnasios públicos al aire libre, resultan no solo infructuosos, sino que en algunos casos, también contraproducentes en la vida de ellos mismos. Bajo las anteriores circunstancias, se hace necesario desarrollar una herramienta soportada en los sistemas de software implementados con técnicas de Inteligencia Artificial (IA), que enseñen el uso correcto de las máquinas disponibles en los gimnasios públicos al aire libre; centrándose en la personalización de los usuarios y su preocupación por mejorar su calidad de vida. Dicha personalización, se determina en este trabajo, a partir de un conjunto de características antropomórficas y patológicas del usuario, que

permitan especificar la conveniencia de uso de las máquinas disponibles y los ejercicios que pueden realizarse sobre las mismas. Así, según los perfiles del usuario, algunas máquinas o ejercicios serán recomendados a determinados usuarios, mientras que para otros perfiles, los ejercicios deben ser adecuados en intensidad, frecuencia, tiempo o tendrá que ser ignorados como parte de su actividad física. El sistema incorpora además, elementos educativos, que constituyen una alternativa para complementar y reforzar el aprendizaje al usuario; indicando claramente la importancia del uso apropiado de la máquina y el ejercicio, cuando se persigue fortalecer la vida sana y saludable del individuo.

Paralelamente, las nuevas tecnologías en el desarrollo de aplicaciones móviles ofrecen los mecanismos necesarios para que un usuario pueda estar en contacto con la información en cualquier momento y lugar, a través de cualquier tipo de dispositivo móvil. Otro aspecto clave de las aplicaciones desarrolladas a través de estas tecnologías, se soporta en la portabilidad y masificación que ofrecen. Finalmente, y debido a su cotidianidad, los usuarios se sentirán en confianza utilizando este tipo de aplicaciones con las que se encuentra familiarizados. Tales características, convierten la tecnología móvil, en una alternativa para proporcionar conocimientos de formación, con mayor facilidad de retención gracias a la disponibilidad de un estudio continuo.

La motivación de este trabajo se concentra en establecer la investigación y el desarrollo de un sistema recomendador móvil que usando técnicas de IA, permita sugerir rutinas de ejercicio, orientadas a mejorar la calidad de vida del usuario, basándose para ello, en su perfil antropomórfico y patológico. Esta motivación es debida, a la falta de un sistema que promueva el ejercicio de la manera apropiada, sobre los elementos disponibles en los gimnasios públicos al aire libre, los cuales nacen como cumplimiento de las obligaciones de las instituciones gubernamentales, basadas en la protección de la vida de los ciudadanos de todas las edades. Lo que significa también, ofrecer espacios adecuados para el cuidado de la salud y para poder realizar actividades físicas de forma cotidiana, fomentando una vida saludable para el individuo y la sociedad.

2. Estado del arte

Usados para combatir los índices de sobrepeso y obesidad presentes en la comunidad de Shanghái y Bejín, China, los gimnasios al aire libre comenzaron a ser aplicados hace más de diez años, como menciona [1] en su estudio "La integración de Gimnasios al Aire Libre y el desarrollo local". En España, por su parte, se incorporaron al espacio ciudadano hace aproximadamente nueve años, con el fin de combatir la aparición de los efectos generados por el envejecimiento, cambiando también su denominación a Circuitos Biosaludables o Parques Geriátricos, según explica Hernández [2]. Estos escenarios, están ubicados al aire libre, en zonas verdes, dotados de máquinas diseñadas para soportar la intemperie, y adaptados para realizar los ejercicios que se realizan en los gimnasios comunes, permitiendo realizar trabajos de fuerza, de tipo aeróbico, de coordinación y movilidad articular [3]. Como explican Sáez [4], la base del trabajo que se realiza en estos espacios, está fundamentada en una rama de la fi-

sioterapia conocida como quinesioterapia o cinesiterapia, la cual se enfoca en el tratamiento de patologías a través del movimiento bien sea pasivo o activo. En Colombia, ciudades como Bogotá, Medellín, Pereira, entre otras, cuentan con estos gimnasios lo largo y ancho de su geografía, generando nuevas posibilidades a la comunidad para la práctica de ejercicio físico.

Si bien se evidencia en los estudios [5], [6], que este tipo de escenarios fue pensado para la población adulta, al estar al alcance de toda la comunidad y no tener ninguna restricción de uso, ni nadie que vigile o acompañe la práctica del ejercicio, la afluencia de usuarios es masiva y de una gran diversidad.

Desde la perspectiva de las aplicaciones móviles, la oferta es aún más variada y de múltiples alcances. La Tabla 1, presenta de manera resumida y por límites de espacios solos algunas de ellas, orientadas específicamente a caracterizar la actividad física en gimnasios, aunque ninguna se oriente a los ubicados al aire libre;

Tabla 1. Aplicaciones móviles asociados a actividad física.

NOMBRE APLICACIÓN	SISTEMA OPERATIVO	TIPO DE EJERCICIO	CARACTERISTICAS
GYM GUIDE	IOS, ANDROID	Gimnasio	• Formato en inglés. • Base de ejercicios. • Imágenes de ejercicios.
UFC GYM	ANDROID	Gimnasio	• GPS para ubicar el gimnasio de entrenamiento. (horarios) • Menú en inglés.
COMPLETE GYM GUIDE	ANDROID	Gimnasio	• Idioma inglés. • Recomendaciones para ejercicio. • Fotos e imágenes para explicar los ejercicios.
EJERCICIOS GYM	ANDROID	Gimnasio	• Video de los ejercicios con introducción en Ingles. • Espacio para diseñar rutina. • Rutinas recomendadas por objetivos.
GYM GUIA COMPLETA	ANDROID	Gimnasio	• Videos de los ejercicios (como avatar) • Explicación del ejercicio abajo del video. • Rutinas por niveles y cantidad de días a la semana. • Tips de entrenamiento. • calculadoras de calorías, IMC entre otras. • Teoría sobre alimentación. • Imágenes de motivación.
GYM SPORT	ANDROID	Gimnasio	• Imágenes de un libro de entrenamiento. • Guía de rutinas por objetivo y por nivel. • Teoría sobre alimentos • Teoría sobre anatomía y antropometría.

Sin embargo, se destaca la articulación con hardware del dispositivo, como el GPS, la incorporación de contenido multimedia para presentar la forma en que se debe realizar el ejercicio, el acompañamiento de calculadores para definir el IMC (índice de Masa Corporal), el peso ideal en otros; también se destaca, el idioma de presentación, donde prima el inglés o su carácter gratuito por un limitado tiempo, además de

la ausencia de consideraciones pedagógicas o patológicas del individuo; sin mencionar que se trata ejercicios físico que deben ser desarrollados en máquinas con especificaciones técnicas acordes con las disponibles en los gimnasios de servicio pago.

Desde el aspecto académico, en la literatura, no existen definiciones formales de rutina de ejercicios; sin embargo, se pueden afirmar de manera categórica que [7] una rutina está compuesta de ejercicios físicos y que estos a su vez, son entendidos como aquellas actividades físicas que son planificadas, estructuradas, con un objetivo definido y con las cuales se pretende mejorar la forma física [8]. Los trabajos, también presentan algunos tipos de sistemas ideados para la recomendación de rutinas, cada uno de ellos proponiendo diferentes métodos o técnicas para hacerlo. A continuación Tabla 2, recopila las técnicas más utilizadas.

Desde esta perspectiva, una de las limitantes más significativa sobre estos sistemas es que exigen tener información previa para evitar el arranque en frío [7], [9]. Otro aspecto preocupante, es la imposibilidad de incorporar dentro de los perfiles, los perfiles patológicos de los usuarios, ya que solo se concentran en recomendar en función de la antropometría del usuario.

Tabla 2. Métodos para la recomendación de rutinas.

Método	Descripción
Sistemas de agentes [10]	Define diferentes agentes encargados de realizar distintas tareas, el agente principal, es denominado entrenador y es encargado de considerar el perfil del usuario y su condición para recomendar.
Sistema experto difuso [11]	Basado en una serie de reglas que permiten tomar decisiones como si se tratara de un experto, adquiere retroalimentación del usuario y usa lógica difusa para cuantificarla.
Basado en Ecuaciones [13]	Según una ecuación que calcula el trabajo necesario para consumir una cantidad de calorías, recomienda la (rutina) al usuario.
Basada en grafos [14]	El modelo de rutina se representa con grafos, calculando el peso de los nodos según tiempo y exigencia, elige el grafo que mejor se ajuste a los requisitos del usuario.
Índices de similitud [15]	Almacenan la información de los usuarios, conocimiento sobre medicina y nutrición en ontologías y con estas alimentan el sistema a fin de elegir la rutina que mejor se ajuste al usuario.

3. Public GYM: arquitectura y funcionalidad

Public Gym es un sistema recomendador móvil que basado en técnicas de Inteligencia artificial, permite sugerir a un usuario una rutina de ejercicio, específicamente orientada a la actividad física en los gimnasios públicos al aire libre; considerando para ello las características antropométricas y patologías del usuario. Public Gym, es soportado por componentes pedagógicos que van desde la teorías de aprendizaje significativo y constructiva [14], hasta el uso de recursos como los objetos de aprendiza-

je (OA) [15], los cuales pueden ser visualizados en distintos dispositivos, a fin de aprovechar su accesibilidad, portabilidad y masividad.

3.1. Descripción de la arquitectura

Este sistema recomendador, debe basarse en una arquitectura de cómputo en la nube que permita su acceso por parte de los usuarios en cualquier momento y lugar, evitando el gasto en equipo cómputo para el mantenimiento del sistema, considerando a su vez, las limitantes de presentación y almacenamiento de los dispositivos móviles. Por otra parte, se hace necesario que los usuarios puedan registrar su perfil, obtener e instalar la aplicación móvil desarrollada desde el sistema y actualizar su información a través del tiempo, razón por la cual, se incorpora un proceso de autenticación y acceso sin límite de tiempo o veces al usuario.

El sistema, permite además, crear una aplicación educativa representando el conocimiento del orientador o entrenador en deportes, para ello se ha decido implementar un sistema experto que contiene una especificación de reglas que permiten recomendar las rutinas según los principales perfiles de usuario. Para lograr lo señalado, se propone una arquitectura modular de cuatro capas, como la señalada en la Fig. 1.

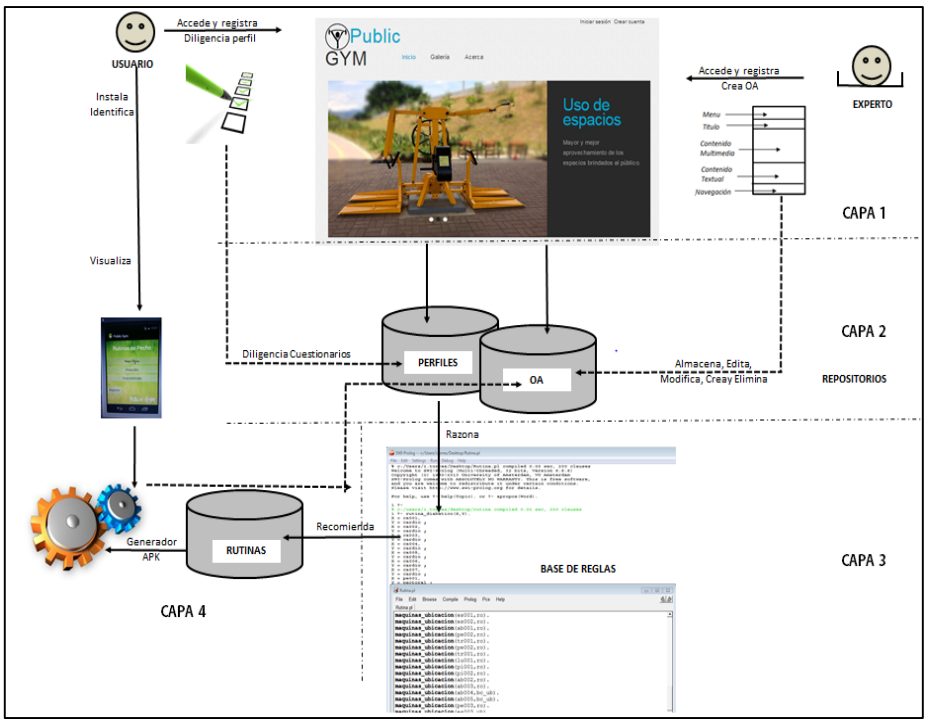


Fig. 1. Arquitectura del sistema recomendador.

En tal arquitectura, se consideran los siguientes Actores:

- A0: Usuario: Es el encargado de consumir los objetos de aprendizaje suministrados de acuerdo a un perfil establecido a través de una rutina de ejercicios y visualizados desde su dispositivo móvil.
- A1: Experto: Es el encargado de crear, editar, eliminar y actualizar, los contenidos registrados en el sistema recomendador, como son los OA y la información de las máquinas de gimnasio al aire libre. Para ello hace uso de los contendidos digitales (textos, imágenes y videos) y un par de plantillas para disponer tal contenido. Es él quien tiene la responsabilidad de diseñar acorde a su formación los parámetros para establecer las rutinas según los perfiles.
- A2: Administrador: Es el encargado de gestionar el sistema desde su aspecto técnico, además de los otros usuarios. Una de sus tareas claves, está representada en la traducción del conocimiento del experto al modelo de representación de reglas del sistema.

La arquitectura de cuatro capas, consta de una página web, de interfaz inicial, a través de la cual se consigue el acceso a la capa dos, la cual contiene el repositorio de OA, junto con la base de datos de los perfiles de cada uno de los usuarios (información suministrada por los usuarios, cuando se registra). En la tercera capa, se encuentra el sistema de reglas corazón del recomendador, además de la representación de la base de conocimiento del experto humano; en esta tercera capa, también se halla el repositorio de gimnasios, máquinas disponibles y actividad muscular permitida en cada una de ellas. En la capa final, se halla el componente responsable de seleccionar la rutina específica recomendada para el perfil de un usuario. Esta última capa, es accedida, desde la aplicación instalada en el dispositivo móvil, otorgando la portabilidad al recomendador. Las tres primeras capas, también están habilitadas para el usuario experto, quien las accede para actualizar el repositorio de objetos de aprendizaje, incorporado nuevos ejercicios, máquinas o la disponibilidad de un nuevo gimnasio.

3.2. Funcionalidad de la arquitectura

El sistema opera de la siguiente manera: Primero, un usuario experto, registra los OA, para ello, se hace uso de dos plantillas. Las rutinas, también son construidas bajo las recomendaciones del experto, y constituyen una secuencia de OA, acompañadas del trabajo del administrador, quien las convierte en conocimiento comprensible por el software. Una vez construido, el sistema de conocimiento, ya puede ser accedido por un usuario asistente a un gimnasio público, quien ingresa (normalmente desde su equipo de cómputo), a la página de acceso de la aplicación publicgym, generando un usuario y una contraseña; Posteriormente, el usuario debe diligenciar un cuestionario, el cual brinda la información necesaria al sistema para procesar mediante su base de reglas (sistema experto), la clasificación del usuario, dentro de uno de los perfiles disponibles (los cuales serán descritos en el apartado siguiente). El perfil del usuario es almacenado, junto con la identificación de la rutina que le corresponde. Una vez, el usuario lo desee, podrá acceder desde su dispositivo móvil, instalar la aplicación y luego de autenticarse, el sistema le habilitará la descarga de su correspondiente rutina. Esto con el fin de ahorrar espacio de almacenamiento, evitando otros elementos inne-

cesarios. Las cuatro capas, numeradas en la Fig. 1, están dispuestas en la nube, a fin de que brinden disponibilidad y acceso libre al usuario, ocupándose el sistema de todos los procesos de cómputo.

3.3. Gimnasios y máquinas

Para identificar cada uno de los gimnasios públicos existentes, se realizó primero, un levantamiento de información directo, que exigía visitar la página web de los entes gubernamentales, a fin de revisar sus registros públicos al respecto. Esta información fue contrastada con el "voz a voz" de distintos usuarios, quienes aportaron información sobre la ubicación de algunos gimnasios, ayudando a construir nuestro registro final el cual, se generó, como un consolidado con toda la información que relacionaba la dirección exacta de ubicación de cada gimnasio. Con el ánimo de dar un acercamiento real, se incorporó información acerca de qué máquinas había y la cantidad y el tipo de cada una de ellas en cada gimnasio. Finalmente y para dar un mayor detalle, se construyó un consolidado, cruzando cada máquina disponible, versus la zona muscular que permite ejercitar. El consolidado ha sido recopilado en una taba, de la cual se muestra una porción en la Fig. 2.

CODIGOS EJERCICIOS			
NOMBRE EJERCICIO	MAQUINA	EJERCICIO	UBICACIÓN
Aero Sky	1	CA001	RO
Elíptica	2	CA002	RO
Spinning	3	CA003	RO
Escalador	4	CA004	RO
Jinete	5	CA005	BC - RO
Pedaleo	6	CA006	RO
Mesa Cardio	7	CA007	RO
Press de Pecho en Mesa Cardio	7	PE001	RO
Press de Hombro en Mesa de Cardio	7	HE001	RO
SKY	8	CA008	BC
Pendúlo	9	CA009	BC
Dominadas Agarre Abierto	10	ES001	RO
Dominadas Agarre Cerrado	10	ES002	RO
Elevación de piernas en paralelas	10	AB001	RO
Fondos en paralelas	10	PE002	RO
Fondos en paralelas	10	TR001	RO
Fondos en paralelas	11	PE002	RO
Fondos en paralelas	11	TR001	RO
Hiperextensiones	11	LU001	RO
Press de Piernas	12	PI001	RO
Press de Pierna Individual	12	PI002	RO
Elevación de Piernas en Banco plano	13	AB002	RO
Crunch en banco Plano	13	AB003	RO
Crunch en banco inclinado	14	AB004	BC - UB
Elevación de Piernas en Banco Inclinado	14	AB005	BC - UB
Press de Pecho en Maquina	15	PE003	RO
Remo en Maquina	16	ES003	UB
Press Plano	17	PE004	BC - RO

Tabla referencias Zona ejercicio	
CA	Cardio
PE	Pectoral
HE	Hombro, Trapecio
ES	Espalda
TR	Triceps
BI	Biceps
AB	Abdomen
LU	Lumbares
PI	Muslo, Pierna, Gluteo

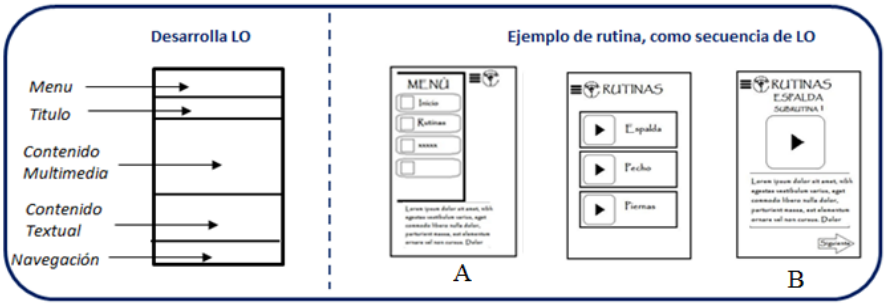
Tabla referencias ubicación	
RO	Robledo
BC	Barrio Cristobal
UB	Unidad Deprotiva Belen

Fig. 2. Información de Gimnasios, Máquinas y zona muscular.

3.4. Objetos de aprendizaje (OA)

Los OA son unidades de estudio, ejercicios o prácticas que pueden ser consumidas en una sección sencilla y que representan gránulos reutilizables que pueden ser creados, sin importar qué tipo de medio de entrega será utilizado [16]. Idealmente, los OA

pueden ser reutilizados y conectados juntos para construir aplicaciones que estén destinadas a servir a un determinado propósito o meta. En consecuencia, los OA necesitan ser libres del entorno, lo que significa que tienen que llevar información útil que describa el tipo y el contexto en el que pueden ser usados [17]. El diseño entonces de los OA, está básicamente representado por un par de plantillas, sobre las cuales se va a recopilar la información pedagógica [18]. Una de ellas, está diseñada para presentar el contenido educativo-motivacional (Fig. 3 A.), el cual corresponde a información clave que indica al usuario la razón por la cual se clasificó en ese perfil, o porque resulta tan importante que él realice los ejercicios bajo los parámetros que su rutina presenta. La segunda, corresponde al aspecto educativo-técnico (Fig. 3 B.), en donde se describe de manera textual y con un corto video, la manera correcta de realizar ese ejercicio y la forma adecuada de adoptar una buena postura sobre la máquina. El conjunto ordenado de OA, bajo determinados criterios definidos por los perfiles, responden a la especificación de una rutina de ejercicio. Tal como se señala en la Fig. 3. Cada OA, es almacenado dentro del repositorio, bajo los metadatos del estándar LOM [16]. Con ello, se facilita su clasificación, almacenamiento y recuperación.



A. Plantilla educativo-motivacional B. Plantilla educativa-técnica

Fig. 3. Plantilla de OA.

