# Análisis heurístico de datos y gestión de conocimiento

# 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 73,** mayo 2014. 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. Queda prohibida la reproducción total o parcial, por cualquier medio, sin el permiso expreso del editor, excepto para uso personal o de estudio haciendo cita explícita en la primera página de cada documento. 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 73**, May 2014. Printing 500. The authors are responsible for the contents of their articles.



Printed in Mexico City, in the IPN Graphic Workshop – Publication Office.

Volume 73

# Análisis heurístico de datos y gestión de conocimiento

**Miguel González Mendoza (Ed.)**


Instituto Politécnico Nacional, Centro de Investigación en Computación
México 2014

**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

# Prefacio

El propósito de este volumen es reflejar las nuevas direcciones de investigación y aplicaciones de los métodos de la Inteligencia Artificial en los campos de análisis heurístico de datos y de gestión de conocimiento.

Los artículos de este volumen fueron seleccionados con base en un estricto proceso de revisión efectuada por los miembros del Comité de revisión, tomando en cuenta la originalidad, aportación y calidad técnica de los mismos. Cada artículo fue revisado por lo menos por dos miembros del Comité de revisión del volumen (los revisores).

Este volumen contiene 13 artículos relacionados con varios aspectos del desarrollo de los métodos de Inteligencia Artificial y ejemplos de sus aplicaciones a varias tareas tales como:

- mapas de conocimiento,
- construcción automática de ontologías,
- detección de reutilización de código fuente,
- ambientes virtuales de aprendizaje,
- análisis basado en conocimiento de las empresas de software, invernaderos, bioreactores, dinámica de automóviles, fallas de motores, entre otras.

Este volumen puede ser interesante para los investigadores y estudiantes de las ciencias de la computación, especialmente en áreas relacionadas con la inteligencia artificial y su aplicación a los diferentes ámbitos de la vida cotidiana; así como, para el público en general interesado en estos fascinantes temas.

En este número especial de la revista RCS, a nombre de la comunidad del programa de Ingeniería en Computación del CU UAEM Zumpango expresamos nuestro agradecimiento al Dr. Jorge Olvera García, Rector de la Universidad Autónoma del Estado de México (UAEM) y al M. en A. Rodolfo Téllez Cuevas, encargado del despacho de la dirección del Centro Universitario UAEM Zumpango, por apoyar de manera ingente la investigación, y el desarrollo de la ciencia y la tecnología, sustentado todo ello en un humanismo que transforma.

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

Miguel González Mendoza
Mayo 2014

# Table of Contents

Page

# Validación de conceptos ontológicos usando métodos de agrupamiento

Mireya Tovar[1,2], David Pinto[2], Azucena Montes[1,3], Gabriel González[1], Darnes Vilariño[2], Beatriz Beltrán[2]

[1]Centro Nacional de Investigación y Desarrollo Tecnológico, Cuernavaca, México

[2] Facultad de Ciencias de la Computación, Benemérita Universidad Autónoma de Puebla, México

[3] Instituto de Ingeniería, Universidad Nacional Autónoma de México, México

{mtovar, amontes, gabriel}@cenidet.edu.mx, {dpinto, darnes,bbeltran}@cs.buap.mx

**Resumen.** En este artículo proponemos un enfoque para validar la información existente en una ontología de dominio mediante la identificación de conceptos sobre un corpus asociado. El mecanismo propuesto está basado en la determinación del grado de cercanía de los conceptos existentes en la ontología. Para llevar a cabo este proceso, inicialmente se representa cada concepto usando la información contextual, y posteriormente se usa dicha representación para agrupar la información obtenida. Dado que la matriz de representación suele ser de alta dimensionalidad, se usan técnicas de análisis semántico latente para reducir la dimensionalidad y permitir un proceso más eficiente de la información. El proceso de agrupamiento se lleva a cabo usando la técnica conocida como "agrupamiento por comités". Los resultados experimentales muestran un comportamiento satisfactorio para las dos ontologías revisadas en este trabajo.

**Palabras clave:** Ontologías de dominio, conceptos, análisis semántico latente, agrupamiento.

## 1. Introduccin

En los últimos años la representación del conocimiento y las ontologías han ganado importancia. Las ontologías se han utilizado en una gran diversidad de aplicaciones, entre las cuales podemos mencionar las siguientes: la comunicación de agentes [13], en el descubrimiento de servicios web [7], en sistemas de recuperación de información [8], en sistemas de pregunta-respuesta [1], y en el procesamiento del lenguaje natural [15].