3.5. Perfiles de usuario

Para identificar la población que asiste a los gimnasios públicos [19], se consolidaron datos mediante el modelo de encuesta, sobre una muestra de 502 usuarios. Las variables que se recogieron, registran formas cualitativas y cuantitativas, definidas desde aspectos antropométricos como: edad, peso y talla, de los usuarios que asisten a los gimnasios públicos. Con tales datos, se pudo determinar el IMC o Índice de Quetelet [8], además de la frecuencia cardiaca máxima del sujeto. Las variables también, incluyen datos sobre el estado actual de salud de los usuarios, considerando la presencia y ausencia de enfermedades; finalmente, se ahondó en aspectos propios del entrenamiento como la hidratación antes, durante y después del ejercicio, la duración, intensidad y frecuencia de la sesión de entrenamiento a la que ellos estaban familiarizados realizar. Se incluyeron también, cuestionamientos relacionados con el nivel educativo, el grado económico e incluso la tenencia de dispositivos móviles. Los datos recogidos en las encuestas, fueron analizados con la ayuda del software estadístico de

acceso libre R versión 3.0.2, que permiten identificar las variables de mayor relevancia y facilitan la caracterización de la población que fue sujeto de estudio. Para este análisis se utilizó una estadística descriptiva en la que se usaron medidas de tendencia central que corresponden a valores numéricos que ayudan a localizar de alguna manera el centro del conjunto de datos. Para ello se manejó una media aritmética, que es aplicada estadísticamente para calcular datos que no se encuentran agrupados. Para determinar su valor sumamos cada uno de los datos y lo dividimos por el número de valores de la muestra de la siguiente manera:

$$\bar{X} = \frac{\sum_{i=1}^{n} Xi}{n} = \frac{X_1+X_2+X_3+\ldots++X_1}{n} \quad (1)$$

donde: $\bar{X}$: Es la media calculada
n: Número de datos
x_i: Es cada uno de los datos de la muestra

Luego de valorar y clasificar los usuarios bajo varios parámetros de las variables antes citadas, se consolidó la información relacionada con el hallazgo de personas con presencia de enfermedades crónicas, como hipertensión y diabetes Mellitus. Este estudio, se acompañó de la identificación del tipo de ejercicio que estos sujetos realizan teniendo en cuenta su condición y estado de salud. Él estudió permitió identificar desde las respuestas que un total de 16 personas con diabetes, 23 con alteraciones de la presión sanguínea, 2 con alguna patología que preferiblemente no mencionaron y 21 que reconocen su situación de sobrepeso, dato que difiere mucho de la realidad encontrada por cálculo de IMC. Los restantes 438, encuestados manifestaron no presentar ningún tipo de enfermedad, - clasificados como sanos-. Con la selección de la muestra y el resultado del estudio, se encontró que de los pacientes con diabetes son equivalentes al 3,2% de la muestra, un 4,6% son hipertensos, el 9,4% registra sobrepeso (dentro de los que está el l,8% que registra sobre peso), el otro 82,4% clasificó como saludable. Dados los análisis de frecuencia de asistencia a los gimnasios, se incluyó el perfil, el sedentario, el cual está representado por un 8,3% de la población.

3.6. Recomendador móvil

Es importante recordar que el propósito de las representaciones que se producen en un sistema de recomendación, se reducen a ayudar a los seres humanos a tomar decisiones y descubrir nuevos elementos, con menos esfuerzo, que si realizaran la actividad de manera manual. Considerando lo expuesto, resulta conveniente incorporar técnicas de IA, que basadas en conocimiento permitan explorar la estructura y las relaciones del mundo o dominio al que pertenece el problema, de la misma forma que permiten la reducción del número de posibilidades, tal como hacen los humanos. Las técnicas de recomendación [13] poseen varias clasificaciones basándose en las fuentes de datos sobre las cuales se hacen las recomendaciones y el uso que se le da a estos datos. La técnica de recomendación basada en el conocimiento (KBR: *Knowledge-Based Recommendation*) intenta sugerir objetos, basados en inferencias sobre las preferencias y necesidades del usuario. Se distingue de las demás técnicas, en que ésta

tiene un conocimiento previo funcional sobre cómo un ítem en particular puede satisfacer la necesidad de un usuario y por tanto puede razonar sobre la relación entre esta necesidad y una posible recomendación. El perfil del usuario, puede ser cualquier estructura de conocimiento que soporte esta inferencia.

El sistema recomendador trabaja en función de la interpretación de la información suministrada por un usuario. En este caso, se ubica al individuo dentro de un grupo poblacional y se le asigna una rutina que posteriormente podrá seguir a través de su visualización en un dispositivo móvil. Se trata entonces, de recomendar una rutina, basada en el conocimiento de un experto y la descripción del perfil del usuario, adicionándolo a un grupo poblacional identificado por las patologías más comunes. Con la información anterior, se construye un sistema de reglas que permite razonar y recomendar la rutina que resulte más conveniente, La Fig. 4, señala apartes de la base de conocimiento y la rutina recomendada para un diabético, bajo prolog. En este caso la rutina para el diabético, está adaptada de acuerdo a dos características principales que son: la zona de aplicación de la insulina (insulinodependientes) y la frecuencia del ejercicio.

Las rutinas que el sistema diseñe se ubican bajo la modalidad de circuitos y tendrán una base de 8 ejercicios de fuerza resistencia (FR) y 1 ejercicio aeróbico. Los 8 ejercicios de FR serán seleccionados teniendo en cuenta la zona de aplicación de la insulina en los pacientes. El sistema puede seleccionar los 8 ejercicios aleatoriamente teniendo en cuenta que un grupo muscular no este seguido de un ejercicio para la misma zona, es decir, en el Circuito se trabajaran los grupos musculares aproximadamente 2 veces por serie pero no de manera continua; así: 1 ejercicio pecho, 1 ejercicio de espalda, 1 ejercicio de brazos, 1 ejercicio de muslo/pierna, 1 ejercicio de abdomen, 1 ejercicio de hombro, 1 ejercicio de pecho, 1 ejercicio de espalda.

```
ejercita(tr003,triceps).
ejercita(tr004,triceps).
ejercita(pi009,muslo_pierna_gluteo).
ejercita(pi010,muslo_pierna_gluteo).
ejercita(es008,espalda).
ejercita(tr005,triceps).
ejercita(bi001,biceps).
ejercita(he005,hombro_trapecio).

%Perfil usuario
perfil(usuario,diabetico).
perfil(usuario,hipertenso).
perfil(usuario,obeso).
perfil(usuario,sano).
perfil(usuario,sobrepeso).


zona_inyeccion_diabetico(muslo).
zona_inyeccion_diabetico(abdomen).
zona_inyeccion_diabetico(brazo).
zona_inyeccion_diabetico(pierna).
```

```
% c:/users/i.torres/desktop/rutina compiled 0.02 sec, 200 clauses
1 ?- rutina_diabetico(X,Y).
X = ca001,
Y = cardio ;
X = ca002,
Y = cardio ;
X = ca003,
Y = cardio ;
X = ca004,
Y = cardio ;
X = ca005,
Y = cardio ;
X = ca006,
Y = cardio ;
X = ca007,
Y = cardio ;
X = pe001,
Y = pectoral ;
X = he001,
Y = hombro_trapecio ;
X = ca008,
Y = cardio ;
X = ca009,
Y = cardio ;
X = es001,
Y = espalda
```

Fig. 4. Base de Conocimiento Vs. Rutina recomendada para un diabético.

Dado que la zona de aplicación de la insulina no podrá ser trabajada, el sistema detectara tras la indicación del usuario dicha zona, y deberá eliminar el ejercicio que se enfoca en ese trabajo y cambiarlo por otro distante a dicha zona. Ejemplo: si en la rutina anterior el usuario indicó que la zona de aplicación fue el abdomen, este ejerci-

cio será omitido y se adicionara uno que será del tren inferior ya que para esta zona existe solo un ejercicio.

Toda esta información es modelada bajo reglas, tal como la mostrada en la Fig. 5. Aunque, en Fig. 5, solo se muestra una de las reglas del perfil diabético, el sistema también permite filtrar las máquinas según la ubicación del usuario y ejercicios según el perfil, definiendo intensidad y frecuencia. Evitando recomendar ejercicios no convenientes al usuario.

```
rutina_diabetico(W,D):- perfil(usuario,diabetico), zona_inyeccion_diabetico(Z), ejercita(W,D), not(D=Z).
```

Fig. 5. Regla base para recomendar rutina de diabético.

Las rutinas son almacenadas en un repositorio como archivos XML, que direccionan las URL de cada OA. Esto facilita su conversión a archivos APK y alivianan la tarea de despliegue del video dentro del archivo, el cual finalmente es visualizado por cada usuario en su dispositivo móvil, tal como lo muestra la Fig. 6.

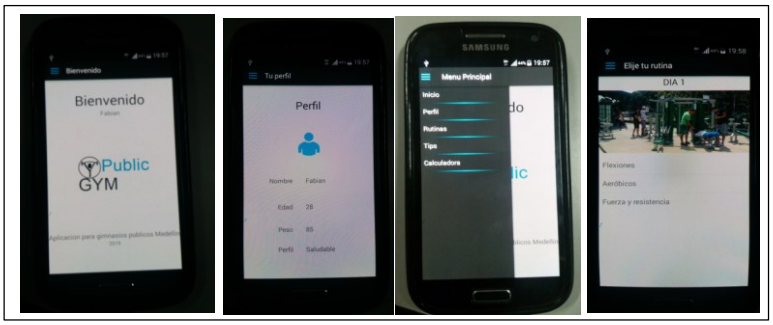


Fig. 6. Aplicación en dispositivo móvil**.**

4. Conclusiones

Este trabajo presenta las bases para el desarrollo de un sistema recomendador móvil, que usando técnicas de IA, permite sugerir rutinas de ejercicio, orientados a fortalecer la calidad de vida del usuario, basándose en su perfil antropométrico y patológico. La arquitectura de PublicGym, proporciona los niveles necesarios para el diseño de OA y la generación de rutinas, como secuencias de los mismos; mientras el sistema de reglas, permite recomendar una rutina basado en un perfil de un usuario, el cual es resultado de la combinación de las características antropométricas, junto con las patológicas. La arquitectura de cómputo en la nube, permite el acceso libre a la aplicación, así como libera al usuario de la carga computacional del proceso de recomendación y de la visualización de los videos, en el dispositivo móvil.

Mediante cada rutina recomendada, se espera el usuario ejecute de manera correcta su actividad fisica, mejorando su calidad de vida, razón por la cual, el recomendador incluye aspectos pedagógicos claves como la motivación y la técnica del ejercicio sobre la correspondiente máquina.

Como trabajo futuro, se espera incorporar técnicas de planificación en IA, para construir de manera dinámica las rutinas de ejercicio, incorporando la posibilidad de considerar algunas preferencias del usuario.

Agradecimientos. Los autores agradecen a Colciencias, a la Institución Universitaria Salazar y Herrera y a la Universidad Nacional de Colombia por cofinanciar el proyecto de investigación "Desarrollo de una plataforma tecnológica para la publicación de objetos de aprendizaje personalizados, aplicados al uso adecuado de los gimnasios al aire libre, en dispositivos móviles", presentado y aprobado en la convocatoria 626 del año 2013.

Referencias

1. Arufe V, Cortés L, Suárez X.: Estudio descriptivo de los servicios ofrecidos para los usuarios de parques biosaludables de Galicia. Retos, nuevas tendencias en educación física, deporte y recreación, no. 24, pp. 60–62 (2013)
2. Hernández E.: Estudio de los circuitos biosaludables para la tercera edad en España. Revista Internacional de Medicina y Ciencias de la Actividad Física y el Deporte, vol. 9, pp. 25–38 (2009)
3. Donciu M., Loniţă M., Dascălu M., Trăuşan-Matu E.: The Runner - Recommender System of Workout and Nutrition for Runners In: Proc. 13th Int. S Symbolic and Numeric Algorithms for Scientific Computing (SYNASC), pp. 230–238 (2012)
4. Sáez C, Rodríguez C., López R.: El parque geriátrico: Fisioterapia para nuestros mayores. Gerokomos, vol. 18, no.2, pp.84–88 (2007)
5. Hernández E, Fernández Rodríguez E, Merino Marbán R, Chinchilla Minguet L: Análisis de los circuitos biosaluables para la tercera edad en la provincia de Málaga (España). Retos, Nuevas Tendencias en Educación Física, Deporte y Recreación, vol. 17, pp. 99–102 (2010)
6. Pérez R, García Soidán J, Chinchilla Minguet J: Circuitos biosaludables y cumplimiento de las recomendaciones sobre actividad física para mayores. Revista Internacional de Medicina y Ciencias de la Actividad Física y el Deporte, vol. 12, no. 47, pp. 445–458 (2011)
7. Böhmer M., and Bauer G.: Exploring the Design Space of Context Aware Recommender Systems that Suggest Mobile Applications. In: CARS-2010, September 26, Barcelona, Spain, (2010)
8. American College of Sports Medicine (ed.): Manual ACSM para la valoración y prescripción del ejercicio, Paidotribo, Badalona (2014)
9. Setten M., Pokraev S., Koolwaaij J.: Context-Aware Recommendations in the Mobile Tourist Application COMPASS. Adaptive Hypermedia and Adaptive Web-Based Systems. Lecture Notes in Computer Science, Volume 3137, Springer-Verlag, pp. 235–244 (2004)
10. Barrera L., Ramos A. C., Barraza A., Martinez S.: ZuRoutine: Perzonalized Model of Exercise Routines. In: 6th Computer Colombian Congress, CCC 2011, pp. 1–6 (2011)
11. Suh M. K., Lee K., Nahapetian A., Sarrafzadeh M.: Interval Training Guidance System with Music and Wireless Group Exercise Motivations. In: International Symposium on Industrial Embedded Systems, SIES, pp. 110–119 (2009)
12. Palomares R. A., Ramírez J., Montaño A., Navarro J. A., Vazquez J. L: The Intelligent Personal Trainer. In: Proc. 16th IEEE International Conference on Electronics, Communications, and Computers (CONIELECOMP), no. 49 (2006)

13. Terveen, L., Hill, W.: Beyond Recomendador Systems: Heling People Help Each Other. In: J. M. Carroll (ed.) Human-Computer Interaction in the New Millennium, Addison-Wesley, ACM Press, New York, ch. 22, pp. 487–509 (2001)
14. Lim J. E., Choi O. H., Na H. S., Baik D. K.: A Context-Aware Fitness Guide System For Exercise Optimization In U-Health. IEEE Trans. Inf. Technol. Biomed., vol. 13, no. 3, pp. 370–379 (2006)
15. Wiley, D. A.: Connecting Learning Objects to Instructional Design Theory: A Definition, A Metaphor, A Taxonomy. Utah State University (2002)
16. IEEE Standards Department (2002). Draft Standard for Learning Object Metadata. IEEE Publication P1484.12.1/D6.4 (2002)
17. Vossen, G. P., Westerkamp G.: UDDI for E-Learning: A Repository for Distributed Learning Objects. In: Proc. 2nd International Conference on Information and Knowledge Sharing (IKS2003), Scottsdale, AZ, USA, pp. 101–106 (2003)
18. Ausubel, D., Novak, J., Hanesian, H.: Introducción a la Investigación Pedagógica. 3rd edn. Interamericana, Mexico (1989)
19. Pérez R, García Soidán J, García Núñez F, Chinchilla Minguet J.: Los parques biosaludables en Galicia. Revista de Investigación en Educación, vol. 8, pp. 55–61 (2010)

Identificación de gases mediante medición de complejidad

Mauricio Martínez M. y Miguel González-Mendoza

Tecnológico de Monterrey, Estado de México, México

A00964166@itesm.mx
mgonza@itesm.mx

Resumen. El análisis de datos provenientes de narices electrónicas son afectados por distintos factores, cuya principal característica es la aleatoriedad. Fenómenos como flujos caóticos y concentraciones irregulares de gases o presencia de gases ajenos a un gas por detectar, dificultan a los algoritmos de clasificación empleados en esta área su identificación. Debe considerarse también que la estructura de los vectores de estos datos son de alta dimensionalidad y pocas muestras, lo que dificulta descubrir patrones en ellos. En este trabajo se empleó el concepto de *Medición de Complejidad*, el cual define la cantidad de desorden presente en un sistema. Para demostrar la eficiencia de este concepto en la identificación de olores, se analizó un conjunto de mediciones de una nariz electrónica expuesta a mezclas de etileno, metano y monóxido de carbono a distintas concentraciones y bajo condiciones de flujo de gases muy cercanos a la realidad. Los resultados indican que la medición de complejidad logra discriminar los gases que constituyeron las distintas mezclas, identificando el desempeño de los sensores para reconocerlos de acuerdo a las magnitudes de complejidad y desorden.

Palabras clave: Medición de complejidad, sensores MOX, identificación de gases, clasificación por medición de complejidad.

1. Introducción

El empleo de narices electrónicas en el reconocimiento de aromas tiene diversas aplicaciones en las áreas de la industria, medicina, monitoreo ambiental, agricultura, etc. [1]. Estos dispositivos construidos bajo distintas tecnologías basan su funcionamiento en la reactividad que ciertos materiales tienen con determinados gases y manifiestan su presencia al afectar algunas de sus propiedades como son la resistencia, capacitancia, transconductancia, piezoeléctricidad entre otras [1] [2]. El uso de tecnología de semiconductores en esta área ha permitido la construcción de narices electrónicas de bajo costo; es el caso de los dispositivos MOX (Metal Oxide Semiconductor) cuya alta sensibilidad a determinados gases, se manifiesta con variaciones de transconductancia. El análisis de este tipo de mediciones tiene como objetivo mejorar la capacidad de discriminación de gases detectados por las narices electrónicas. Este proceso se ve afectado por

distintos factores como son la calibración del dispositivo, la forma en que se dispersan los gases, la presencia de otros gases o la concentración que tienen [2–4]. La naturaleza de los datos implica otra serie de dificultades para los algoritmos empleados en su análisis; son vectores de alta dimensionalidad y un número mínimo de los mismos. El proceso de análisis de esta información se desarrolla en las etapas de preprocesamiento, reducción y selección de atributos para finalmente llevar a cabo un proceso de clasificación o identificación del olor en el caso de la nariz electrónica [5–7].

Los investigadores en esta área están interesados en determinar el desempeño de los sensores para identificar, discriminar y conocer el grado de concentración de distintos olores en tiempo real [8–11]. Lo que permitiría detectar la fuga de combustibles, gases explosivos o tóxicos, o el monitoreo oportuno de calidad del aire entre otras situaciones donde el tiempo es crítico. Distintos algoritmos provenientes del campo de aprendizaje de máquina se emplean actualmente en este tipo de información en la reducción, selección de atributos y clasificación; esta última actividad tiene la dificultad de presentar una respuesta lenta debido al proceso de entrenamiento que requiere, y la cual debe llevarse a cabo nuevamente cuando los sensores fallan o terminan su vida útil. Además, establecer un proceso de clasificación de los datos es difícil, debido al comportamiento que tienen los sensores para su estabilización en el tiempo, y las lecturas de transconductancia presentan un cierto grado de aleatoriedad; aún trabajando en condiciones similares. Los algoritmos empleados comúnmente en estos procesos van desde K-NN (K nearest neighbors), Redes neuronales, Máquinas de Soporte Vectorial (SVM), Modelos de densidad, etc. [12–14].

El análisis de de datos obtenidos de narices electrónicas busca determinar la relación que existe entre un conjunto de variables independientes; lecturas de salida de un arreglo de n sensores, y un conjunto de variables dependientes denominadas olores. Cada sensor generará una respuesta en el tiempo, asociada al olor para el que fue diseñado reconocer. Por lo tanto, la salida de un arreglo de sensores o nariz electrónica tendrá una representación $X_{ij}(t)$, donde i representa algún sensor de la nariz y j el olor que reconocen [15].

Lo anterior representa un problema de clasificación, los algoritmos empleados en ésta área están basados en técnicas estadísticas o *paramétricas*; los cuales emplean las características de las distribuciones de datos como la media y desviación estándar para establecer la relación entre las salidas de los sensores y los olores que reconocen o clases. En tanto que los *no paramétricos* son implementados mediante técnicas de regresión lineal, rangos, distancia, etc. de tal forma que de las lecturas de los sensores, se pueda discernir su pertenencia a una instancia o clase. La forma en que se realizan las actividades anteriores pueden ser de forma supervisada, y esto se refiere a que durante el proceso de clasificación existe un conocimiento previo de relación entre datos y clases. El cual es utilizado para determinar la pertenencia de nuevos vectores de datos a estas clases mediante un proceso de aprendizaje o entrenamiento ya sea por calibración, correlación, regresión, etc. En cambio, la clasificación no supervisada no dispone de un conocimiento previo y separa los vectores de datos mediante

técnicas de agrupación, proyección lineal, etc. [2, 15].

Los algoritmos de clasificación al enfrentar la aleatoriedad tienen problemas en la identificación de regularidades durante el proceso de clasificación; algunos de ellos, demandan un conjunto más grande de muestras para mejorar su desempeño. Por otra parte, La naturaleza de los datos provenientes de narices electrónicas es de alta dimensionalidad y un número de muestras mínimo; lo que implica una demanda computacional alta, durante el proceso de entrenamiento de los algoritmos de clasificación [16]. Además, el proceso de aprendizaje tiene que ser llevado a cabo nuevamente cuando la nariz electrónica es ajustada y alguno o varios de sus sensores son reemplazados por deterioro.

Considerar la aleatoriedad como una característica de análisis en lecturas de salida de narices electrónicas ha sido pocas veces abordada. La medición de esta característica fue planteada por C. Shannon en su concepto de *Entropía de información* y desde entonces se han derivado varias técnicas de análisis de datos basados en él; ecuación 1 [17]. Se pueden mencionar Ganancia de Información, Máxima Entropía, Divergencia de Kullback-Leibler, entre otros. Estos algoritmos están clasificados entre las técnicas de selección y reducción de atributos. Basan su funcionamiento en la evaluación de la entropía de los datos y la valoración de su capacidad de ser informativos bajo este concepto. Son rápidos y permiten superar con facilidad la alta dimensionalidad que presentan algunos conjuntos de datos. La aleatoriedad es una característica que puede ser medida en ellos, y todo elemento aporta la información pertinente sin ser sesgado por la influencia de otros datos.

Entre los investigadores que han empleado conceptos derivados de entropía de información en la clasificación de olores con narices electrónicas están Alexander Vergara et al. [18] quienes utilizaron la distancia de Kullback-Leibler para identificar gases, calculando la divergencia de información que existe entre las distintas distribuciones de probabilidad de datos obtenidos de las lecturas de una nariz electrónica. Entendiendo por divergencia de información el grado de discrepancia que existen entre un par de distribuciones medido por el grado de entropía de de información mutua que poseen. El experimento llevado a cabo por Alexander Vergara tuvo como objetivo distinguir dos mezclas de gases; una de etanol con acetaldehído y acetona, otra de etanol con etileno. Los autores afirman que tuvieron una eficiencia del 100 por ciento en la discriminación de los gases analizados bajo condiciones controladas y liberación directa de los gases sobre los sensores. Otro trabajo que reporta el uso de teoría de la información en el análisis de datos provenientes de narices electrónicas lo llevaron a cabo X. Rosalind Wang et al. [6] empleando máxima información mutua. Este concepto es una variante de Kullback-Leibler que determina la similaridad de dos distribuciones de datos, calculando la entropía de su probabilidad conjunta y el producto de sus marginales. Con lo anterior los autores fueron capaces de descubrir los atributos más informativos contenidos en los datos que colectaron y los utilizaron en el proceso de clasificación de los gases mediante Máquinas de Soporte Vectorial y t-test. Concluyeron que los algoritmos SVM incrementan su eficiencia y para el caso de t-test no existe una mejora notable.

El objetivo de este trabajo es aplicar el concepto de medición de complejidad en el proceso de identificación de gases mediante sensores MOX. Demostrando, que un algoritmo de clasificación basado en este concepto, identifica de manera rápida y clara los gases sensados, e identificando el grado de aleatoriedad que tienen los datos ante las circunstancias fortuitas bajo las que se realiza el proceso de colecta de datos. Jordi Fonollosa et al. desarrollaron un experimento de detección de gases en condiciones muy similares a las condiciones de trabajo en las que son utilizadas las narices electrónicas; pusieron a disposición sus conjuntos de datos y se evaluará el desempeño del algoritmo propuesto en este trabajo con los datos de Jordi Fonollosa [19].

2. Métodos

El concepto de *Medición de complejidad* ha sido planteado por J. S. Shiner y Matt Davison como una función del *desorden* u *orden* que posee un sistema, ecuación 5. A su vez, el *desorden* es expresado como la entropía normalizada de los datos con respecto al nivel máximo de entropía que estos tendrían bajo una distribución uniforme, ecuaciones 3 y 2 respectivamente [20].

$$S = \sum_{i=1}^{n} -P_i log_2 P_i \tag{1}$$

$$S_{max} = Log_2 N \tag{2}$$

$$\Delta \equiv S/S_{max} \tag{3}$$

$$\Omega \equiv 1 - \Delta \tag{4}$$

$$\Gamma_{\alpha\beta} \equiv \Delta^{\alpha} \Omega^{\beta} \tag{5}$$

La aplicación de este concepto en datos muestreados en series de tiempo es simple, ya que las mediciones muestran eventos identificables y es posible determinar su frecuencia a partir de las lecturas obtenidas de los sensores MOX. El concepto de entropía de información se basa en la identificación de la probabilidad de eventos que conforman el conjunto de datos de algún experimento y la valoración de estos por su capacidad de ser informativos, ecuación 1.

La forma de aplicar el concepto de Medición de complejidad se hace de la manera más simple; los parámetros α y β se establecen con valor uno y produce la expresión más sencilla de medición de complejidad; la cuadrática.

Las condiciones del experimento llevado a cabo por Jordi Fonollosa *et al.* [19] en la detección de mezclas de Etileno con Monóxido de Carbono (CO) o Metano a distintos concentraciones son reportadas en la sección de materiales; donde se describe la forma en que se realizaron las mezclas y su dispersión que simuló turbulencias reales.

2.1. Materiales

El experimento utilizó un túnel de viento con un volumen de cámara de 2.5x1.2x0.4 m^3, 3 fuentes de gas de flujo controlado conteniendo etileno, metano

y monóxido de carbono respectivamente, y un arreglo de 8 sensores MOX. Los sensores detectan cuatro gases: Etileno, Metano, monóxido de Carbono (CO) y Propano; Tabla 1.

Tabla 1. Sensores MOX con material que sensan [19]

Tipo de sensor	Número de unidades	Gas que detectan
TGS2600	1	Hidrógeno, Monóxido de carbono
TGS2602	2	Amonio, Compuestos Organicos Volátiles (VOC)
TGS2610	1	Propano
TGS2611	1	Metano
TGS2612	1	Metano, Propano, Butano
TGS2620	2	Monóxido de carbono, Gases de combustible, VOC

2.2. Protocolo

Se propusieron treinta y cuatro experimentos con seis mediciones cada uno. Cada experimento es la combinación de distintas concentraciones de los gases a sensar, etiquetados en cuatro niveles: cero, bajo, mediano y alto. El flujo de los gases se realizó en condiciones estándar a 0 ^oC y un atm y su concentración fue manejada en partes por millón (ppm); Tabla 2.

Tabla 2. Tabla de experimentos [19]

		Etileno @ 2500 ppm			
		20 sccm	14 sccm	8 sccm	0 sccm
Metano @ 1000 ppm	300 sccm	6	6	6	6
	200 sccm	6	6	6	6
	100 sccm	6	6	6	6
	0 sccm	6	6	6	
CO @ 4000 ppm	200 sccm	6	6	6	6
	140 sccm	6	6	6	6
	80 sccm	6	6	6	6
	0 sccm	6	6	6	

*ppm denota partes por millón y $sccm$ el flujo en unidades estándar de centímetro cúbico por minuto.

Las salidas de medición de los sensores (transductancia) fueron registradas durante 300 s de acuerdo al siguiente esquema: 60 s sin la liberación de ningún gas, en el segundo 60 las fuentes de gas son abiertas de acuerdo a la combinación de gases a experimentar y la razón de flujo especificada (cero, baja, mediana y alta), esto se hace durante 180 s; después de este tiempo se cierra el flujo de

gas en las fuentes y se esperan 60 s hasta alcanzar la recuperación de línea base de los sensores. En cada experimento la señal de los sensores fue muestreada cada 20 x 10^{-3} s; reportando también la humedad relativa y la temperatura. lo anterior genera 8 vectores de mediciones correspondientes al número de sensores empleados en la nariz electrónica.

2.3. Algoritmo

Los vectores de medición de transconductancia de cada sensor, fueron normalizados bajo el concepto de *Escalamiento de atributos*, esto permite asegurar la comparación entre los vectores compensando las diferencias de escalas entre los mismos; pues coloca los datos entre un rango de cero y uno.

Al tener identificados los diferentes experimentos se dispone de un conjunto de clases que pueden ser distinguidas entre sí (Tabla 2). Cada experimento corresponde a una clase; y ésta a su vez, corresponde a una mezcla específica de gases. Por cada clase, se elige aleatoriamente un conjunto de datos, con los que se calcula el *desorden* de la transconductancia normalizada de las lecturas obtenidas por cada sensor; para posteriormente obtener su medición de complejidad. Mediante un gráfico de Medición de complejidad contra desorden se ilustra el desempeño que cada sensor tiene para detectar el gas para el que fue diseñado en función de las etiquetas de *cero*, *bajo*, *mediano* y *alto*. El algoritmo para realizar lo anterior se presenta en el *procedimiento Complejidad*.

1: **procedure** COMPLEJIDAD(DataSensor)
2: $DataNorm \leftarrow \frac{DataSensor_i - min(DataSensor)}{max(DataSensor) - min(DataSensor)}$
3: $FrecData \leftarrow Freq(DataNorm_i)$
4: $DesorData \leftarrow \frac{\sum_{j=1}^{N} -P_{FrecData_j} log_2(P_{FrecData_j})}{-log_2(\frac{1}{N})}$
5: $CompData \leftarrow DesorData * (1 - DesorData)$
6: **return** $CompData$
7: **end procedure**

3. Resultados

La presentación de los resultados se organizó de acuerdo al tipo de gas que detectan los sensores y estos fueron agrupados para comparar el desempeño que cada uno de ellos tiene en cada experimento. Los gráficos de medición de complejidad contra desorden de los sensores TGS2602 y TGS2602.1 fueron anexos entre sí por que sensan etileno (gas base), los sensores TGS2611 y TGS2612 por que sensaron metano, TGS2600, TGS2620 y TGS2620.1 monóxido de carbono y el sensor TGS2610 propano. Por ejemplo; en el experimento de etileno en cualquiera de sus concentraciones mezclado con monóxido de carbono a nivel medio mostró el desempeño del algoritmo y la capacidad de los sensores TGS2602, TGS2602.1 y TGS2600, TGS2620 y TGS2620.1 para identificar Etileno y CO respectivemente;

conforme se aumenta la cantidad de etileno, la complejidad de los datos de los sensores que lo sensan también lo hacen. Lo anterior supone una reducción del desorden; Fig. 1.

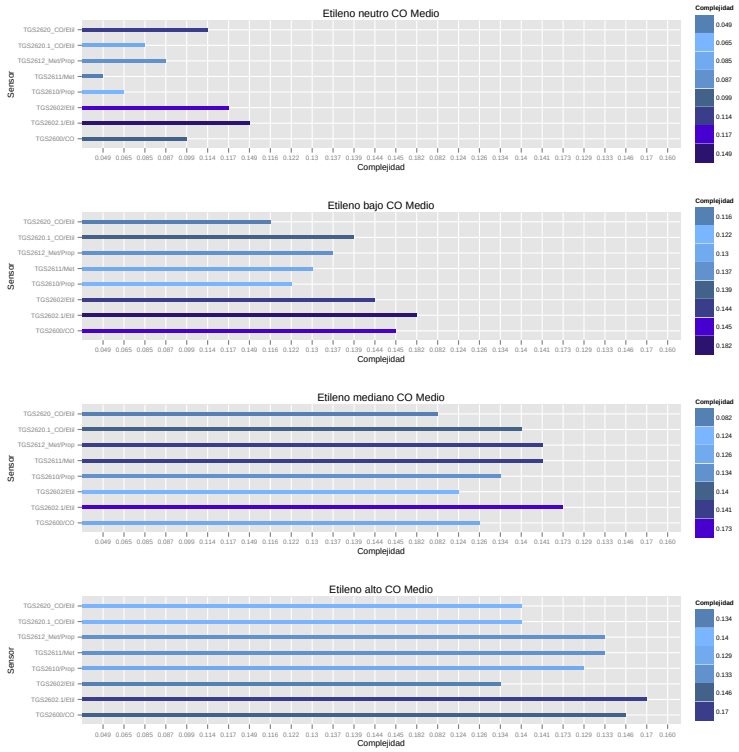


Fig. 1. Medición de complejidad por sensor. Experimento Etileno con CO medio

El arreglo de gráficas representa cada uno los sensores que constituyen a la nariz electrónica utilizada en el experimento. Cada gráfica muestra la Medición de complejidad correspondiente a las medidas de transconductancia de cada sensor. La variaciones de concentración de los gases mezclados se reflejan en cambios de magnitud de complejidad.

La complejidad en sensores como TGS2600, TGS2620 y TGS2620.1; los cuales detectan CO para el primero y VOC y CO para los dos últimos, muestran un aumento de complejidad conforme aumenta la concentración de etileno. Solo considérese que el etileno está catalogado como un VOC (Compuesto Orgánico Volátil); Fig. 1.

En los experimentos de etileno en sus distintas concentraciones con CO y Metano altos presentaron un comportamiento en complejidad similar al de

etileno con CO y Metano medio; Fig. 2. Sin embargo, el caso de etileno con CO y Metano bajos los sensores específicos para estos dos gases mostraron su más alto nivel del complejidad en ausencia de etileno o en un nivel bajo del mismo.

Finalmente, la detección de etileno en sus distintas concentraciones sin mezcla de CO o metano permitió observar un aumento natural de la complejidad en las mediciones de transconductancia de todos aquellos sensores que detectan este gas y una complejidad baja para el sensor de CO; excepto el sensor para propano; TGS2610, el cual presentó un aumento en la complejidad de sus mediciones para cada aumento de concentración de etileno.

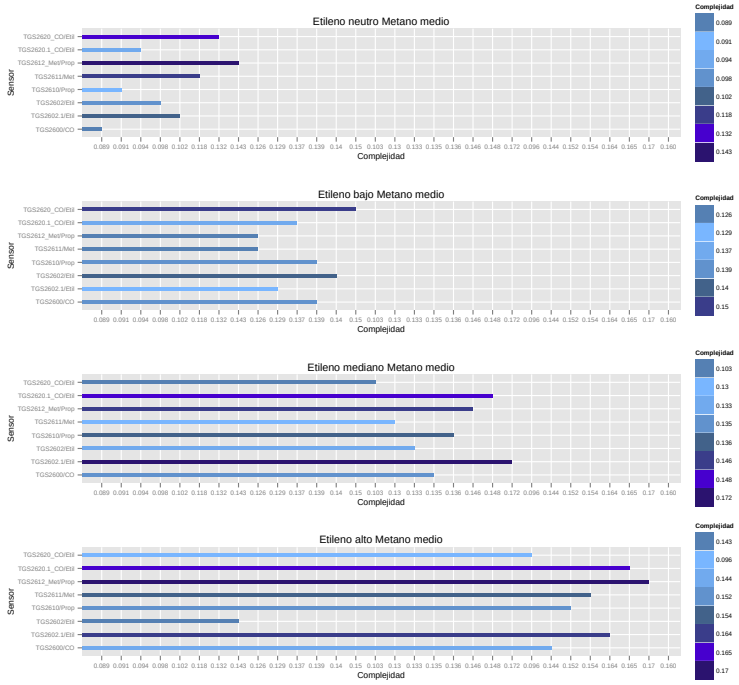


Fig. 2. Medición de complejidad por sensor. Experimento Etileno con Metano medio

4. Discusión

Los trabajos de Jordie Fonollosa *et al.* y Sepideh Pashami *et al.* [19], [14] plantean la identificación de gases en ambientes turbulentos y la detección del cambio de sus concentraciones como las condiciones reales bajo las cuales trabajan las narices electrónicas. Consideran oportuno tomar en cuenta lo anterior principalmente en aplicaciones donde el tiempo es un factor crítico; como en

la fuga y detección de gases tóxicos. Aunque emplearon Máquinas de Soporte Vectorial Inhibitorias (ISVM) y Razón de Verosimilitud generalizada (GLR); una variante del cálculo de verosimilitud, estos no dejan de ser afectados por la incertidumbre de los datos y la alta dimensionalidad de las muestras.

La medición de complejidad en este aspecto resulta ventajosa dadas las características que aporta el concepto base del mismo; que es la entropía de información. Ante un evento seguro (lecturas de mediciones constantes) o la ausencia de eventos, la entropía es cero; Eventos de baja probabilidad determinan bajos niveles de entropía y una distribución uniforme de probabilidad de los mismos supone el máximo valor de entropía que un sistema puede tener.

Ahora bien, existe una reducción de dimensiones al mapear las distintas lecturas de un sensor, al contar por el número de ocurrencias con el que pueden ser representadas bajo un conjunto finito de ellas o eventos. La forma de la distribución de los datos no afecta el análisis bajo este concepto; como lo pueden ser los métodos de estadísticos. La interpretabilidad es otra ventaja; si los datos presentan un alto de nivel de entropía no se puede aseverar una tendencia entre ellos por su misma aleatoriedad; pero si esta es baja, comienza entonces a reconocerse cierta predictibilidad en los datos y por tanto la detección de un patrón o clase.

Jordie Fonollosa *et al.* reportan una precisión de clasificación del 97 por ciento en la clasificación de las distintas mezclas de gases que analizaron empleando ISVM. En el mismo artículo muestran el comportamiento de las mediciones de transconductancia de los sensores que componen la nariz electrónica, durante los 300 s de cada prueba. De los cuales; solo dos sensores despliegan un comportamiento discriminativo en sus señales, los cuales es de suponer son los de compuestos orgánicos volátiles a los cuales pertenece el etileno. En tanto que el resto de las señales se traslapan en cierto grado y pertenecen al resto de los sensores que no se especializan en él. Si Jordie Fonollosa *et al.* evalúan en forma global la detección de etileno con las lecturas de los distintos vectores es posible que la mayoría afectará la capacidad de clasificación de la ISVM utilizada; pues solo dos sensores estaban dedicados al etileno; además de tener que superar los factores ya descritos.

En el caso de medición de complejidad; el análisis fue por sensor, y los resultados muestran la magnitud de esta característica que cada conjunto de mediciones de transconductancia sustenta. Bajo las circunstancias en que se desarrollaron los experimentos, los gráficos de complejidad muestran la capacidad de cada sensor para identificar los componentes de las mezclas de gases o hacer evidente la incapacidad de los sensores para identificarlos.

Es de notar que la capacidad de clasificación o discriminación de los sensores mediante los conceptos de medición de complejidad y desorden permiten interpretar el desempeño de los mismos. Magnitudes bajas de complejidad y de alto desorden, implica que las lecturas de transconductancia no definan una tendencia clara o lo que es lo mismo; la distribución de probabilidad de sus mediciones tiende a ser uniforme. En el análisis por medición de complejidad no existen respuestas determinísticas, pues se enfrenta a un fenómeno afectado por

factores no controlables, por tanto es más natural presentar la clasificación de sus resultados en función de una medida de aleatoriedad. El algoritmo no necesita de entrenamiento, puede ser clasificado dentro de los métodos no supervisados como un método de rangos. Permite distinguir puntos de cambio en la concentración de los gases y la separación de los estatus (clases) o su falta de ella.

5. Conclusiones

La aplicación del concepto de medición de complejidad en la identificación y clasificación de gases es propiamente un reflejo de comportamiento transitivo que tienen las lecturas de transconductancia en los sensores ante los cambios que se presentan en la detección de gases. Este método es rápido e independiente de la distribución estadística de los datos por lo tanto no depende de la elección de un modelo para el proceso de clasificación. En el aspecto de alta dimensionalidad existe una reducción de esta característica al mapear los datos a un conjunto de eventos finitos junto con sus frecuecias. Puede visualizarse el grado de separación o traslapamiento de las clases a identificar, pero no ofrece una respuesta definitiva de clasificación.

Referencias

1. K. Arshak, E. Moore, *et al.*: A review of gas sensors employed in electronic nose applications. Emerald Group Publishing. Sensor Review (2004)
2. Leonardo Tomaseli Duarte: Design of Smart Chemical Sensor Array: an Approach Based on Source Separation Methods. Institut Polytechnique de Grenoble (2010)
3. Javier G. Monroy, Javier González-Jiménez, Jose Luis Blanco: Overcoming the Slow Recovery of MOX Gas Sensors through a System Modeling Approach. Open Access Sensors (2012)
4. Kok Seng Eu, Kian Meng Yap: Overcoming Long Recovery Time of Metal-Oxide Gas Sensor with Certain Factor Sensing Algorithm. In: Proceedings of the 8th International Conference on Sensing Technology (2014)
5. Ayoub Einollahi: Selectivity Enhancement for Temperature Modulated Electronic Nose Using Phase Space and Dynamic Moments. Orebro University (2012)
6. X. Rosalind Wang, Joseph T. Lizier *et al.*: Feature Selection for Chemical Sensor Arrays Using Mutual Information. PLOS one Open Acces (2014)
7. Thomas Nowotny, Amalia Z. Berna, *et al.*: Feature selection in Enose applications. PLOS one Open Acces (2014)
8. V. E. Bochenkov, G. B. Sergeev: Sensitivity, Selectivity, and Stability of Gas-Sensitive Metal-Oxide Nanostructures. American Scientific Publishers (2010)
9. Javier G. Monroy, Achim Lllienthal, *et al.*: Calibration of MOX gas sensors in open sampling systems based on Gaussian Processes. (2011)
10. Hasim Alam, S. Hasan Saeed: Subharmonic solutions with prescribed minimal Modern Applications of Electronic Nose: A review. International Journnal of Electricall Computer Enginnering (2013)
11. X. Rosalind Wang, Joseph T. Lizier *et al.*: Human breath-print identification by E-nose, using information-theoretic feature selection prior to classification. Sensors and Actuators B: Chemical, Elsevier (2014)

12. Amine Bermak, Sofiane Brahim Belhouari, *et al.*: Patttern Recognition Techniques for Odor Discrimination in Gas Sensor Array. American Scientific Publishers (2006)
13. Ricardo Gutierrez-Osuna: Pattern Analysis for Machine Olfaction: A review. IEEE Sensors Journal (2002)
14. Sepideh Pashami, Achim J. Lllienthal, Marco Trincavelli: Detecting Changes of a Distant Gas with an Array of MOX Gas. Open Access Sensors (2012)
15. Evor L. Hines, *et al.*: Pattern Analysis for Electronic Noses. Handbook of Machine Olfaction: Electronic Nose Technology. Edited by T.C. Pearce, S.S. Schiffman, H.T. Nagle, J.W. Gardner (2003)
16. Martin Lngkvist, Lars Karlsson, Amy Loutfy: A review of unsupervised feature learning and deep learning for time-series modeling. Pattern Recognition Letters, Elsevier (2013)
17. C. E. Shannon: A Mathematical Theory of Communication. The Bell System Technical Journal, Vol. 27, pp. 379-423, 623-656 (1948)
18. Alexander Vergara, Mehmet K. *et al.*: Kullback-Leibler distance optimization for artificial chemo-sensors. IEEE Sensors Journal (2009)
19. Jordi Fonollosa, Irene Rodríguez-Lujan *et al.*: Chemical Discriminatio in Turbulence Gas Mixtures with MOX Sensors Validated by Gas Chromotography-Mass Spectrometry. Open Access Sensors (2014)
20. J. S. Shiner, Matt Davison: Simple measure for complexity. The American Physical Society, Physical Review, Volume 59, number 2 (1999)

Optimización mediante algoritmo de hormigas aplicado a la recolección de residuos sólidos en UNAM-CU

Elizabeth Mancera-Galván, Beatriz A. Garro-Licón, Katya Rodríguez-Vázquez

IIMAS-UNAM, Ciudad Universitaria, D.F.,
México

elizabethal_20@hotmail.com,
{beatriz.garro, katya.rodriguez}@iimas.unam.mx

Resumen. En este artículo se aplica la metaheurística de colonia de hormigas (ACO) para resolver el problema de ruteo que se presenta al realizar la tarea de recolección de residuos sólidos en UNAM-CU. Este espacio está dividido en varios circuitos de los cuales el circuito CCU será nuestro primer caso de estudio. El encontrar la mejor ruta para este circuito, puede verse como un problema de optimización, que en una primera etapa se plantea como el problema clásico del agente viajero (TSP). Para resolver el problema, cuatro algoritmos ACO son utilizados: Sistema de hormigas (AS), Sistema de hormigas elitista (EAS), Sistema de hormigas Max-Min (MMAS) y Sistema de colonia de hormigas (ACS). Los resultados muestran reducción en la distancia recorrida con respecto a la ruta actualmente adoptada empíricamente en CU así como el desempeño de los algoritmos al realizar un estudio previo de sensibilidad de los parámetros.

Palabras clave: optimización por colonia de hormigas, ruteo de vehículos, TSP, recolección de residuos sólidos.

1. Introducción

Actualmente, la aplicación de modelos matemáticos a problemas reales de ruteo ha cobrado gran importancia debido al impacto positivo reflejado en costos, tiempo y calidad de servicio. Dentro de estos problemas reales, podemos encontrar la optimización de sistemas de recolección de residuos en zonas urbanas [10, 14, 15], cuyos objetivos principales son la reducción del uso de vehículos para el transporte, minimizar la distancia recorrida y cubrir un servicio a un cliente, donde se maximicen ganancias y el tiempo empleado sea mínimo.