Una ontología se define como "una especificación explícita y formal de una conceptualización compartida" [6]. En general, este tipo de recurso semántico está formado por conceptos o clases, relaciones, instancias, atributos, axiomas, restricciones, reglas y eventos. Las ontologías de dominio son un sistema de

representación del conocimiento que se pueden organizar en estructuras taxonómicas y ontológicas de conceptos de algún área o dominio de conocimiento específico. Análogamente, podemos decir que un corpus de dominio es aquel que está formado por textos de carácter específico.

El aprendizaje de ontologías o generación automática de ontologías, es un proceso que puede facilitar la construcción automática o semiautomática de las mismas. El término aprendizaje de ontologías se atribuye originalmente a Alexander Mädche y Steffen Staab [12] y se describe como la adquisición de un modelo de dominio desde los datos. El aprendizaje de ontologías necesita datos de entrada, como textos estructurados o no estructurados, desde los cuales se aprenden conceptos relevantes para un dominio específico, sus definiciones y relaciones establecidas entre estos. El aprendizaje de ontologías a partir de textos no estructurados se le conoce simplemente como "aprendizaje de ontologías desde textos" [4].

La construcción automática o semi-automática de ontologías es una área de trabajo ampliamente estudiada, sin embargo, la validación de la información contenida en los recursos obtenidos no lo es tanto. En la mayoría de los casos se asume que, por ejemplo, los conceptos encontrados son correctos en su mayoría. De ahí la intención de construir métodos que permitan validar la calidad de este tipo de recursos construidos.

En particular, en este trabajo estamos interesados en la identificación de conceptos de dominio en textos no estructurados. Partimos de la suposición de que los conceptos que están semánticamente relacionados, tienden a estar "cercanos" en un texto. Por lo tanto, un concepto se define como una idea que forma el entendimiento[1]. Desde el punto de vista de la filosofía, un concepto es una unidad de ideas que consiste de dos partes, la extensión y la intensión [9]. Cimiano [4] concibe la intensión como una definición no extensional de un cierto concepto o relación. Es decir, describir intuitivamente el significado de un concepto en lenguaje natural, como las glosas de recursos léxicos como WordNet [14]. La parte extensional es proporcionada por una base de conocimiento que contiene afirmaciones acerca de las instancias de los conceptos y las relaciones como se definen en la ontología. Por ejemplo, un "animal es un ser orgánico que vive, siente y se mueve por propio impulso"[2], una instancia para este concepto es "araña", una relación léxica entre un par de conceptos como mamífero y animal es, por ejemplo, hiperonímia ("un mamífero es un animal").

Para la extracción o descubrimiento de conceptos, algunos autores han considerado algoritmos como el análisis de conceptos formales (FCA, *Formal Concept Analysis*) y construyen jerarquías de conceptos al mismo tiempo [5]. Algunos otros autores han considerado enfoques de agrupamiento y consideran a los grupos de términos relacionados como conceptos [11], [18]. Otros aplican técnicas de reducción de dimensiones tales como el análisis semántico latente (LSA) [10], que revelan conexiones inherentes entre palabras, lo que conduce a la formación de grupos. En particular, el enfoque propuesto verifica los conceptos existentes

[1] Definición de concepto en la real academia española; (`http://www.rae.es`).
[2] Definición de animal en la real academia española; (`http://www.rae.es`).

en dos ontologías de dominio y en sus correspondientes corpus de dominio que están formados por documentos no estructurados. Para la identificación de conceptos se utiliza primeramente LSA para reducir la dimensionalidad de una matriz de representación, cuya versión reducida es introducida en un método de agrupamiento basado en comités (CBC, *Clustering By Committee*). Se parte de la hipótesis de que los conceptos que estan semánticamente relacionados, tienden a estar "cercanos" en un contexto y/o diferentes contextos.

El resto de este artículo se organiza como sigue: en la sección 2 se presenta el método LSA y algunos trabajos relacionados con la identificación de conceptos. En la sección 3 se muestra el algoritmo de agrupamiento basado en comités. En la sección 4 se presenta nuestra propuesta para la identificación de conceptos. En la sección 5 se muestran los resultados experimentales de la propuesta. Finalmente, las conclusiones se presentan en la sección 6.

## 2. Análisis semántico latente

El análisis semántico latente o *Latent Semantic Analysis* (LSA) es un modelo computacional utilizado en procesamiento de lenguaje natural, considerado en sus inicios como un método de representación del conocimiento [22].

LSA se considera una herramienta no supervisada de reducción de la dimensionalidad, como el análisis de los componentes principales (PCA, *principle component analysis*) [20]. Parte de la idea de que palabras en el mismo campo semántico tienden a aparecer juntas o en contextos similares [10], [21].

LSA tiene su origen en una técnica de recuperación de información llamada LSI (*Latent Semantic Indexing*) cuyo propósito es reducir la dimensión de una matriz de términos-documentos utilizando una técnica de algebra lineal llamada descomposición de valores singulares (SVD, *Singular Value Decomposition*). La diferencia con LSA, es que ésta utiliza una matriz de palabra-contexto. El contexto puede ser una palabra, una oración, un párrafo, un documento, un ensayo, etc.