La recolección de residuos, puede efectuarse por diferentes métodos, entre los que destacan por contenedores o aceras. El primer caso se plantea como un problema de ruteo por nodos (VRP). Mientras que el segundo trata el problema como ruteo de arcos (CARP). En el caso del VRP con un solo vehículo el problema se reduce al clásico problema del agente viajero (TSP) [2, 8] y en el caso del CARP, este puede ser planteado como el problema del cartero chino [6]. Este último, considera un carte-

ro cuya tarea es entregar la correspondencia en un vecindario. Para lograrlo, es necesario que parta de la oficina de correros, recorra todas las calles (arcos) y regrese a dicha oficina, de tal manera que camine la menor distancia posible sin repetir arcos y con la posibilidad de pasar por un mismo nodo varias veces (intersecciones entre arcos).

Para resolver el problema de recolección de residuos, se han utilizado métodos exactos [7, 12, 13], no obstante, la aplicación de estos métodos es limitada pues problemas complejos como el VRP y el TSP, pueden volverse NP-duros. En este caso, los métodos exactos ya no encuentran una solución óptima en un tiempo razonable. Es por esto, que se acude a la aplicación de técnicas más eficientes como las heurísticas, que si bien, no siempre encuentran las soluciones óptimas, son capaces de resolver el problema en un tiempo polinomial y de no alcanzar el óptimo, las soluciones encontradas son buenas aproximaciones al mismo.

En la literatura, el problema de recolección de residuos ha sido estudiado y abordado desde las dos perspectivas mencionadas anteriormente (VRP y CARP). Algunos trabajos publicados que pueden ser citados son: el de Thierry Kulcar [12], en cuyo trabajo se representa el problema de recolección de residuos visto como un CARP. El objetivo de este problema es minimizar los costos de transporte de residuos en Bruselas, al utilizar un método de ramificación y acotamiento. Por otra parte, Bonomo *et al.* [1] presentan un modelo de TSP para diseñar las rutas de recolección en el sur de Buenos Aires. Hornig y Fuentealba [8], aplican un algoritmo ACO a la recolección de residuos por contenedores, en un municipio de Chile. Por último, Karadimas *et al.* [11], también aplican un algoritmo ACO para establecer rutas de recolección en una zona de Grecia.

En este trabajo, se utiliza la metaheurística ACO para resolver un problema de recolección de residuos sólidos, visto como un TSP; donde el objetivo, es minimizar la distancia recorrida por los vehículos. Nuestro caso de estudio se enfoca al ruteo de vehículos al recolectar residuos sólidos en UNAM-CU (Universidad Nacional Autónoma de México-Ciudad Universitaria). Este problema en la actualidad, está resuelto de manera empírica, lo cual impide que la recolección de residuos se realice de manera eficiente en tiempo y costo para la institución. Este trabajo resulta importante debido a la aplicación a un problema real, en donde no han sido utilizadas técnicas de optimización para resolver un problema de ruteo tan importante en la UNAM-CU. Cabe destacar que un problema de recolección de residuos mal planeado genera problemas de tránsito, de insalubridad, elevados costos monetarios y de tiempo. Por este motivo, se propuso investigar sobre el tema y generar una solución a la planificación de dicho ruteo en la UNAM-CU.

Este artículo se encuentra organizado de la siguiente manera: en la Sección 2 se presenta a detalle el caso de estudio a resolver, mientras que en la Sección 3 se presentan los diferentes algoritmos ACO que serán aplicados al TSP. En la Sección 4, se presentan los resultados obtenidos al realizar un análisis de sensibilidad de los parámetros de los algoritmos y la aplicación de los mismos al resolver el problema dado. Finalmente, las conclusiones y trabajos futuros son descritos en la Sección 5.

2. Descripción del problema y formulación matemática

Ciudad Universitaria (CU), uno de los centros educativos más importantes de México, se localiza en el sur de la ciudad de México y cuenta con 730 hectáreas de superficie. La Dirección General de Conservación y Obras (DGOC) de la UNAM calculó que en promedio en CU, se generan 15 toneladas de basura con una población que es superior a las 150,000 personas. Esto sugiere que en promedio se generan diarios 0.1 Kg de residuos por persona.

El problema de recolección de residuos sólidos en CU, es el principal caso de estudio en este trabajo debido a la necesidad diaria de mantener en condiciones salubres la Universidad y reducir los costos que esto genera. Este caso, puede ser resuelto si se considera como un problema de optimización combinatoria, donde se busca obtener la mínima distancia recorrida en la ruta que abarca un número considerable de puntos de recolección y rutas a seguir por los vehículos.

Por políticas previamente establecidas, la zona de CU se encuentra dividida en dos áreas (CCU y ZN), en cada una de ellas existe un depósito donde los camiones inician y finalizan su ruta. Para propósitos de este artículo, se trabajará solamente con el área CCU. Sin embargo, en un próximo trabajo se pretende incorporar el resto de la zona, para ser resuelto como un problema asimétrico de VRP, utilizando más de un vehículo con capacidades limitadas y con otras restricciones.

En CU la Dirección General de Obras y Conservación, un organismo interno propio de la universidad, es el encargado de coordinar la recolección de residuos sólidos. Los datos sobre la localización de los puntos de recolección, así como la ruta y el mapa de la zona que más adelante se muestran, fueron obtenidos de dicho organismo [9].

Actualmente, en CCU operan con un solo vehículo y en total se localizan 31 puntos de recolección que se encuentran distribuidos como se muestra en la Figura 1. Los puntos de recolección 25, 26, 27 y 28 han sido agrupados en uno solo, pues los residuos de los puntos 26, 27 y 28 se concentran en el 25, previo a la recolección. La distancia recorrida diariamente por el vehículo de la DGOC en esta zona es de 34,132.37 m, siguiendo la numeración consecutiva de los diferentes puntos.

El camión recolector trabaja dos turnos: en la mañana y en la tarde; y cada recorrido se lleva a cabo siguiendo el orden de los puntos de recolección. Cabe mencionar, que antes de regresar al depósito, el camión debe ir a dejar los residuos a un depósito del gobierno del Distrito Federal, conocido como "Estación de transferencia". Esta estación se ubica lejos de los límites de CU, pero en el mapa queda representada cerca de las inmediaciones de CU sólo para su visualización.

En este trabajo, el problema de recolección de residuos se plantea como un TSP, en donde el vehículo partirá del depósito, recorrerá todos los puntos de recolección sólo una vez y regresará al depósito después de haber ido a la estación de transferencia.

Formalmente, el problema puede ser representado por un grafo completo, dirigido y asimétrico G=(V,A), donde V={0,1,…,n+1} es el conjunto de vértices (puntos de recolección) y A es el conjunto de arcos (trayectorias entre puntos de recolección). El vértice D={0} representa el depósito, los vértices V'={1,2,..,n} representan los puntos de recolección, y el vértice I={n+1}, representa la estación de transferencia. Por otro

lado, a cada arco se le asocia un costo c_{ij} no negativo, el cual representa la distancia entre los puntos de recolección *i* y *j*.

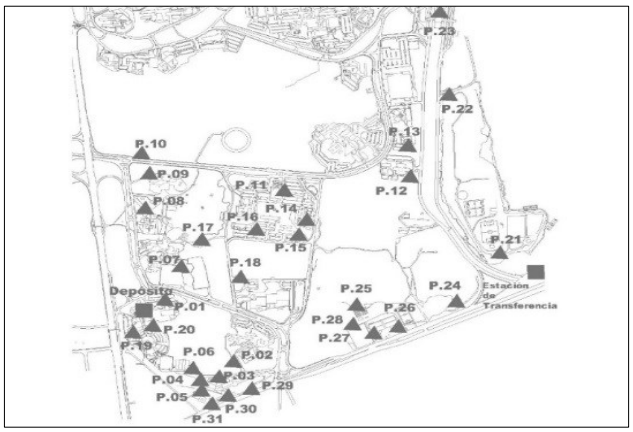


Fig. 1. Puntos de recolección en CCU.

El modelo matemático para nuestro problema de ruteo, el cual se basa en el clásico TSP, se presenta a continuación. Donde x_{ij} es una variable binaria que toma el valor de uno si el arco (i, j) se encuentra en el tour óptimo y cero en otro caso.

$$Min \sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij} \tag{1}$$

Sujeto a

$$\sum_{\substack{i \in V \\ i \neq j}} x_{ij} = 1 \qquad \forall j \in V \tag{2}$$

$$\sum_{\substack{j \in V \\ j \neq \mathrm{i}}} x_{ij} = 1 \qquad \forall i \in V \tag{3}$$

$$x_{n+1,0} = 1 \tag{4}$$

$$x_{i0} = 0 \qquad \forall i \in V' \tag{5}$$

$$\sum_{i \in S} \sum_{j \in S} x_{ij} \leq |S| - 1 \quad \forall S \subseteq V \backslash \{0\}, |S| \geq 2 \tag{6}$$

$$x_{ij} \in \{0,1\} \qquad \forall i,j \in V \tag{7}$$

La Ecuación (1) representa la función objetivo, la cual consiste en la minimización de la distancia total de la suma de los arcos utilizados en el tour. La restricción (2) señala que todos los puntos de recolección deben ser visitados exactamente una vez, mientras que la restricción (3) indica que el vehículo debe salir del punto que visitó. En la restricción (4) se señala que después de que el vehículo haya ido a la estación de transferencia, sólo podrá dirigirse al depósito. La restricción (5) expresa que el vehículo no puede regresar al depósito después de haber visitado un punto de recolección. La restricción (6) asegura la continuidad del tour, es decir, que el tour sea conexo y así evitar la generación de pequeños sub-tours. En (7) se especifica que la variable x_{ij} sólo puede tomar los valores uno o cero.

Una vez identificado el problema que queremos resolver, procedemos a explicar en la siguiente Sección, qué algoritmos se utilizan para dar solución al problema de recolección de residuos en CCU.

3. Algoritmos ACO

Los algoritmos basados en la optimización por colonia de hormigas (Ant Colony Optimization-ACO), se encuentran dentro de las metaheurísticas denominadas como Inteligencia de Enjambres. Este tipo de metaheurísticas, simulan el comportamiento social de enjambres de insectos para resolver problemas de optimización. La metaheurística ACO simula el comportamiento de las hormigas reales cuando éstas se encuentran buscando comida. Específicamente, las hormigas son capaces de encontrar el camino más corto entre el nido y la fuente de alimento mediante un tipo de comunicación indirecta, la cual se basa en seguir un rastro de feromona que es depositado por cada hormiga a su paso y reforzado a su regreso al nido [2]. La simulación en ACO sobre el comportamiento de las hormigas reales es mediante el uso de hormigas artificiales, las cuales son capaces de aprender sobre el espacio de búsqueda durante la ejecución del algoritmo y usan esta experiencia adquirida para construir, mejores soluciones en cada iteración. Este proceso de construcción puede entenderse como una toma secuencial de decisiones regida por una regla de transición estocástica.

Parte esencial de los algoritmos ACO, es la combinación de información heurística y el rastro de feromona. La información heurística, también llamada *visibilidad*, mide lo deseable que es un nodo *j* para ser visitado desde *i*, con base en la información a priori del problema. Vale la pena mencionar que, esta información no es modificada por las hormigas. En cuanto al rastro de feromona, intuitivamente se puede decir que mide qué tan deseable es un nodo con base en lo que han aprendido las hormigas. En este caso, el rastro de feromona es modificado por dichos insectos virtuales.

El parámetro τ refleja el rastro de feromona, mientras que η representa la información heurística (visibilidad). Ambos parámetros son utilizados en el algoritmo para calcular la probabilidad que permitirá a cada hormiga decidir cómo moverse a través del grafo.

La metaheurística ACO fue introducida por Marco Dorigo [5] y a partir de dicho desarrollo se han generado diversas propuestas que buscan mejorar el funcionamiento del algoritmo original. A continuación, se describe el algoritmo básico llamado Sistema de hormigas (Ant System-AS) y tres de sus variantes: Sistema de hormigas elitista (Elitist Ant System-EAS), Sistema de hormigas Max-Min (Max-Min Ant system-MMAS) y Sistema de colonia de hormigas (Ant Colony System-ACS).

3.1 Sistema de hormigas (AS)

El primer algoritmo propuesto dentro de la metaheurística ACO fue el Sistema de hormigas [4, 5]. De manera general, en el algoritmo AS, se posiciona un número de hormigas k en cada nodo. Cada hormiga construye una solución factible al problema, al aplicar de manera iterada la siguiente regla de transición que combina τ y η para elegir con cierta probabilidad p_{ij} la siguiente ciudad *j* a visitar desde la ciudad *i*.

$$p_{ij}^{k} = \begin{cases} \dfrac{\tau_{ij}^{\alpha}\eta_{ij}^{\beta}}{\sum_{u\epsilon \mathrm{N}_i^k} \tau_{iu}^{\alpha}\eta_{iu}^{\beta}} & si\ \ j\epsilon \mathrm{N}_i^k \\ 0 & en\ otro\ caso \end{cases} \quad (\mathbf{8})$$

Donde los parámetros α y β, determinan la importancia de la feromona y la información heurística, respectivamente. Por otro lado η_{ij}, es el recíproco de la distancia entre *i* y *j*.

Una vez que todas las hormigas han construido un tour completo, actualizan la feromona, de acuerdo a la Ecuación (9).

$$\tau_{ij} = (1-\rho)\tau_{ij} + \sum_{k=1}^{m} \Delta\tau_{ij}^{k} \quad (\mathbf{9})$$

Donde ρ es la tasa de evaporación, *m* es el número de hormigas, y $\Delta\tau_{ij}^{k}$ es la cantidad de feromona depositada en el arco (*i,j*) por la hormiga *k*. $\Delta\tau_{ij}^{k}$ y se calcula como Q/L_k, con L_k que representa la longitud del tour construido si (*i,j*) pertenece al tour hecho por la hormiga *k* y es cero en otro caso.

3.2 Sistema de hormigas elitista (EAS)

Esta propuesta es una variante del algoritmo original AS [2], en la que se busca dar un peso adicional a la mejor solución conocida hasta el momento s_{bs} (mejor solución global) en la actualización de la feromona. Esta solución es generada por una hormiga, a la cual se le llama *hormiga elitista*. Para construir las soluciones en EAS se sigue la Ecuación (8) como en el algoritmo AS. Para actualizar la feromona, se agrega una cantidad ($\Delta\tau_{ij}^{bs} = \frac{1}{L_{best}}$) de feromona extra en los arcos que pertenecen a la mejor solución global generada por la hormiga elitista. La actualización de feromona modificada es la siguiente:

$$\tau_{ij} = (1-\rho)\tau_{ij} + \sum_{k=1}^{m} \Delta\tau_{ij}^{k} + e\Delta\tau_{ij}^{s_{bs}} \quad (\mathbf{10})$$

donde e es un parámetro que pondera la influencia de la mejor solución global.

3.3 Sistema de hormigas Max-Min (MMAS)

El algoritmo sistema de hormigas Max-Min [16] toma como base el algoritmo AS, pues utiliza la misma regla de construcción de soluciones (Ecuación (8)). Sin embargo, se añaden tres modificaciones al algoritmo básico, las cuales se generan tomando en cuenta que en el MMAS se busca explotar la mejor solución global o la mejor solución encontrada en la iteración actual. Las modificaciones son: el rastro de feromona se encuentra acotado inferior y superiormente; el valor inicial de feromona queda definido como el límite superior del mismo y por último, los niveles de feromona son reinicializados si durante cierto número de iteraciones, la solución no mejora. La actualización de feromona queda definida por la Ecuación (11).

$$\tau_{ij} = \left[(1-\rho)\tau_{ij} + \Delta\tau_{ij}^{s_{bs}}\right]_{\tau_{min}}^{\tau_{max}} \quad (\mathbf{11})$$

donde τ_{max} y τ_{min}, son el límite superior e inferior de los niveles de feromona respectivamente. Para calcularlos, usamos las ecuaciones presentadas en [16].

3.4 Sistema de colonia de hormigas (ACS)

El algoritmo sistema de colonia de hormigas [3] es la variante que más difiere del AS, al integrar tres cambios. En primer lugar, para construir soluciones las hormigas utilizan la Ecuación (12). Donde *j* es el siguiente nodo a visitar, q es una variable aleatoria uniforme en [0,1], q_0 es un parámetro entre [0,1]. J denota el nodo a elegir siguiendo la Ecuación (8).

$$\mathrm{j} = \begin{cases} \arg max_{\mathrm{l}\epsilon \mathrm{N}_i^k} \left({\tau_{ij}}^{\alpha} \eta_{ij}^{\beta}\right) & si \;\; q \leq q_0 \\ J & en\ otro\ caso \end{cases} \quad (\mathbf{12})$$

Como segunda y tercera modificación, en el algoritmo ACS, la actualización de feromona queda dividida en dos procesos: uno global y uno local. En la actualización local todas las hormigas modifican el nivel de feromona cada vez que atraviesan un arco. Esta regla (Ecuación (13)), conocida como regla de actualización local de feromona, está determinada por:

$$\tau_{ij} = (1-\varphi)\tau_{ij} + \varphi\tau_0 \quad (\mathbf{13})$$

donde φ es un coeficiente de decaimiento que toma valores en (0,1], y τ_0 es el valor inicial de feromona.

Para realizar la actualización global, se toma en cuenta una sola hormiga, la que generó la mejor solución global. La Ecuación (14) es utilizada para añadir la feromona a los arcos, después de cada iteración.

$$\tau_{ij} = \begin{cases} (1-\rho)\tau_{ij} + \rho\Delta\tau_{ij}^{best} & si\ \ (i,j)\ está\ en\ el\ mejor\ tour \\ \tau_{ij} & en\ otro\ caso \end{cases} \quad (14)$$

4. Resultados experimentales

Diversos experimentos se llevaron a cabo para validar la solución propuesta utilizando las distancias entre todos los posibles puntos de recolección.

Para definir los valores de los parámetros utilizados en los algoritmos ACO, se tomaron como base los siguientes: para el algoritmo de AS los parámetros fueron *m*=10, α=1, β=3, $\tau_o = 1/L_{nn}$[1], ρ=0.1, Q=1, para el EAS además de los anteriores se incluye *e*=1. Por otra parte, para el algoritmo ACS, se utilizan también los parámetros φ =0.1 y $q_0 = 0.9$. Por último, en el MMAS el mecanismo de reinicialización es activado si después de 250 iteraciones la solución no mejora. Estos parámetros fueron elegidos porque son de los más utilizados en la literatura y aunque en ACS estos valores varían, a partir de los mismos parámetros es posible comparar el comportamiento de los algoritmos. Como se verá en la discusión de resultados, fue necesario evaluar la influencia de cada parámetro tomando diferentes valores para cada uno dentro de sus rangos establecidos.

El número de iteraciones realizadas fueron 1000 para todos los algoritmos y se realizaron 30 corridas para cada uno. Dado que los algoritmos ACO son heurísticos y dependen de los valores iniciales para realizar su búsqueda de soluciones, los resultados pueden otorgar valores diferentes en cada una de las corridas del algoritmo. Por lo que, para validar el resultado de los mismos es necesario realizar vari0s experimentos.

En la Tabla 1, se muestran los resultados obtenidos con los parámetros ya especificados anteriormente. Se puede observar que los cuatro algoritmos llegan a la misma solución (31742 mts). Esta distancia resulta menor al ser comparada con la distancia recorrida actualmente (generada empíricamente) en CCU por vehículos de la DGOC, la cual es de 34132.37 mts. Por otro lado, se aplicó la heurística del vecino más cercano para comparar los resultados obtenidos con los algoritmos ACO. La distancia encontrada por esta heurística fue de 35746.3 mts.

A pesar de que todos los algoritmos arrojan la misma solución, el MMAS presenta una mayor estabilidad. Esto puede observase en el promedio de las soluciones ya que para el MMAS el valor es menor entre los cuatro algoritmos así como la desviación estándar obtenidas en las 30 corridas. Adicionalmente, el número de iteraciones promedio para encontrar la mejor solución resulta menor comparado con el AS, EAS y ACS. Otro factor que influye para determinar que la estabilidad del algoritmo MMAS es la desviación estándar del número de iteraciones ya que es menor. Esto se traduce en que, para el MMAS se necesitan entre 341 y 602 iteraciones para encontrar el óp-

[1] L_{nn} es la longitud del tour mínimo encontrado utilizando la técnica de vecino más cercano.

timo, mientras que para los tres algoritmos restantes, la solución óptima podría encontrarse después de las 800 iteraciones.

Tabla 1. Resultados.

Parámetro	Algoritmo	Mejor solución (B)	Promedio soluciones (P)	Peor Solución (W)	Desv. Estándar (S)	Prom. Iter. (PI)	Desv. Estándar Iter. (SI)
Base	AS	31742.0	32179.0	32822.1	393.0	537.7	310.7
	EAS	31742.0	31905.1	33129.2	308.4	510.6	330.3
	MMAS	31742.0	31899.7	32904.8	202.6	471.7	130.3
	ACS	31742.0	32340.4	33471.0	602.0	591.7	233.8
β=0.5	AS	32380.6	33340.3	34196.5	469.4	671.8	207.5
	EAS	31742.0	32090.0	32865.0	343.6	727.0	252.0
	MMAS	31742.0	32616.8	33658.4	444.7	684.9	198.9
	ACS	31912.7	33435.7	34848.0	667.8	138.7	25.6
β=1	AS	31742.0	32256.5	32683.8	348.7	456.2	278.5
	EAS	31742.0	31814.1	32604.0	172.2	507.4	226.1
	MMAS	31742.0	31805.4	32394.8	150.3	597.0	144.8
	ACS	31742.0	32202.7	33043.9	481.3	458.2	276.4
β=5	AS	31742.0	32738.9	33416.0	391.7	348.4	322.2
	EAS	31742.0	32311.7	33554.3	586.0	543.6	332.6
	MMAS	31742.0	31997.0	33129.2	297.8	569.9	215.2
	ACS	317420	32548.6	33605.1	439.8	600.4	248.9
β=7	AS	32474.3	33089.8	34152.4	447.0	312.7	281.4
	EAS	31742.0	32955.5	34447.7	576.8	432.6	274.0
	MMAS	30993.5	31837.7	32552.3	377.1	534.5	237.5
	ACS	31762.0	32971.7	34200.0	669.5	261.8	200.3
m=5	AS	31880.3	32570.5	33264.7	344.8	612.4	267.6
	EAS	31742.0	32471.7	33947.5	592.7	549.4	336.3
	MMAS	31742.0	32716.7	34150.7	884.8	518.4	315.1
	ACS	31880.3	32833.8	34208.9	601.5	133.0	70.9
m=20	AS	317420	32004.2	32602.1	236.0	392.8	300.7
	EAS	31742.0	31849.4	31900.3	58.8	367.1	251.2
	MMAS	31742.0	31802.8	31968.9	73.0	474.7	113.6
	ACS	31742.0	32175.4	33605.1	588.1	403.8	255.5
m=25	AS	31880.3	32003.7	32602.1	245.6	391.4	298.5
	EAS	31742.0	31848.0	31880.3	59.5	311.4	270.9
	MMAS	31742.0	31909.3	33146.2	350.1	572.8	257.9
	ACS	31742.0	32176.8	33159.9	498.8	84.9	18.5
m=29	AS	31742.0	32011.9	32632.6	248.0	285.6	275.9
	EAS	31742.0	31844.1	31900.3	62.7	389.0	295.8
	MMAS	31742.0	31756.3	31880.3	38.8	446.4	86.4
	ACS	31742.0	31932.4	32990.9	338.2	570.8	321.2

Parámetro	Algo-ritmo	Mejor solución (B)	Promedio soluciones (P)	Peor Solución (W)	Desv. Estándar (S)	Prom. Iter. (PI)	Desv. Estándar Iter. (SI)
ρ=0.01	AS	31742.0	32095.2	32778.4	306.6	500.4	289.1
	EAS	31742.0	31918.1	32802.9	252.7	511.0	276.7
	MMAS	31742.0	32494.0	33397.5	512.3	112.2	103.2
	ACS	31742.0	32114.6	32891.8	353.0	535.2	287.9
ρ=0.3	AS	31762.0	32427.5	33129.2	389.2	411.9	355.3
	EAS	31742.0	32103.2	33004.3	411.5	476.9	270.0
	MMAS	32612.5	34116.4	36825.3	949.6	30.3	11.6
	ACS	31742.0	32173.1	33872.8	597.4	432.1	254.6
ρ=0.5	AS	31880.3	32552.0	32961.4	310.3	391.3	287.0
	EAS	31742.0	32419.9	33265.3	504.1	365.0	284.6
	MMAS	31742.0	32306.8	33972.0	700.3	580.7	296.3
	ACS	33212.7	35284.5	38193.9	1323.3	18.3	8.7
ρ=0.7	AS	31742.0	32728.3	33221.1	392.5	412.4	307.4
	EAS	31742.0	32342.3	32994.4	480.2	565.8	313.8
	MMAS	31742.0	32403.6	33783.9	652.3	450.4	321.7
	ACS	32691.5	36406.6	40255.8	1822.6	10.6	4.8
e=3	EAS	31742.0	32135.9	33649.5	473.7	297.5	240.2
e=5	EAS	31742.0	32097.1	33254.0	493.7	403.9	287.3
e=7	EAS	31742.0	32399.9	33397.5	561.7	398.2	306.1
e=10	EAS	31742.0	32597.7	34057.6	586.4	380.5	333.8
Sin reini-cialización	MMAS	31742.0	31969.3	32990.9	276.2	521.6	163.8
S^{bs}	MMAS	31742.0	32530.1	34031.1	684.4	280.8	95.8
φ=0.01	ACS	31742.0	32428.0	34529.2	876.5	539.4	308.5
φ=0.3	ACS	31742.0	31902.4	32474.3	248.8	663.8	229.2
φ=0.5	ACS	31742.0	32092.9	33383.6	427.5	632.9	258.9
φ=0.7	ACS	31742.0	32151.5	33272.0	383.0	595.9	238.3
q_0=0.3	ACS	31742.0	31991.7	33405.7	478.4	710.8	258.4
q_0=0.5	ACS	31742.0	32158.4	33146.2	541.4	574.7	242.8
q_0=0.7	ACS	31742.0	32070.6	33129.2	446.6	557.7	217.2
q_0=0.95	ACS	31742.0	32177.5	33720.3	624.5	574.5	266.2

4.1 Discusión de los resultados

Una forma de entender mejor el comportamiento de los algoritmos ACO, es mediante el análisis de sensibilidad aplicado a los parámetros de dichos algoritmos. En este trabajo los parámetros tomados en cuenta para llevar a cabo el análisis de sensibilidad, son los siguientes: m, β, ρ, e, φ y q_0, según el algoritmo. Es importante mencionar que si uno de los parámetros es modificado, los demás permanecen constantes, con los valores definidos previamente. Todo el análisis se basa en los resultados mostrados en la Tabla 1.

Dado que casi siempre se llega a la solución óptima, nos centraremos en revisar el promedio y desviación estándar de las soluciones e iteraciones. La razón de tal revisión es que si bien, con todos los algoritmos encontramos el óptimo en este problema, es importante comparar el tiempo que tardan en converger a la solución. Esto se puede analizar al observar el número de iteraciones que se necesitan para alcanzar los mejores resultados. De manera similar, es deseable que en las 30 corridas, las soluciones obtenidas no difieran en gran cantidad, lo cual se puede observar con el promedio y desviación estándar de las soluciones.

De acuerdo a β en todos los algoritmos, a excepción del AS, existe una mejora del promedio de las soluciones (P) cuando β=3 (ver Figura 2), y empeoran cuando β aumenta o disminuye; comportamiento que también presenta la desviación estándar del promedio (S). Cuando β=7, los algoritmos MMAS, AS y ACS no encuentran la mejor solución (31742.0). Por otro lado, aunque el promedio de iteraciones (PI) no siempre es menor cuando β=3; al comparar con los demás valores, se observa que conforme disminuye dicho promedio, el promedio de las soluciones aumenta. Dado lo anterior, podemos decir que para los algoritmos EAS, MMAS y ACS un valor de β=3, permite que las hormigas encuentren mejores soluciones.

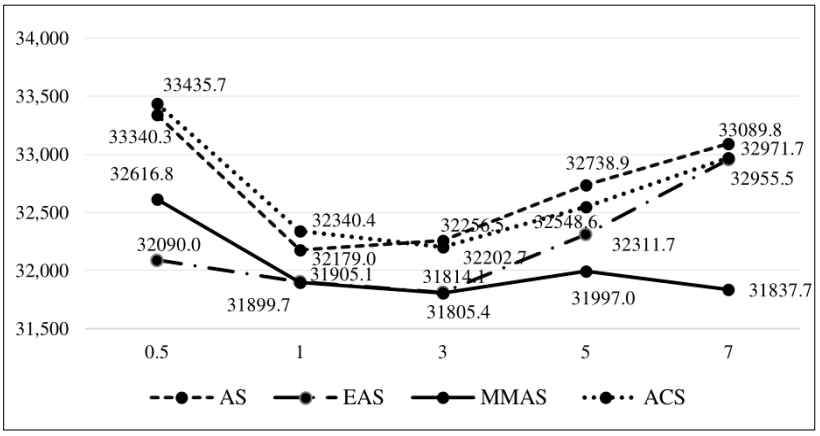


Fig. 2. Análisis del promedio de soluciones para β.

De manera general, para el parámetro *m* (número de hormigas), utilizar más hormigas genera cambios favorables en el promedio de las soluciones (ver Figura 3). Si se utilizan 29 hormigas (igual al número de puntos de recolección) P disminuye en todos los algoritmos, con respecto a los resultados obtenidos cuando *m*=10. Este hecho tiene mayor impacto en el algoritmo ACS. Por otra parte, cuando se utilizan 20 o 25 hormigas, las diferencias con respecto a utilizar 29 son muy pequeñas, excepto para ACS. Sin embargo, el PI para ACS y EAS aumenta si *m*=29. Podemos concluir, para este trabajo, que utilizar un número cercano al número de puntos de recolección mejora las soluciones obtenidas y, en el caso del MMAS y AS, permite que los algoritmos tengan una rápida convergencia.

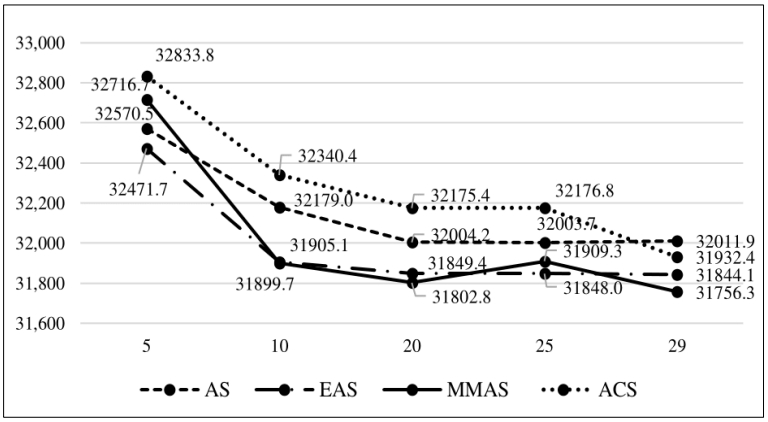


Fig. 3. Análisis del promedio de soluciones para *m*.

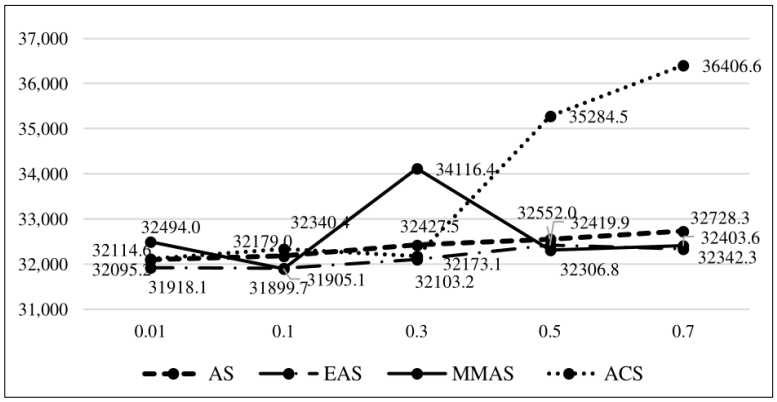


Fig. 4. Análisis del promedio de soluciones para ρ.

Si comparamos el comportamiento de los cuatro algoritmos, al modificar los valores de ρ, es interesante que para EAS y MMAS aumentar o disminuir el valor de dicho parámetro repercute negativamente en P en especial para el algoritmo Max-Min (ver Figura 4). Otra cuestión importante, se observa en PI, pues para el algoritmo MMAS el promedio de iteraciones disminuye considerablemente cuando ρ=0.01 o ρ=0.3, es decir, el algoritmo se estanca rápidamente en una mala solución sin encontrar la mejor. Dado lo anterior, podemos decir que el valor de ρ=0.1, genera las mejores soluciones para EAS y MMAS. Por otra parte, al disminuir el valor de ρ a 0.01, AS y ACS mejoran el promedio de las soluciones al igual que el tiempo de convergencia a la solución. Sin embargo, al aumentar el valor de dicho parámetro, el promedio de las soluciones empeora, comportamiento que también presenta la desviación estándar del promedio (S). Más aun, con tales valores, no se encuentra la mejor solución. Al observar la convergencia del ACS, se puede inferir que el algoritmo se estanca rápidamente en una mala solución. Por lo tanto, para AS y ACS, parece ser más conveniente utilizar un parámetro ρ=0.01.

De manera particular, para el parámetro *e* (solamente utilizado en EAS), se observa que el número de iteraciones promedio disminuye al aumentar dicho valor. Sin embargo el promedio de las soluciones empeora de manera significativa. Además, la desviación estándar del promedio (S) también aumenta conforme el valor de *e* lo hace; por lo tanto, utilizar un valor de *e*=1, otorga los mejores resultados.

Enfocándonos en el mecanismo de reinicialización del MMAS, se observa que al quitarlo, el promedio de las soluciones no mejora, incluso el promedio de iteraciones aumenta. Por otro lado, utilizar la mejor solución global en lugar de la mejor solución de la iteración, tiene un efecto negativo en el promedio de soluciones y, solamente hace que el algoritmo converja más rápido. Como es mencionado en [16], para problemas con una dimensión no muy grande, es preferible utilizar la mejor solución en la iteración para actualizar la feromona. Por lo tanto, para nuestro problema utilizar el mecanismo de reinicialización, así como actualizar la cantidad de feromona mediante la mejor solución de la iteración, arroja mejores resultados con el MMAS.

Particularmente en el algoritmo ACS se utiliza el parámetro φ, que funciona como un reductor de feromona. Para nuestro trabajo, se puede observar que cuando disminuye a 0.01, el promedio de las soluciones empeora, pero si aumenta, el promedio de soluciones mejora, obteniéndose el mejor resultado al fijar el valor en 0.3. También se puede observar una relación entre el promedio de las soluciones y el promedio de iteraciones, ya que conforme el primero mejora, el segundo aumenta. Esto significa que al aumentar el valor de este parámetro a 0.3 permitimos que haya mayor exploración, es decir, ampliar la búsqueda en el espacio de soluciones lo que implica una convergencia tardía. Sin embargo, si φ rebasa el valor de 0.5, ya no se genera una mejora en el promedio de las soluciones. Dado lo anterior, se concluye que utilizar un valor φ=0.3, mejoraría los resultados obtenidos con ACS.

Otro parámetro utilizado específicamente en ACS es q_0, este parámetro permite que haya mayor exploración si este es menor. Dado los resultados que se muestran en la Tabla 1, cuando existe mayor exploración mejora el promedio en las soluciones, obteniendo el mejor resultado con $q_0 = 0.7$. Sin embargo, al utilizar dicho valor la convergencia a la solución es más lenta.

5. Conclusiones

En este artículo se resolvió un problema real de ruteo para la recolección de residuos en UNAM-CCU. Este problema fue planteado como un TSP asimétrico con un par de restricciones extras, necesarias para la adaptación del problema. Para resolverlo, utilizamos cuatro algoritmos ACO, y con todos se encontró la misma solución, lo que se debe a que el número de puntos de recolección no es muy grande. Además, con nuestra propuesta logramos reducir en 7% la distancia total recorrida actualmente de manera empírica en CCU. Los resultados generados con los diferentes algoritmos ACO fueron comparados con el algoritmo de vecino más cercano cuyo resultado indica que se llegó al mínimo recorrido al aplicar heurísticas ACO. Esta reducción resulta importante porque, además de minimizar la distancia, se tiene un efecto positivo en tiempos y costos. Por otra parte, al comparar el promedio de las soluciones obtenidas

durante todas las corridas, los mejores resultados se obtienen para el algoritmo MMAS y los peores para el AS. Aunque con el algoritmo AS las soluciones obtenidas no son tan buenas como las que proporcionan sus sucesores, también se puede mencionar que es el algoritmo ACO más sencillo de implementar y el que involucra un menor número de parámetros. Por esta razón, utilizarlo para problemas pequeños es conveniente.

Con base en el análisis de sensibilidad, podemos mencionar que los parámetros tienen diferente impacto en los resultados obtenidos con los cuatro algoritmos. De manera particular, en nuestro caso de estudio se tiene, por ejemplo, que al aumentar el número de hormigas todos los algoritmos presentan un mejor desempeño. Por otra parte, disminuir la tasa de evaporación impacta positivamente al AS y ACS y, por el contrario, afecta al MMAS y al EAS. Adicionalmente, es interesante observar que el algoritmo MMAS y EAS obtienen soluciones similares, análogamente el comportamiento que muestran al cambio en los parámetros es parecido. Para el caso de φ y q_0 se observa que el comportamiento de ACS converge en mayor tiempo cuando se aumenta la exploración con ciertos valores de los mismos. En conclusión, escoger el algoritmo a utilizar es tan importante como determinar los valores de los parámetros, ya que pequeños cambios en ellos pueden conducir a cambios significativos en los resultados finales.

Finalmente, vale la pena mencionar que a partir de este trabajo, los trabajos futuros que se desprenden de éste consiste en implementar los algoritmos de hormigas en todo el circuito CU y así, tener una propuesta completa de nuevas rutas que optimicen la distancia recorrida por los vehículos que se encargan de la recolección de residuos en dicha universidad. También es deseable que la aplicación de estas técnicas de optimización se extienda y se apliquen en la ciudad de México, así como se ha hecho en otras zonas urbanas del mundo.

Agradecimientos. Los autores agradecen el apoyo otorgado al proyecto PAPIIT con número IN107214, así como a la Dirección General de Obras y Conservación de la UNAM, a la Ing. Araceli Flores Soto y al Arq. José Guadalupe González Cruz por el apoyo brindado al aportar información sobre la planeación actual de la recolección de residuos en UNAM-CU. Finalmente, Beatriz A. Garro-Licón agradece al CONACYT por la beca posdoctoral otorgada.

Referencias

1. Bonomo, F., Durán, G., Larumbe, F., Marenco, J.: A method for optimizing waste collection using mathematical programming: a Buenos Aires case study. Waste Management & Research, vol. 30, no. 3, pp.311–324 (2012)
2. Dorigo, M., Birattari, M., Stützle, T.: Ant colony optimization-artificial ants as a computational intelligence technique. IEEE Comput. Intell. MAG, vol. 1, pp. 28–39 (2006)
3. Dorigo, M., Gambardella, L. M.: Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem. IEEE Transactions on Evolutionary Computation, vol. 1, no. 1, pp. 53–66 (1997)
4. Dorigo, M., Maniezzo, V., Colorni, A.: Ant system: an autocatalytic optimizing process. Université Libre de Bruxelles (1991)

5. Dorigo M.: Optimization, Learning and natural algorithms. Ph.D Thesis, Dip. Electtronica e Informazion, Politecnico di Milano, Italy (1992)
6. Filipiak, K. A., Abdel-Malek, L., Hsieh, H.-N., Meegoda, J. N.: Optimization of municipal solid waste collection system: case study. Practice Periodical of Hazardous, Toxic, and Radioactive Waste Management, vol. 13, no. 3, pp. 210–216 (2009)
7. Hemmelmayr, V. et al.: A heuristic solution method for node routing based solid waste collection problems. Journal of Heuristics, vol. 19, no. 2, pp. 129–156 (2013)
8. Hornig, E.S., Fuentealba, N. R.: Modelo ACO para la recolección de residuos por contenedores. Revista chilena de ingeniería, vol. 17, no. 2, pp. 236–243 (2009)
9. Secretaría Administrativa, Dirección de Obras y Conservación, http://www.obras.unam.mx/Pagina/, Actualizado: 23 de mayo de 2015.
10. Ismail, Z., Loh, S.: Ant colony optimization for solving solid waste collection scheduling problems. Journal of mathematics and statistics, vol. 5, no. 3, pp.199–205 (2009)
11. Karadimas, N. V. et al.: Optimal solid waste collection routes identified by the ant colony system algorithm. Waste Management & Research, vol. 2, no. 2, pp. 139–147 (2007)
12. Kulcar, T.: Optimizing solid waste collection in Brussels. European Journal of Operational Research, vol. 90, no. 1, pp. 71–77 (1996)
13. Mansini, R. et al.: A linear programming model for the separate refuse collection service. Computers & Operations Research, vol. 25, no. 7, pp. 659–673 (1998)
14. Mourao, M., Almeida, M. T.: Lower-bounding and heuristic methods for a refuse collection vehicle routing problem. European Journal of Operational Research, vol. 121, no. 2, pp. 420–434 (2000)
15. Ogwueleka, T. C.: Route optimization for solid waste collection: Onitsha (Nigeria) case study. Journal of Applied Sciences and Environmental Management, vol. 13, no. 2, pp. 37–40 (2009)
16. Stützle, T., Hoos, H.H.: Improving the Ant System: A detailed report on the MAX–MIN Ant System. Technical Report AIDA–96–12, F.G. Intellektik, F.B. Informatik, T.U. Darmstadt, Germany (1996)

Sintonización de un controlador Proporcional-Integral Derivativo aplicado a una celda termoeléctrica: Una comparación entre algoritmos genéticos

Juan Fernando García Mejía, Juan Carlos Suarez Sanchez, Allan Antonio Flores Fuentes, José Arturo Pérez Martínez, Carlos Eduardo Torres Reyes

Centro Universitario UAEM Atlacomulco,
Atlacomulco, México

fgarciam@uaemex.mx

Resumen. Las celdas termoeléctricas son dispositivos semiconductores que se usan en la refrigeración móvil, su comportamiento, es decir la forma que responden a ciertos estímulos eléctricos puede manipularse por medio de técnicas de ingeniería de control, la más popular, el controlador Proporcional-Integral-Derivativo cuyos parámetros son sintonizados por medio del criterio de Zigler-Nichols; el cual, en este artículo, es comparado con un algoritmo genético canónico de 40 cromosomas y con una versión reducida de este, denominado micro algoritmo genético de 5 individuos. En este trabajo ambas técnicas evolutivas son codificadas con números reales y recombinadas por medio de operadores de cruce aritmético no uniforme. Los resultados generados por los algoritmos propuestos son simulados mediante Scilab.

Palabras clave: algoritmo genético, celda termoeléctrica, micro algoritmo genético.

Abstract. The thermoelectric cells are semiconductor devices used in mobile refrigeration, their behavior, it is to say, the way they respond to certain electrical stimulus, can be manipulated through control engineering techniques, the most popular, the Proportional-Integral-Derivative controller whose parameters are tuned by the Zigler-Nichols criterion; which in this article is compared with a canonical genetic algorithm of 40 chromosomes and with a reduced version of this called genetic micro algorithm of 5 individuals. In this paper, both evolutive techniques are coded with real numbers and recombined through arithmetic crossover operators, non-uniform. Note that this proposal is simulated on Scilab, a type GNU licensed mathematic software.

Keywords: genetic algorithm, thermoelectric cooler, micro genetic algorithm

1. Introducción

Una celda termoeléctrica (TEC, por sus siglas en inglés) es un dispositivo semiconductor que permite el intercambio de temperatura entre las superficies que la

forman, lo anterior en función de la aplicación de una determinada señal eléctrica. Mientras que una superficie genera un efecto de enfriamiento, la otra disipa energía en forma de calor, esto se revierte mediante un cambio de polaridad del voltaje aplicado a las celdas. Dado lo anterior las TEC's tienen aplicación en micro refrigeración y refrigeradores móviles [1].

Existe una relación entre la temperatura de enfriamiento y la corriente eléctrica de polarización de una celda termoeléctrica, esta se caracteriza por medio de una función de transferencia en términos de la variable compleja s tal como se muestra en la ecuación 1 [2].

$$\frac{\tilde{T}_L(s)}{\tilde{I}(s)} = G_I(s) = -6.4061\left(\frac{0.064s + 0.00854}{s^2 + 0.5964s + 0.00855}\right) \quad (1)$$

donde $\tilde{I}(s)$ es la corriente de alimentación de la celda y $\tilde{T}_L(s)$ la temperatura de la celda.

La figura 1 muestra una respuesta típica de una TEC en lazo abierto con una corriente de alimentación de 1.6 ampers, la cual se puede manipular por medio de técnicas de ingeniera de control tales el controlador Proporcional Integral Derivativo (PID) , este es el más usado en el sector industrial, su función de transferencia en su implementación paralela se describe en la ecuación 2 y consiste de tres ganancias denominadas k_p, k_i, k_d.

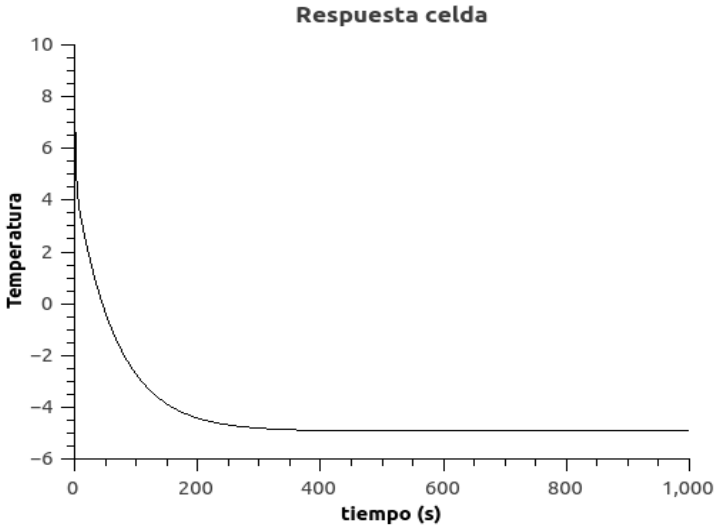


Fig. 1. Respuesta de la celda termoeléctrica.

En base a las ecuaciones 1 y 2 es posible construir un esquema de control como se muestra en la figura 2.

El ajuste de los valores k_p, k_i, k_d se realiza de manera habitual por metódos analíticos como el criterio de Zigler-Nichols, una alternativa a este se encuentra en el

uso de técnicas evolutivas, las cuales son series de pasos definidos con aplicación en la optimización con la capacidad de evolucionar [3], una de ellas es el algoritmo genético simple o canónico (GA, por sus siglas en inglés), desarrollado por John Holland en la década de los 60 en la Universidad de Michigan, en base a los principios biológicos presentes en la naturaleza descritos por Charles Darwin y Gregory Mendel [4].

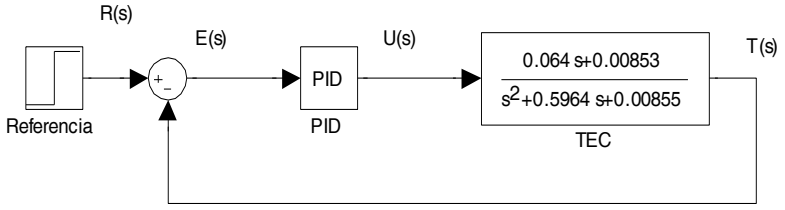


Fig. 2. Esquema de control.

$$PID = C(s) = \frac{U(s)}{E(s)} = k_p + \frac{k_i}{s} + k_d s \tag{2}$$

Los GA's han sido objeto de estudio de diversas investigaciones, las cuales tienen como resultado nuevas formas de codificación de los cromosomas, variantes de los operadores de cruzamiento y mutación y así como adaptaciones que permiten resolver múltiples objetivos.

Una de las variantes de los GAs se denomina micro algoritmo genético (μ-GA), desarrollado por Kalmanje Krishnakumar en 1989, este hace referencia a GA con población reducida (no más de 5 cromosomas) a diferencia de los 40 a 300 individuos de la versión canónica [5] lo cual permite obtener un algoritmo evolutivo de bajo costo computacional.

El uso de técnicas evolutivas y en específico de algoritmos genéticos en la sintonización de controladores PID es documentado en la literatura especializada, como muestra Mohd S et al [6] donde realizan una comparación entre evolución diferencial, algoritmos genéticos y criterio de Zigler-Nichols, demostrando que las técnicas evolutivas presentan ventajas en cuanto al desempeño con respecto a técnicas analíticas como Zigler-Nichols. Li J y otros [7] realizaron un algoritmo genético con codificación real, el cual no demuestra diferencias significativas con respecto al uso de un enjambre de partículas. Reynoso G et al [8] muestra un procedimiento de sintonización automática de parámetros de un controlador PID en forma serie, el cual muestra ventajas sobre Zigler-Nichols al controlar la respuesta de un conjunto de plantas de prueba propuestas de manera teórica. Renato A y otros [9] utiliza un algoritmo genético con codificación real para sintonizar un PID de dos grados de libertad, como planta de estudio se propone la función característica de un servomotor. Valarmathi otros [10] controla el nivel de líquido de un tanque modelado como un sistema no lineal utilizando un controlador PID. Yang M et al controlan la velocidad de rotación de un motor a través de un algoritmo genético cuyo criterio de paro es la convergencia del algoritmo.

En base a lo documentado en el estado del arte se puede aseverar que la sintonización de controladores tipo PID realizada por algoritmos genéticos ofrecen mayores ventajas que el criterio de Zigler-Nichols, [6, 8, 10], cabe destacar que la codificación empleada en los citados trabajos fue binara, esto puede presentar errores de truncamiento en el momento de representar las variables k_p, k_i, k_d; esto es solucionado por una codificación real en [7] y [9], por último en [11] se documenta el paro de algoritmo genéticos por medio de la convergencia de la función objetivo.

El uso de micro algoritmos genéticos en ingeniería de control, de forma específica en la sintonización de controladores PID se documenta en [12, 13, 14] donde se ajustan estructuras de controlador PID, aplicadas observadores de estado, controladores difusos y un PID con estructura variable respectivamente, usando cromosomas con números reales.

De acuerdo a la literatura especializada, los métodos basados en algoritmos evolutivos presentan mejores resultados en relación con los analíticos. Los trabajos consultados no reportan estudios sobre la estabilidad del algoritmo empleado, es decir la repetibilidad de estos en relación con el número de ejecuciones realizadas.

En este trabajo, se propone realizar la sintonización de ganancias k_p, k_i, k_d considerando los siguientes casos:

1. Un algoritmo genético canónico con cruzamiento aritmético no uniforme generacional
2. Un algoritmo genético canónico con cruzamiento aritmético no uniforme durante el cruzamiento
3. Un micro algoritmo genético con cruzamiento aritmético no uniforme generacional
4. Un micro algoritmo genético con cruzamiento aritmético no uniforme durante el cruzamiento

Posteriormente se realiza un estudio de repetibilidad, para determinar la estabilidad de estos con respecto al número de ejecuciones, despues los resultados de estos algoritmos son contrastados con el ajuste de Zigler-Nichols

2. Algoritmos genéticos

En esta sección se describen los algoritmos que se emplean en el desarrollo de la propuesta que se documenta en el presente trabajo.

2.1. Algoritmo genético propuesto

Un algoritmo genético simula algunos aspectos propios de la teoría de la evolución de las especies de Darwin. Los mejores individuos de una determinada población tienen mayores posibilidades de supervivencia y reproducción; las posibilidades disminuyen o son nulas para los débiles. Los pasos que caracterizan a un algoritmo genético simple son los siguientes [15] y [16].

i. definir una función de aptitud o función objetivo
ii. generar una serie de posibles soluciones de manera aleatoria (población)
iii. codificar la población
iv. evaluar con la población, iniciando así la *i*-ésima generación
v. seleccionar soluciones que se reproducirán
vi. aplicar una operación de cruzamiento
vii. mutar algunos elementos resultantes del cruzamiento
viii. reemplazar elementos de la población de la *i* esima generación con los mejores elementos de vi y vii
ix. detener, si se cumple criterio de paro, y en caso contrario ir al paso a iv

2.2. Micro algoritmo genético propuesto

Los micro algoritmos genéticos tienen las siguientes características de diseño:

1. Una población pequeña (3 a 5 cromosomas) generados de forma aleatoria.
2. Debe de encontrarse una convergencia nominal y tener un procedimiento de reinicio.
3. Es necesario preservar al menos al mejor individuo resultante del proceso de convergencia nominal, esto por medio del elitismo.

De manera general el pseudocódigo del algoritmo propuesto es el siguiente [17]:

i. definir una función de aptitud o función objetivo.
ii. generar una población de trabajo con posibles soluciones de manera aleatoria.
iii. codificar las soluciones generadas.
iv. evaluar las soluciones.
v. seleccionar soluciones que se reproducirán.
vi. aplicar operadores de cruzamiento para obtener nuevas soluciones.
vii. reemplazar elementos de la población con los mejores elementos del paso vi.
viii. si se cumple el criterio de convergencia nominal establecido pasar a ix, en caso contrario ir al paso a iv.
ix. conservar a un porcentaje de soluciones obtenidas de vii.
x. generar una nueva población de trabajo a partir de los elementos de ix y complementados con cromosomas generados de forma aleatoria.
xi. si se cumple la convergencia general se detiene el algoritmo, caso contrario saltar a iv.

Los algoritmos evolutivos constituyen técnicas que se engloban bajo el concepto de soft computing, el cual es un enfoque que remarca la habilidad de la mente humana para razonar y aprender en un ambiente de incertidumbre e imprecisión. Este término fue acuñado por Lofti Zadeh en 1992 [18] Cuando existe una sinergia entre las técnicas antes mencionadas se denomina inteligencia computacional [19].

3. Metodología

En esta sección se determinan los parámetros de los operadores de los algoritmos propuestos, lo cual se llevó a cabo para optimizar el controlador PID, aplicado a la celda termoeléctrica. Donde el objetivo es la minimización del valor cuadrático medio (RMS) del error, el cual se explica como la diferencia que existe entre la respuesta del TEC y la referencia deseada. Con esta finalidad se probaron dos técnicas evolutivas, la primera un algoritmo genético canónico y la segunda un micro algoritmo genético.

3.1. Función objetivo

Como se mencionó en la sección 2 de este documento, un algoritmo genético tiene una función objetivo, para este caso la función objetivo se puede construir a partir de la función de transferencia en lazo cerrado del esquema propuesto en la figura 2 y las ecuaciones 1 y 2 obteniéndose la ecuación 3.

$$T(s) = \frac{G_I(s)C(s)}{1 + G_I(s)C(s)}R(s) \tag{3}$$

De las ecuaciones 2 y 3 se puede observar que la salida $T(s)$ depende de los valores k_p, k_i, k_d , así mismo de la figura 2 se puede definir la ecuación 4.

$$E(s) = R(s) - T(s) \tag{4}$$

Una función objetivo se puede definir como se muestra en la ecuación 5 [20], a partir de esta información y del concepto de valor cuadrático medio es posible construir la función objetivo que se muestra en la ecuación 6 donde T es el tiempo de simulación.

$$\min(\max) f(x), x = [x_1, x_2, \dots x_n]^T \in \mathbb{R}^n \tag{5}$$

$$f_{obj}(k_p, k_i, k_d) = \max\left(\frac{1}{1 + \sqrt{\frac{1}{T}\int_0^T E(k_p, k_i, k_d)^2}}\right) \tag{6}$$

3.2. Población y codificación

La colección de sujetos propuestos como posibles soluciones son generados de manera aleatoria (40 en total) con una distribución uniforme, codificando los cromosomas con números reales, por lo tanto es posible describirlos matemáticamente como: cromosoma $= \left[\mathrm{k_p}, \mathrm{k_i}, \mathrm{k_d}\right]^{\mathrm{T}}$.

Para la versión del micro algoritmo genético, de acurdo a la literatura especializada, se plantea una población de trabajo de 5 individuos conservando la forma del cromosoma propuesto.

3.3. Selección

Los cromosomas que se seleccionaron para el cruzamiento en sucesivas generaciones fueron escogidos mediante una ruleta, donde los sujetos con mayor valor de afinidad, (mayor valor de $f_{obj}(k_p, k_i, k_d)$ se privilegian sobre los de menor afinidad. El operador de ruleta es el más estocástico de los métodos de selección, en relación con el torneo y el elitismo, es por eso que fue empleado en este trabajo. Por otra parte el micro algoritmo genético de acuerdo a lo estipulado por emplea operadores de elitismo.

3.4. Cruzamiento

El cruzamiento es determinado por el tipo de codificación. En este caso se emplea el operador de cruce denominado aritmético, cuyo procedimiento se muestra en esta sección. Sean dos cromosomas $\mathrm{C_1} = \left[\mathrm{k_p^1}, \mathrm{k_i^1}, \mathrm{k_d^1}\right]$ y $\mathrm{C_2} = \left[\mathrm{k_p^2}, \mathrm{k_i^2}, \mathrm{k_d^2}\right]$ que fueron seleccionados mediante un procedimiento de ruleta, los descendientes de estos $\mathrm{H_k} = \left[\mathrm{k_p^k}, \mathrm{k_i^k}, \mathrm{k_d^k}\right]$ donde $\mathrm{k} = 1{,}2$ son generados mediante, para $\alpha = [0{,}1]$. Esto se muestra en las ecuaciones 7 y 8

$$\mathrm{H_1} = \alpha(\mathrm{C_1} + ((1-\alpha) * \mathrm{C_2}) \quad (7)$$

$$\mathrm{H_2} = \alpha(\mathrm{C_2} + ((1-\alpha) * \mathrm{C_1}) \quad (8)$$

Cuando el valor α varia en las generaciones o en los cruzamientos se trata de un cruzamiento no uniforme. En este trabajo se realizó el contraste entre las dos posibles situaciones presentes en el valor de α tanto en el algoritmo genético canonico como en el micro algoritmo genético así como el método de ajuste analítico, el criterio de Zigler-Nichols.

3.5. Mutación

Con el operador de mutación que se muestra en la ecuación 9, se alteran dos individuos por cada generación del algoritmo genético canónico, para esto se

utiliza el operador genético de mutación por paso cuyo proceso se muestra a continuación. A partir de un cromosoma C_i' se puede obtener un cromosoma transformado o mutado C_i'' a partir de la siguiente expresión donde el tamaño de paso de la mutación $\beta = [0,1]$ y la dirección de la misma se representa por d. Por otra parte el algoritmo evolutivo de población reducida no utiliza este operador.

$$C_i'' = C_i' + \beta * d \quad (9)$$

3.6. . Criterio de paro

Como se muestra en el pseudocódigo listado en la sección 2.2 el algoritmo se ejecutara hasta que se cumpla un determinado criterio, los cuales en términos generales son dos: un determinado número de ejecuciones (denominadas generaciones) o la convergencia del algoritmo, este último es el empleado en esta propuesta. Por otra parte el micro algoritmo genético tiene dos tipos de convergencia, la nominal y la general, la primera es definida 5 generaciones iteraciones y a segunda se define por un ciclo de 50 iteraciones. Los parámetros del equipo de cómputo empleado fue una computadora Mac Pro de 8GB de memoria RAM, se usó Scilab como software libre.

4. Resultados

Para encontrar cual método de ajuste de parámetros de un PID es más eficiente en la minimización del error cuadrático medio de una celda termoeléctrica se desarrollaron una serie de simulaciones codificadas en Scilab con una temperatura de referencia o de set point de -5 grados centígrados (una temperatura estándar para muchos congeladores de uso industrial), en la primera se sintonizó el controlador por medio del ajuste de Zigler-Nichols, obteniéndose como respuesta la mostrada en la figura 3. Posteriormente se realizó un conjunto de 50 ejecuciones a los algoritmos genético canónico y micro genético con la finalidad de determinar la estabilidad de estos con respecto al número de veces que se ejecuta el algoritmo, esta se calcula a partir de la desviación estándar relativa (desviación estándar sobre media aritmética) los resultados de esta prueba de repetibilidad se observan en la tabla 1.

Tabla 1. Resultados de la estabilidad de los algoritmo propuestos

Técnica	Desviación estándar relativa de la función objetivo (%)	Desviación estándar relativa de la ganancia k_p	Desviación estándar relativa de la ganancia k_i	Desviación estándar relativa de la ganancia k_d
GA no uniforme generacional	2.2139	19.7723	26.7916	26.1840

Técnica	**Desviación estándar relativa de la función objetivo (%)**	**Desviación estándar relativa de la ganancia k_p**	**Desviación estándar relativa de la ganancia k_i**	**Desviación estándar relativa de la ganancia k_d**
GA no uniforme durante la cruza	3.1341	24.6446	29.5704	28.6616
μ-GA no uniforme generacional	0.5212	5.7215	15.4956	7.4130
μ-GA no uniforme durante la cruza	0.6667	8.7023	12.9856	8.1989

Ahora bien los criterios de desempeño del controlador PID que se evaluaron a la par que el error cuadrático medio fueron t_s o tiempo de establecimiento que se define como el tiempo que la celda termoeléctrica llega a su temperatura final y el porcentaje de sobreimpulso M_p (%) representa el valor pico máximo de la curva de respuesta de la celda termoeléctrica. Estos datos, que se muestran en la tabla 2, son obtenidos de los promedios de las ganancias k_p, k_i, k_d de los algoritmos genéticos propuestos y del ajuste de Zigler Nichols. En la figura 4 se grafican las respuestas promedios obtenidas de los algoritmos genéticos canónicos; mientras que en la figura 5 se representan las respuestas de los μ-GA.

Tabla 2. Resultados de las técnicas de ajuste empleados.

Técnica	$\mathbf{k_p}$	$\mathbf{k_i}$	$\mathbf{k_d}$	Error RMS	$\mathbf{t_s}$	$\mathbf{M_p}$ (%)
GA no uniforme generacional	-2.382	-2.062	-2.079	0.2792	25	11.624
GA no uniforme durante la cruza	-2.227	-2.128	-2.104	0.2831	25	12.670
μ-GA no uniforme generacional	-4.531	-3.997	-4.411	0.1725	21	8.020
μ-GA no uniforme durante la cruza	-4.438	-4.128	-4.394	0.1732	25	8.424
Ajuste de Zigler Nichols	-2.4	-0.6	-2.4	0.3379	35	11.624

En la figura 6 se muestra de forma gráfica las convergencias de los algoritmos genéticos canónicos propuestos en este trabajo, mientras que la figura 7 representa la convergencia del μ-GA.

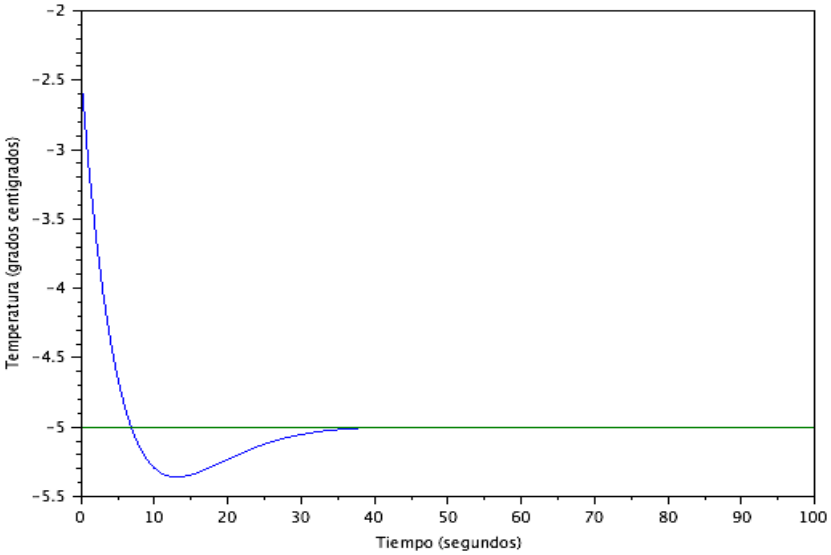


Fig. 3. Respuesta de la TEC con PID ajustado por Zigler-Nichols.

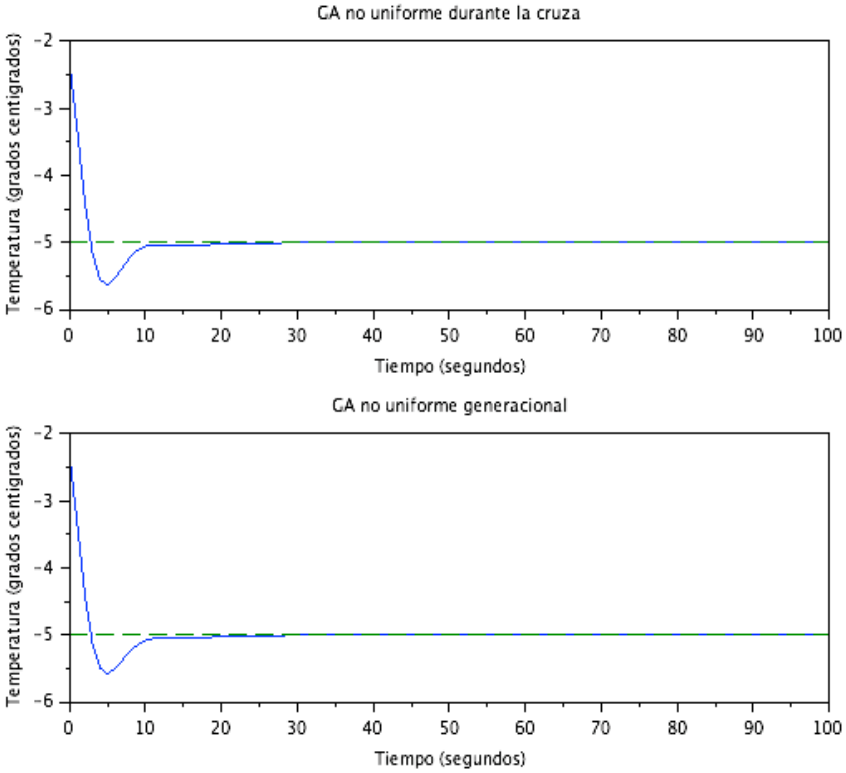


Fig. 4. Respuestas promedio obtenidas de los algoritmos genéticos canónicos.

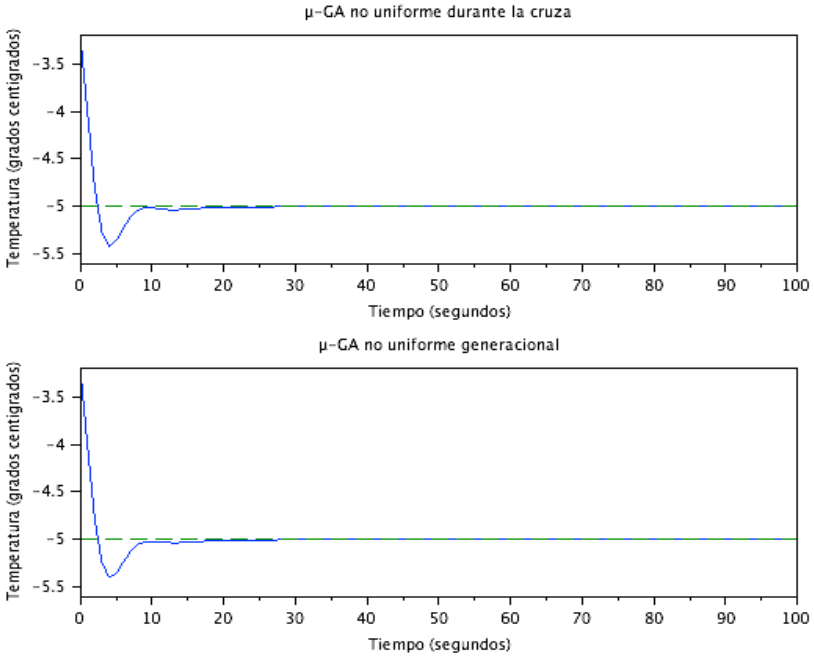


Fig. 5. Respuestas promedio obtenidas de los micro algoritmos genéticos.

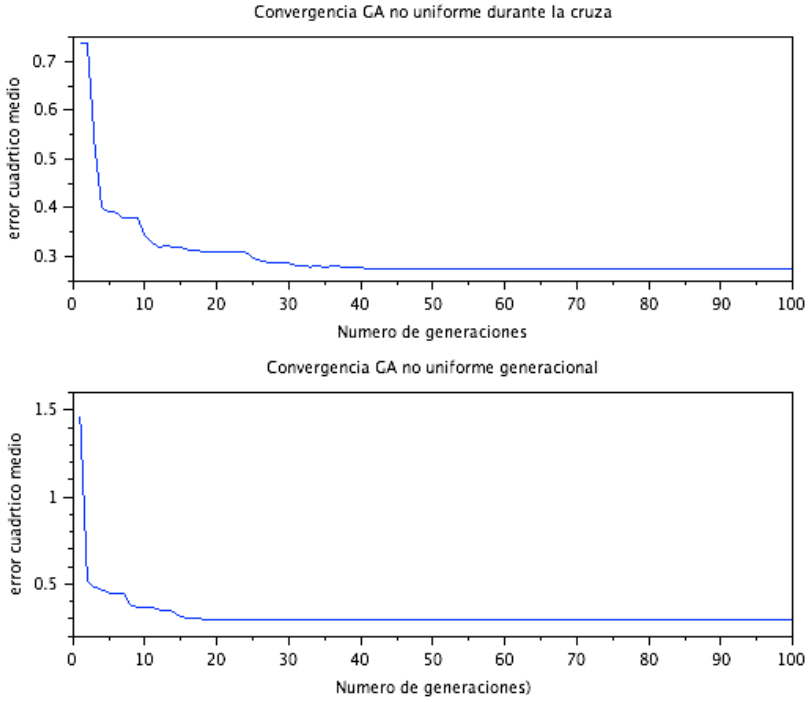


Fig. 6. Convergencia del algoritmo genético con respecto al error cuadrático medio.

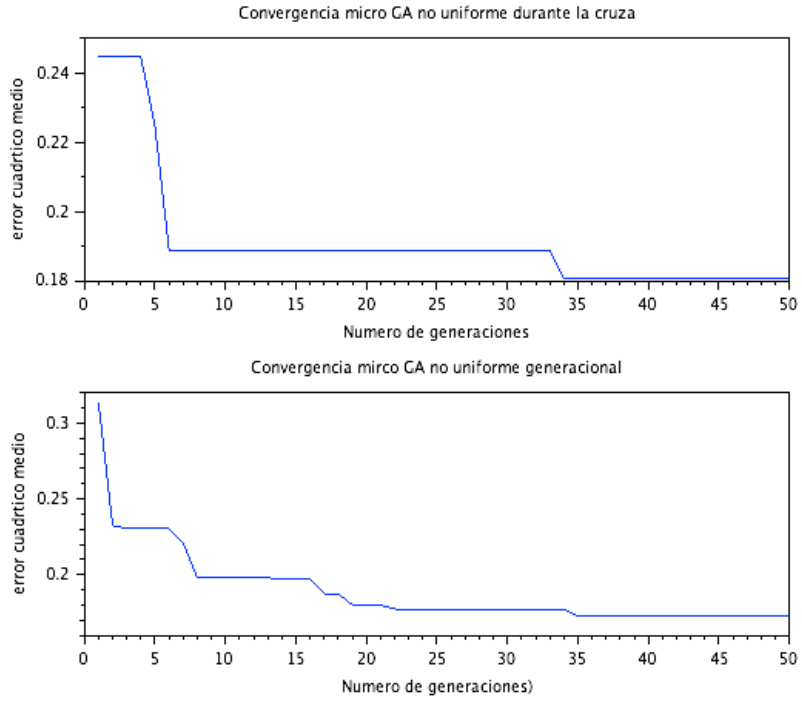


Fig. 7. Convergencia del mirco algoritmo genético con respecto al error cuadratico medio

5. Conclusiones

De acuerdo a las figuras y las tablas producto de las simulaciones realizadas se observa que la técnica evolutiva más adecuada para la sintonización de un PID aplicado a una celda termoeléctrica es el μ-GA con cruzamiento no uniforme generacional. Esta afirmación se soporta en los siguientes hechos:

1. Presenta mayor repetibilidad con respecto al resto de los algoritmos propuestos.
2. Los tiempos de establecimiento es decir el tiempo en que la celda se establece a la referencia de -5 grados centígrados, el valor cuadrático medio del error, así como el valor del sobre impulso son menores al resto de los algoritmo propuestos y al criterio de Nichols, esto se refleja en un mejor desempeño, de acuerdo a lo estipulado por ingeniería de control.

Además de los hechos descritos es posible afirmar que el μ-GA no uniforme generacional al igual que el μ-GA no uniforme durante la cruza tiene un costo computacional menor, a pesar de tener una convergencia general más grande con relación a la versión canónica, esto se verifica en las figuras 6 y 7. Es destacable que no existen diferencias estadísticas significativas entre las variantes de los micro

algoritmos, mismo hecho que se repite en los algoritmos canónicos, esto se refleja en las tablas 1 y 2.

En cuanto a la convergencia de los algoritmos esta refleja que ambos micro algoritmos necesitan mayor número de iteraciones, esto debido al número de cromosomas generados, pero en relación con los tiempos de simulación obtenidos estos son menores con respecto a la versión canónica.

A partir de esto, como trabajos a futuro se sugiere el estudio de otros operadores de cruza [21] aplicados a micro algoritmos genéticos, ademas, de acuerdo a los tiempos de ejecución de estos es posible teorizar sobre implementaciones de este en la aplicación práctica del control de temperatura de una celda termoeléctrica a partir de elementos de alta escala de integración como los procesadores digitales de señales o tarjetas SBC (single board computer).

Agradecimientos. Los autores agradecen a la Secretaria de Investigación y Estudios Avanzados de la UAEMex por el apoyo brindado en la realización de este proyecto.

Referencias

1. Tarter R.: Solid-state power conversion handbook. United State of America: John Wiley and Sons (1993)
2. Shaojing, S., Qin, Q.: Temperature Control of Thermoelectric Cooler Based on Adaptive NN-PID. In: Electrical and Control Engineering (ICECE), International Conference on, pp. 2245–2248 (2010)
3. Burger, C.: Propeller performance analysis and multidisciplinary optimization using a genetic algorithm. Auburn University, ProQuest Dissertations and Theses (2007)
4. Yang X. S.: Nature-Inspired Metaheuristic Algorithms United Kingdom, Luniver Press, p. 148 (2011)
5. Krishnakumar, K.: Micro-genetic algorithms for stationary and non-stationary function optimization. In: SPIE Proceedings: Intelligent Control and Adaptive systems, pp. 289–296 (1989)
6. Saad, M.S., Jamaluddin, H., Darus, I.Z.M.: Implementation of PID Controller tuning using differential evolution and Genetic Algorithms. Int. J. of Innovative Computing, Information and Control (ICIC), vol. 8, no. 11, pp. 7761–7779 (2012)
7. Junli, L., Jianlin, M., Guanghui, Z.: Evolutionary algorithms based parameters tuning of PID controller. In: Control and Decision Conference, Mianyang, China, pp. 416–420 (2011)
8. Reynoso-Meza, G., Sanchis, J., Herrero, J.M., Ramos, C.: Evolutionary auto-tuning algorithm for PID controllers, IFAC Conf. on Advances in PID control PID'12, Brescia(Italy), March 28-30,.FrB1.6. (2012)
9. Krohling R.A., Rey, J. P.: Design of Optimal Disturbance Rejection PID Controllers Using Genetic Algorithms. IEEE Transactions on evolutionary computation, vol. 5, no. 1 (2001)
10. Valarmathi, R., Theerthagiri, P.R., Rakeshkumar, S.: Design and Analysis of Genetic Algorithm Based Controllers for Non Linear Liquid Tank System. Advances in Engineering, Science and Management (ICAESM), pp. 616–620 (2012)
11. Yang, M. ; Zhao, W., Hu, Z.: Optimization of dc motor rotary speed controller based on the genetic algorithm. In: World Automation Congress (WAC), pp. 1–4 (2012)
12. Yao, L., Wen, Hong-Kang: Design of Observer Based Adaptive. International Journal of

Innovative Computing, Information and Control, vol. 9, no. 2, pp. 667–677 (2013)
13. Bedwani, W.A.; Ismail, O.M.: Genetic optimization of variable structure PID control systems. In: Computer Systems and Applications, ACS/IEEE International Conference on, pp. 27–30 (2001)
14. Mendoza, E., Morales, D. A., López, R. A., López, E. A., Vannier, J. C., Coello, C,: Multiobjective location of automatic voltage regulators in a radial distribution network using a micro genetic algorithm. Power Systems, IEEE Transactions on, vol. 22, no. 1, pp. 404–412 (2007)
15. Mitchell, M.: An introduction to genetic algorithms. MIT Press, USA (1998)
16. Gen, M., Cheng, R.: Genetic algorithms and engineering optimization. John Wiley and Sons (2000)
17. Herrera-Lozada, J. C., Calvo, H., Taud, H., Portilla-Flores, E.A.: Propuesta de una Metodología Generalizada para Diseñar Micro Algoritmos Bioinspirados. In: Congreso internacional de computo en optimización y software, Cuernavaca, pp. 189–199 (2009)
18. Konar, A.: Computational intelligence principles, techniques, and applications. Springer-Verlag (2005)
19. Yu, X. Introduction to Evolutionary Algorithms. Springer (2010)
20. Ponstein, J.P.: Approaches to the Theory of Optimization. Cambridge University Press (2004)
21. Sánchez, A.M., Lozano, M., Herrera, F.: Algoritmos Genéticos para Codificación Real con Operador de Cruce Híbrido con Múltiples Descendientes. In: VI Congreso Español sobre Metaheurísticas, Algoritmos Evolutivos y Bioinspirados, Málaga, pp. 411–418 (2009)

Combinación de clasificadores para el análisis de sentimientos

Monserrat Ramirez García[1], Maya Carrillo Ruiz[1] y Abraham Sánchez López[1]

Benemérita Universidad Autónoma de Puebla,
Facultad de Ciencias de la Computación, Puebla, México

{mramirez88, cmaya, asanchez}@cs.buap.mx

Resumen. El presente trabajo, propone una arquitectura fusionando los clasificadores: SVM, árboles de decisión y Naive Bayes, mediante mayoría de votos, ventanas y cascada, para explorar el desempeño de la tarea de análisis de sentimientos. Se utilizó un corpus en español de 2625 opiniones, previamente preprocesado. Para representarlo, se emplearon bigramas, bolsa de palabras con pesado tf-idf, etiquetado POS y una representación basada en la teoría de la valoración. Los resultados obtenidos muestran una mejora en medida F del 18.13 %, con respecto a los resultados de los clasicadores base.

Palabras clave: combinación de clasificadores, análisis de opinión, ensamble de clasificadores.

1. Introducción

La combinación de múltiples clasificadores es considerado un reto importante para lograr la robustez y precisión en diversas tareas, como el análisis de opinión o sentimientos. A pesar de los avances tecnológicos, el análisis de sentimientos sigue siendo un tema de investigación abierto, especialmente en el idioma español. En este trabajo se exploran diversas formas de combinar las decisiones de múltiples clasificadores como una forma viable de mejorar el rendimiento de la tarea de análisis de sentimientos. Al respecto cabe mencionar el trabajo desarrollado por Shoushan Li [1] et al. quienes utilizan máquinas de soporte vectorial y atributos como unigramas, adjetivos, adjetivos+advervios, nombres, entre otros y mejorar la precisión obtenida por los clasificadores individuales al combinar los tres mejores de los seis utilizados. El presente artículo está organizado de la siguiente manera: en la sección 2 se describe la tarea de análisis de sentimientos, en la sección 3 se describe la tarea de clasificación, así como cada uno de los clasificadores utilizados, posteriormente en la sección 4 se define el concepto de ensamble de clasificadores, y los métodos de combinación empleados. En la sección 5 se describe la arquitectura propuesta, en la sección 6 se muestran los experimentos y resultados obtenidos, en la sección 7 se presentan las conclusiones y finalmente en la sección 8 se presenta el trabajo futuro.

2. Análisis de sentimientos

Las opiniones son fundamentales para casi todas las actividades humanas, porque son importantes factores de influencia en nuestros comportamientos. El análisis de sentimientos (AS), también llamado minera de opinión es un campo de estudio que analiza las opiniones, sentimientos, evaluaciones, actitudes de las personas hacia entidades como productos, servicios, organizaciones, individuos, cuestiones, eventos, tópicos y sus atributos.

El problema se define de la siguiente manera:

Dado un conjunto de documentos de texto de evaluación D que contienen opiniones (o sentimientos) acerca de objetos, se pretende extraer atributos y componentes de objetos que han sido comentados en cada documento d en D y determinar si los comentarios son positivos, negativos o neutros [2].

Aquí, opinión es una cuádrupla:

$$(g, s, h, t) \tag{1}$$

Donde g es la opinión, s es el sentimiento de la opinión, h es la persona que expresa la opinión, y t es el tiempo o la fecha en que se expresa la opinión.

Los indicadores mas importantes de sentimientos, son las palabras que expresan sentimiento, llamadas palabras de opinión (opinion words). Estas son palabras que comúnmente son usadas para expresar sentimientos positivos o negativos. Por ejemplo, bueno, maravilloso y estupendo son palabras que expresan sentimiento positivo, en cambio malo, peor y terrible son ejemplos de palabras que expresan sentimiento negativo, a dichas palabras se les conoce comúnmente como lexicón de opiniones (sentiment lexicon o opinion lexicon). A pesar de que las palabras y frases con sentimiento son muy importantes, para el análisis de sentimientos no son suficientes para obtener éxito, la tarea es mucho mas compleja, es decir que el lexicón de opiniones es necesario pero no suficiente para el AS. A continuación se describen algunas situaciones que hacen de AS un problema complejo [7].

- Una palabra que expresa un sentimiento negativo o positivo puede tener orientaciones opuestas, según el contexto de la oración.
- Una oración que contiene una palabra considerada como expresion de sentimiento, puede no expresar un sentimiento.
- Oraciones Sarcásticas.
- Opiniones spam

2.1. Representaciones textuales

Para analizar las opiniones es necesario representarlas de manera que puedan ser procesadas en una computadora, en este trabajo se utilizaron cuatro representaciones textuales, tres de ellas comúnmente empleadas: bolsa de palabras con pesado tf-idf, n-gramas, etiquetas POS y una representación basada en la teoría de la valoración.

N-gramas Es una representación tradicional en recuperación de la información, que consiste de palabras individuales (unigramas), o conjuntos de palabras (n-gramas), con sus frecuencias asociadas. En algunos casos podemos representar mejor un concepto mediante la unión de n palabras que se encuentran adyacentes al término principal, lo que se le conoce como n-gramas. La importancia de esta representación radica en que la posición de las palabras es considerada, puesto que el significado de una palabra, no tiene sentido sin las palabras adyacentes que le acompañan en cualquier texto, por lo que la posición de una palabra afecta potencialmente en el sentido del significado de la oración, es decir el sentimiento o la subjetividad dentro de una unidad textual. Para el trabajo realizado se utiliza n-gramas de tamaño n = 2, es decir, bigramas.

Partes de la oración (POS) Una técnica de representación muy utilizada se basa en las reglas linguísticas, donde las palabras y frases son categorizadas como sustantivos, verbos, adjetivos y adverbios. De acuerdo con Turney, son características gramaticales que tienen la capacidad de expresar subjetividad [3]. Existen investigaciones enfocadas principalmente en adjetivos y adverbios, como en el trabajo reportado por Farah Benamara et al [4], en donde expone que las expresiones de una opinión dependen principalmente de algunas palabras, por ejemplo, la palabra *bueno* es utilizada comúnmente para una opinión positiva, y la palabra *malo*, para algo negativo, dichas palabras son identificadas como adjetivos en términos linguísticos.

En general los adjetivos son importantes indicadores en una opinión, son considerados características especiales, sin embargo no significa que otras partes de la oración no contribuyan a la expresión de sentimientos. Existen trabajos en donde los sustantivos, verbos, adverbios y sustantivos subjetivos también han tenido buenos resultados [14].

TF-IDF (term frequency-inverse document frequency) Es un esquema de ponderación de términos comúnmente utilizado para representar documentos de texto como vectores, que se ajusta al modelo denominado bolsa de palabras, donde cada documento es representado como serie de palabras sin orden. Se trata de una medida estadística de cuan importante es una palabra para un documento y para un corpus. Dicho esquema, se utiliza frecuentemente en tareas de ordenamiento o reordenamiento de los resultados de búsqueda, generación de resúmenes de texto, agrupación y clasificación de documentos, identificación del autor de algún texto, recomendación de documentos, etc.

Cálculo del TF Un término t_i que aparece muchas veces en un documento d_j es mas importante que otro que aparece pocas.

$$tf_{ij} = \frac{(n_{ij})}{\sum_{k=1}^{N} n_{kj}} = \frac{(n_{ij})}{|d_i|} \tag{2}$$

Donde n_{ij} es el número de ocurrencias del término t_i en el documento d_j y $\sum_{i=1}^{N}$ es la suma del número de ocurrencias de todos los términos en el documento *dj*, es decir el tamaño del documento *dj*.

Cálculo del IDF

Un término t_j que aparece en pocos documentos, discrimina mejor que uno que aparece en muchos.

$$idf_j = log\left(\frac{N}{n_j}\right) \tag{3}$$

Donde N es el número total de documentos, y n_j es el número de documentos que contiene el término t_j.

Representación final del documento

Cada elemento queda representado como un vector de características d_j:

$$d_j = (d_{j1}, ..., d_{jn}) \tag{4}$$

$$\text{donde, } d_{ij}{=}tf_{ij}{*}idf_{ij}$$

Es decir finalmente se seleccionan n términos con los valores más altos en todos los documentos.

Teoría de la valoración utilizando reglas sintácticas La teoría de la valoración propuesta por Peter R.R White [5], se ocupa de los recursos lingüísticos por medio de los cuales las personas expresan alguna opinión. Particularmente del lenguaje (expresiones lingüísticas), la valoración, la actitud y la emoción del conjunto de recursos que explícitamente posicionan de manera interpersonal las propuestas y proposiciones textuales. Es decir, trabaja con los significados de las palabras que hacen variar o modificar los términos del compromiso del hablante en sus emisiones, por lo que modifican lo que está en juego en la relación interpersonal.

Dicha representación fue implementada por Morales de Jesús V. M. en su trabajo de tesis [6], que haciendo uso de un diccionario de actitud, el cual contiene elementos taxonómicos de la teoría de la valoración *(juicio, apreciación y afecto*), y utilizando sintagmas adverbiales, busca obtener un significado más preciso de las expresiones de valoración presentes en los textos. El objetivo es contabilizar los valores de positivo, negativo, juicio, apreciación y afecto, que están presentes en una opinión cualquiera, como si se tratase de una bolsa de palabras ponderada, sin embargo las reglas sintácticas, empleadas para identificar los sintagmas adverbiales, juegan un papel primordial en este proceso, ya que dependiendo del tipo de regla, los valores asignados a los elementos de actitud pueden aumentar, disminuir, o intercambiarse, afectando de esa manera los valores finales asignados al sentimiento de cada opinión.

3. Clasificación

Los algoritmos de clasificación son métodos que dado un conjunto de ejemplos de entrenamiento infieren un modelo de las categorías en las que se agrupan los

datos, de tal forma que se pueda asignar a nuevos ejemplos una o más categorías de manera automática mediante analogías a los patrones de dicho modelo.

En todo proceso de clasificación supervisada, se cuenta con dos conjuntos de ejemplos etiquetados, uno de entrenamiento y otro de pruebas. Primeramente se utiliza el conjunto de entrenamiento a fin de construir el modelo de clasificación y se verifica haber alcanzado el resultado adecuado según la métrica definida, entonces el proceso termina, sino el proceso de entrenamiento se repite, hasta obtener el resultado deseado.

Para clasificar documentos, primero se representan, empleando representaciones, como las descritas en la sección 2.1. Una vez representados, podrán introducirse al clasificador seleccionado. Recientemente se han desarrollado técnicas para construir conjuntos de clasificadores cuyas decisiones son combinadas (de forma pesada o no) para clasificar nuevos ejemplos. Lo que se ha encontrado es que en general son mejores clasificadores que los clasificadores individuales (o base) que se usaron en su construcción.

La clasificación, puede definirse como la tarea de predecir una variable discreta *"y"* usando un conjunto de características x_1, x_2, ..., x_n como variables independientes. Para realizar el entrenamiento del clasificador se necesita una función hipótesis h de una colección de ejemplos de entrenamiento. Dicha colección tiene la forma *(X,Y)* y usualmente se refiere a un conjunto de datos. Cada entrada del conjunto de datos es una tupla *(x,y)*, donde x es el conjunto de características y y es la clase o etiqueta la cual es una variable discreta con c posibles categorías. Cuando los resultados posibles son restringidos a valores binarios, $y_i \epsilon +1, -1, \forall$ i ϵ $1, ..., N$ [10].

En este trabajo se utilizan tres clasificadores base: Máquina de Soporte Vectorial, Naive Bayes y Arboles de Decisión. Cada uno de estos algoritmos se describe brevemente a continuación:

3.1. Naive bayes

Es un clasificador probabilístico que aplica el Teorema de Bayes para estimar la probabilidad posterior *P(y | x)* de la clase y dada la variable x

$$P(y|x) = \frac{P(y|x)P(y)}{P(x)} \tag{5}$$

Naive Bayes se centra en las probabilidades *P(x|y)* que se refieren a la verosimilitud y representan la probabilidad de observar el valor x, dado el valor de clase y. Debido a esto Naive Nayes es considerado un *clasificador generativo*.

3.2. Máquina de soporte vectorial

La máquina de Soporte Vectorial SVM es un clasificador binario discriminante, dirigido a encontrar el hiperplano óptimo $(w^T * x + b)$ que separa los dos posibles valores de la variable etiquetada y ε {*+1,-1*}, de acuerdo al espacio de características representado por x. El hiperplano óptimo es aquel que maximiza

el margen entre las instancias positivas y negativas en el conjunto de datos de entrenamiento formado por N observaciones. La tarea de aprendizaje de una SVM se formaliza con el siguiente problema de optimización:

$$\min_{w,b} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}\xi_i \tag{6}$$

$$\text{sujeto a } \; y_i(w^T x_i + b) \geq 1 - \xi_i \; \forall i\varepsilon\{1, ...N\}$$
$$\xi_i \geq 0, \; \forall i\varepsilon\{1, .., N\}$$

El objetivo del problema se enfoca en dos aspectos, el primero, obtener el máximo margen en el hiperplano y minimizar el error $\sum_i^N xi_i$. El parámetro C se refiere al parámetro suave de regularización de margen y controla la sensibilidad de la SVM para los posibles valores atípicos.

3.3. Arboles de decisión

Un árbol de decisión describe un conjunto de reglas organizadas de forma jerárquica, que implementan una estructura de decisión. Se compone de hojas y nodos. Una hoja registra una respuesta *(clase)* y un nodo especifica algunas condiciones de las pruebas que se llevarán a cabo en un valor único, rasgo de una instancia, con una rama y sub-árbol para cada posible resultado de la prueba. Para un determinado vector, se toma una decisión partiendo de la raíz de un árbol, y se recorre el árbol en función del resultado de una prueba de estado en cada nodo, hasta que se encuentra una hoja [11]. El proceso de construcción de un árbol de decisión es una partición recursiva de un conjunto de entrenamiento.

4. Ensamble de clasificadores

La idea de un ensamble de clasificadores, es combinar un conjunto de clasificadores para resolver una tarea en conjunto, en donde el objetivo principal es combinar las salidas de los clasificadores base, para generar una salida en donde sean considerados todos los clasificadores y dicha salida sea mejor que la obtenida por cualquier clasificador base, bajo una cierta evaluación [9].

Un ensamble de clasificadores es un grupo de clasificadores quienes individualmente toman decisiones que son fusionadas de alguna manera, para finalmente obtener una decisión por consenso. Los métodos de ensamble son muy efectivos, debido principalmente a que varios tipos de clasificadores tienen sesgos inductivos, y provocan que la diversidad de los clasificadores utilizados reduzca el error de la varianza, sin incrementar el error bias [8].

La combinación de clasificadores y por lo tanto la creación de ensamble de clasificadores fue propuesto para mejorar los resultados obtenidos por los clasificadores base. La llave para producir un ensamble exitoso, es elegir los métodos de clasificación apropiados y seleccionar los clasificadores base indicados para el problema planteado.

Dado el potencial uso del ensamble de clasificadores, existen algunos factores que deben ser diferenciados entre los métodos de ensamble. Los principales factores se listan a continuación:

1. Relación inter-clasificadores. ¿Cómo cada clasificador afecta a otros clasificadores?
2. Método de combinación. La estrategia de combinar los clasificadores generados por un algoritmo de inducción. El combinador simple determina la salida exclusivamente a partir de las salidas de los inductores individuales.
3. Generador de diversidad. Con el objetivo de realizar un ensamble eficiente, debe existir diversidad entre los clasificadores involucrados. La diversidad puede ser obtenida a través de diferentes presentaciones de entrada de datos, como en bagging, variaciones en el diseño de aprendizaje, aplicando una sanción a las salidas para fomentar la diversidad.

Por otra parte, existen diferentes estructuras para combinar los clasificadores, a continuación se describen las utilizadas en el presente trabajo.

4.1. Cascada

Es una arquitectura para combinar clasificadores, que puede presentar n niveles, sin embargo normalmente presenta dos niveles, en donde el nivel 1 es entrenado con el conjunto de datos original, el nivel 2 con un conjunto de datos aumentado, el cual contiene las características del conjunto de datos original junto con la salida del clasificador del nivel 1. La salida del clasificador del nivel 1 es un vector que contiene la distribución de probabilidad condicional $(p_1, ..., p_c)$, donde c es el número de clases del conjunto de datos original, y p_i es la estimación de probabilidad calculada por el clasificador del nivel 1, de que la instancia pertenezca a la clase i.

El entrenamiento del clasificador del nivel 2 es influenciado por el clasificador del nivel anterior, debido a que considera su salida obtenida, derivando un esquema global sobreentrenado. Sin embargo, en cascada se reduce este problema debido a dos razones: en cada nivel se utiliza un clasificador de diferente naturaleza al otro y además el clasificador del nivel 2 no se entrena únicamente con la salida del clasificador de nivel 1, sino que además tiene en cuenta las características originales.

4.2. Mayoría de votos

Es un método simple de combinación de clasificadores base, en el cual todos los clasificadores incluidos proveen un voto a alguna de las clases, el método realiza la sumatoria de dichos votos y la clase que recibe más votos es seleccionada como la decisión final. Dicho método es representado por la siguiente ecuación:

$$x \rightarrow w \; if \; w = \arg\max_{w \epsilon \theta} \sum_{i=1}^{T} 1(C_i(x) = w) \tag{7}$$

x es una instancia, θ es el conjunto de etiquetas de clase, w es la clase asignada para la instancia x y C_1,...,C_T son los clasificadores base.

4.3. Ventanas

El método de *Ventanas* es una técnica general, que tiene por objetivo mejorar la eficiencia de los métodos de aprendizaje o clasificadores utilizados, mediante la identificación de un subconjunto adecuado de instancias de entrenamiento. Dicho método se lleva a cabo mediante el uso de un procedimiento de submuestreo.

El método funciona de la siguiente manera: Se selecciona un subconjunto aleatorio de instancias para el entrenamiento de un clasificador *(una ventana)*, el resto de instancias son utilizadas para los datos de prueba, si la precisión obtenida del clasificador es insuficiente, las instancias de prueba clasificadas erróneamente se eliminan de las instancias de prueba y se añaden al conjunto de instancias para el entrenamiento en la siguiente iteración. El proceso continúa hasta que se obtiene una precisión suficiente.

Es importante mencionar que ventanas no combina clasificadores, su tarea radica en mejorar el resultado de un clasificador.

Una vez que se han explicado los métodos de clasificación y arquitecturas utilizadas, es importante conocer las métricas que permitirán evaluar el resultado obtenido por los mismos.

4.4. Métricas de evaluación

Para realizar la evaluación de los métodos de clasificación aplicados sobre un conjunto de datos, existen métricas, a continuación se describen la utilizadas, donde TP son las instancias clasificadas correctamente como positivas, *FP*, son las instancias clasificadas erróneamente como positivas y de la misma manera para las instancias negativas, *FN*, son las instancias clasificadas erróneamente como negativas y *TN* son las clasificadas correctamente como negativas. Ahora teniendo las salidas antes descritas los siguientes criterios de evaluación pueden ser utilizados.

Precisión. Es la fracción de observaciones clasificadas correctamente como positivas, sobre todas las predicciones clasificadas como positivas.

$$Precision = \frac{TP}{TP + FP} \tag{8}$$

Recuerdo. Es la fracción de observaciones clasificadas correctamente como positivas, sobre todas las observaciones positivas.

$$Recuerdo = \frac{TP}{TP + FN} \tag{9}$$

Medida F. Es el significado armónico entre precisión y recuerdo

$$MedidaF = \frac{(1+\beta^2)(2 * Presicion * Recuerdo)}{(\beta^2 * Precision) + Recuerdo} \tag{10}$$

Las medidas de evaluación son promediadas por todas las submuestras, asegurando que todas las observaciones fueron usadas para entrenamiento y prueba.

5. Arquitectura propuesta

La arquitectura propuesta para el corpus en español, consiste de tres niveles los cuales se describen a continuación.

1. En el primer nivel, se aplica el método de mayoría de votos en dos fases.
   a) En la primera fase, se realiza mayoría de votos con los clasificadores base: SVM, Naive Bayes y arboles de decisión, para cada representación del corpus.
   b) En la segunda fase, se aplica mayoría de votos a las mejores salidas de cada representación.
2. El segundo nivel se incluye el método de Cascada. El cual recibe como entrada la salida obtenida por mayoría de votos mejores, obtenido en 1 b). Posteriormente incorpora la entrada al conjunto de datos originales del problema, y realiza la clasificación empleando el clasificador que produzca los mejores resultados en cuanto a medida F de los considerados en el paso 1 a) y 1 b)
3. En el tercer y último nivel, se utiliza el método de ventanas.
   Dado que los resultados obtenidos por cascada tienen una mejora, el método de ventanas toma como entrada la salida proporcionada por Cascada, y selecciona automáticamente el método de clasificación base mejor en medida F, respecto a representaciones y clasificadores, y define un valor N para el número máximo de iteraciones. Teniendo estos parámetros, realiza la clasificación N veces, en cada iteración el algoritmo selecciona las instancias clasificadas de manera errónea y las agrega al conjunto de datos de entrenamiento, intercambiando instancias, hasta que el valor de N se cumpla.
   Los resultados obtenidos son los resultados finales de la arquitectura.

6. Experimentos y resultados

En esta sección se describen las condiciones de los experimentos realizados y los resultados obtenidos, no sin antes describir el corpus utilizado.

6.1. Corpus utilizado

Corpus en español de películas de cine, creado por Fermín L. Cruz, et al, como se describe en [12], con 3878 críticas que contiene una puntuación asignada del 1 al 5 donde 1 es la más negativa y 5 es la más positiva, del cual se tomaron 2625 críticas (1351 positivas, 1274 negativas) no incluyendo las criticas neutras, es decir con puntuación 3.

6.2. Aplicaciones desarrolladas

1. Se realizó una aplicación en Microsoft Visual Studio 2012, para implementar el módulo de representación de textos.
2. Se construyó también una aplicación en Matlab 2014b, para la implementación del sistema de clasificación.

En la tabla 1, se muestra la longitud del vocabulario del corpus empleado, después de realizar el preprocesamiento de los textos.

Tabla 1. Cardinalidad del vocabulario.

Corpus	Vocabulario completo	Vocabulario Truncado
Español	57713	19876 (34.43 %)

6.3. Preprocesamiento de los datos

Una vez elegido el corpus y antes de realizar el Análisis de Sentimientos, primero se realizó un preprocesamiento de los datos, ya que el corpus fue construido a partir de opiniones introducidas por usuarios comunes de la web y no por críticos especializados. Se eliminaron: palabras vacías, símbolos no alfanuméricos, números y signos de puntuación.

6.4. Condiciones de ejecución

Las representaciones utilizadas, fueron bolsa de palabras, bigramas, etiquetas POS considerando adjetivos y adverbios, y la representación de la teoría de la valoración.

Los resultados reportados son el promedio de 10 ejecuciones, aplicando validación cruzada a 10 pliegues, el mecanismo de selección para los diferentes ejemplos, fue aleatorio y sin remplazo, considerando el 50 % de instancias con clase positiva y el otro 50 % de clase negativa, tanto para el entrenamiento como para la prueba, es decir con corpus balanceados.

Los experimentos realizados fueron variando el porcentaje de datos de entrenamiento y de prueba, con 80 % - 20 % y 60 %- 40 % respectivamente. Empleando los clasificadores mencionados previamente: SVM, arboles de decisión y Naive Bayes.

Para el clasificador SVM, se utilizó un kernel lineal.

Tabla 2. Resultados de la arquitectura propuesta

Corpus	Representación	Clasificador	Precisión	Recuerdo	Medida F
60 %-40 %		Mayoría de votos Mejores	0.7323	0.7028	0.7172
	Bigramas	SVM(Nivel 1)	0.7157	0.7761	0.7447
		Cascada(Nivel 2)	**0.8346**	0.8360	**0.8353**
		Ventanas(Nivel 3)	0.8091	**0.8385**	0.8235
	Bolsa de palabras	Naive Bayes(Nivel 1)	0.6057	0.5468	0.5748
		Cascada(Nivel 2)	0.7647	**0.7860**	**0.7752**
		Ventanas(Nivel 3)	**0.9685**	0.6178	0.7544
	POS	Árboles(Nivel 1)	0.5758	0.5431	0.5590
		Cascada(Nivel 2)	0.6352	**0.6581**	**0.6464**
		Ventanas(Nivel 3)	**0.8946**	0.4773	0.6225
	Valoración	SVM(Nivel 1)	0.6932	0.5596	0.6193
		Cascada(Nivel 2)	0.7575	**0.7506**	0.7541
		Ventanas(Nivel 3)	**0.8632**	0.7047	**0.7760**
80 %-20 %		Mayoría de votos Mejores	0.9159	0.8920	0.9038
	Bigramas	SVM(Nivel 1)	0.8241	0.8213	0.8227
		Cascada(Nivel 2)	0.9252	0.9340	0.9296
		Ventanas(Nivel 3)	**0.9788**	**0.9652**	**0.9719**
	Bolsa de palabras	Naive Bayes(Nivel 1)	0.6555	0.5361	0.5898
		Cascada(Nivel 2)	0.9240	**1**	**0.9592**
		Ventanas(Nivel 3)	**0.9526**	0.9198	0.9359
	POS	Naive Bayes(Nivel 1)	0.5799	0.6735	0.6232
		Cascada(Nivel 2)	0.9204	0.6198	0.5750
		Ventanas(Nivel 3)	**0.9221**	**0.9457**	**0.9338**
	Valoración	SVM(Nivel 1)	0.7500	0.5979	0.6654
		Cascada(Nivel 2)	**0.9828**	**0.9385**	**0.9602**
		Ventanas(Nivel 3)	**0.9828**	**0.9385**	**0.9602**

6.5. Experimentos

En la tabla 2 se muestran los resultados obtenidos, para cada nivel de la arquitectura propuesta. La salida del primer nivel, son los resultados obtenidos por el clasificador con mejor medida F, mismos que entran al segundo nivel. Por tal motivo el clasificador base de entrada al segundo nivel varía.

Mayoría de votos mejores, se refiere a la aplicación del método de mayoría de votos, a los mejores resultados obtenidos por SVM, Naive Bayes y arboles de decisión empleando las representaciones: tf-idf, bigramas, POS y teoría de la valoración.

Como puede observarse en la tabla 2 en la partición 60-40 en tres casos se alcanza la mejor medida F en el segundo nivel de la arquitectura, siendo 0.8353 la más alta obtenida con bigramas, seguida por teoría de la valoración con 0.7760, digno de recalcarse pues es una representación que emplea vectores de dimensión 8.

En la partición 80-20, la mejor medida F igual a 0.9719, se obtiene también con bigramas en el tercer nivel de la arquitectura, lo que representa una mejora de 18.13 % con respecto a los resultados del mejor clasificador base SVM con medida F de 0.8227. La clasificación empleando bolsa de palabras y teoría de la valoración también superan al mejor clasificador base desde el segundo nivel de la arquitectura y POS en el tercer nivel.

7. Conclusiones

Existen trabajos limitados de análisis de sentimientos para textos en español, en los que es notorio que la medida F alcanzada es mucho más baja que los métodos reportados para el idioma inglés.

Se han realizado varios experimentos con distintos métodos de clasificación y distintas formas de representación de los datos, los resultados obtenidos han sido muy diversos. Se ha podido distinguir que las características utilizadas han sido un factor determinante para obtener resultados satisfactorios, la selección de características es un aspecto muy importante, puesto que el éxito de la clasificación depende de tomar las características o los atributos que mejor representen a los documentos, ya que de esto dependerá expresar la polaridad correcta de los documentos y obtener resultados satisfactorios.

Los resultados obtenidos con la arquitectura propuesta, son superiores en precisión, recuerdo y medida F, con respecto a los obtenidos por los clasificadores base.

Bigramas resultó ser la representación de textos que permitió obtener mejores resultados, en la arquitectura.

SVM es el clasificador base que mostró mejor desempeño en la arquitectura propuesta.

La arquitectura propuesta obtuvo una mejora del 12.52 %, con respecto a un trabajo publicado en 2011, que utiliza el mismo corpus [13].

8. Trabajo futuro

Se tiene considerado para trabajo futuro, realizar un análisis cualitativo de las instancias clasificadas erróneamente, con el objetivo de identificar similitudes y características de las instancias, para estudiarlas y posteriormente, proponer algún método que contribuya a clasificar correctamente las instancias mal clasificadas.

También se tiene planeado aplicar la arquitectura propuesta, en otras áreas de conocimiento, con el fin de observar el comportamiento de la misma.

Referencias

1. Shoushan L., Chengqing Z., Xia W.: Sentiment classification through combining classifiers with multiple feature sets. In: Natural Language Processing and Knowledge Engineering, NLP-KE 2007. International Conference IEEE, pp. 135–140 (2007)
2. Bo Pang, Lillian Lee: Opinion Mining and Sentiment Analysis. Found. Trends Inf. Retr., Vol 2. Now Publishers Inc., Hanover, MA, USA (2008)
3. Peter D. T.: Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews. In: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics (ACL '02), Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 417–424 (2002)
4. Benamara F., Cesarano C., Picariello A., Reforgiato D., V.S. Subrahmanian: Sentiment analysis: Adjectives and adverbs are better than adjectives alone. In: Proceedings of the International Conference on Weblogs and Social Media (ICWSM) (2007)
5. Peter R. R. White, J. R. Martin: The Language of Evaluation: Appraisal in English. Palgrave Macmillan, London/New York (2005)
6. Morales de Jesús, V. M.: Utilización de expresiones de actitud para el Análisis de Sentimientos. Tesis de Licenciatura, Benemérita Universidad Autónoma de Puebla, Puebla, México (2014)
7. Liu, B.: Sentiment Analysis and Opinion Mining. Vol. 5, Morgan & Claypool Publishers (2012)
8. Kai Ming T., Ian H. W.: Stacked Generalization: when does it work? (Working paper 97/03), Hamilton, New Zealand: University of Waikato, Department of Computer Science, pp. 866–871 (1997)
9. Kagan T., Joydeep G.: Linear and Order Statistics Combiners for Pattern Classification. In: Combining Artificial Neural Networks, Ed. Amanda Sharkey, Springer Verlag, pp. 127–161 (1999)
10. Witten, I. H., Mark A.: Data Mining: Practical Machine Learning Tools and Techniques. (3rd ed.) Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2011)
11. Duda, R.O., Hart, P.E., Stork D.G.: Pattern Classification. New York: John Wiley & Son (2001)
12. Fermín L. C., José A. T., Fernando E., F. Javier O.: Clasificación de documentos basada en la opinión: experimentos con un corpus de críticas de cine en español. Procesamiento del lenguaje natural, vol. 41, pp. 73–80 (2008)
13. Eugenio M. C., Martín V.: Opinion classification techniques applied to a Spanish corpus. In: Natural Language Processing and Information Systems, Vol. 6716, Springer Berlin/Heidelberg, pp. 169–176 (2011)

14. Bo P., Lillian L.: Opinion mining and sentiment analysis. Foundations and Trends in Information Retrieval, Vol. 2, Computer Science Department, Cornell University, Ithaca, NY, U.S.A., No. 1-2, pp. 1–135 (2008)

Reviewing Committee

Ricardo Acevedo
Moisés Alencastre Miranda
Roberto Alonso Rodriguez
Joanna Alvarado-Uribe
Gustavo Arroyo Figueroa
Christian Arzate
Ivonne Maricela Ávila Mora
Jorge Bautista López
Ivo Buzon
Maria Victoria Carreras
Felix Castro Espinoza
Noé Alejandro Castro Sánchez
Bárbara Cervantes
Jair Cervantes
Efren Chavez Ochoa
Gustavo Delgado Reyes
Sofia N. Galicia-Haro
Natalia Garcia
Alexander Gelbukh
David Gonzalez
Miguel Gonzalez-Mendoza
Hugo Gustavo González
Hernández Mario Graff
Fernando Gudiño
Pedro Guevara López
Yasmin Hernandez
Neil Hernandez Gress
Oscar Herrera
Rodolfo Ibarra
Yulia Ledeneva
Asdrubal Lopez Chau
Juan Carlos Lopez Pimentel
Ivan Adrian Lopez Sanchez
Antonio Marin-Hernandez
Lourdes Martínez
Martinez Medina
Miguel Angel Medina Perez
Efrén Mezura-Montes
Sabino Miranda-Jiménez
Daniela Moctezuma
Raul Monroy
Jaime Mora-Vargas
Saturnino Job Morales Escobar
Lourdes Muñoz Gómez
Antonio Neme
Alberto Oliart Ros
Mauricio Osorio
Elvia Palacios
Hiram Eredin Ponce Espinosa
Carlos Pérez Leguizamo
Maricela Quintana López
Carlos A. Reyes-Garcia
Carlos Alberto Rojas Hernández
Rafael Rojas Hernández
Dafne Rosso
Oscar S. Siordia
Grigori Sidorov
Abraham Sánchez López
Israel Tabarez Paz
Eric Sadit Tellez
Nestor Velasco-Bermeo
Elio Villaseñor
Alisa Zhila

Impreso en los Talleres Gráficos
de la Dirección de Publicaciones
del Instituto Politécnico Nacional
Tresguerras 27, Centro Histórico, México, D.F.
mayo de 2015
Printing 500 / Edición 500 ejemplares