Venegas [22] considera que LSA se caracteriza por ser una técnica matemático-estadística que permite la creación de vectores multidimensionales para el análisis semántico de las relaciones existentes entre los diferentes contextos.

El propósito de la reducción de la dimensionalidad es eliminar el ruido presente en las relaciones existentes entre los términos y los contextos, dado que generalmente es posible expresar el mismo concepto con distintos términos.

LSA no considera la estructura lingüística de los contextos, sólo las frecuencias de aparición y co-ocurrencia de los términos. Sin embargo, usando LSA se ha logrado en algunos casos identificar relaciones semánticas como sinonímia [10].

## 3. Agrupamiento por comités

El algoritmo de agrupamiento por comités (CBC, *Clustering By Committee*) permite descubrir automáticamente conceptos a partir de textos [11,18]. Inicialmente descubre un conjunto de grupos estrictos llamados comités que estan

dispersos en el espacio de similitud. El vector de características del grupo es el centroide de los miembros del comité y se procede a asignar elementos a sus grupos más similares.

El algoritmo CBC consiste de tres fases:

1. Encontrar los elementos más similares. Para calcular las palabras más similares de una palabra $w$, primero se ordenan las características de la palabra $w$ de acuerdo a su información mutua con $w$.
2. Encontrar los comités. Cada comité que se descubre en esta fase define uno de los grupos finales de la salida del algoritmo.
3. Asignar elementos a los grupos. Cada elemento se asigna al grupo que contiene al comité más similar.

CBC también se ha utilizado para encontrar los sentidos de una palabra $w$ [16] (algoritmo en su versión flexible), y para agrupamiento de textos (algoritmo en su versión fuerte) [17]. Otros autores, como Chatterjee y Mohan [3], han utilizado con éxito este algoritmo en su versión flexible para el descubrimiento de sentidos de las palabras, incluyendo además *Random Indexing* para disminuir la dimensionalidad de la matriz de contextos.

## 4. Enfoque propuesto para la identificación de conceptos

El enfoque que se propone en este artículo, para la identificación de conceptos de ontologías de dominio en sus correspondientes corpus de dominio, realiza los siguientes pasos:

1. Preprocesamiento del corpus de dominio y de las ontologías de dominio. El corpus de dominio se divide en oraciones y se eliminan palabras cerradas o vacías (como preposiciones, artículos, etc). El algoritmo de truncamiento de Porter [19] también se aplica sobre las palabras contenidas en estas oraciones. Los conceptos y relaciones de las ontologías son extraídos utilizando Jena[3]. El mismo proceso es aplicado a cada uno de los conceptos de la ontología con la finalidad de mantener consistencia en la representación terminológica (eliminación de palabras vacías y el algoritmo de truncamiento de Porter).
2. Aplicación del algoritmo LSA para disminuir la dimensionalidad de la matriz de contextos. En este caso, se utiliza del paquete S-Space[4], el algoritmo LSA[5]. El algoritmo recibe como párametros las oraciones del corpus de dominio, la lista de conceptos de la ontología y la cantidad $K$ de dimensiones (que en nuestro caso fue definida como 300). La salida del algoritmo LSA son vectores semánticos de dimensión $K$ para cada palabra o concepto identificado por LSA en el corpus.
3. Aplicación del algoritmo CBC en su versión flexible. La salida de LSA (entrada del algoritmo CBC) son las palabras y conceptos agrupados.

[3] `http://jena.apache.org/`
[4] `https://github.com/fozziethebeat/S-Space`
[5] http://code.google.com/p/airhead-research/wiki/LatentSemanticAnalysis

4. Identificación de los conceptos de la ontología en los grupos generados por el algoritmo de agrupamiento por comités.

En la siguiente sección se presentan los resultados obtenidos del enfoque propuesto.

## 5. Experimentos

En esta sección se presentan los datos utilizados y los resultados obtenidos en los experimentos. Primeramente mostramos la información asociada con los conjuntos de datos usados en los experimentos (ontologías y corpora asociado) para posteriormente mostrar los resultados obtenidos. El criterio de evaluación que se considera en la validación de conceptos es el de exactitud [2]. Es decir, se determina la cantidad de conceptos generados por el enfoque que existen en las ontologías de dominio.

### 5.1. Conjunto de datos

En los experimentos, por el momento, sólo se consideran dos ontologías debido a que están libremente disponibles y sus corpora contienen información asociada a los conceptos y relaciones que permiten afirmar o negar su validez. Como trabajo a futuro se considera incluir más ontologías y formar sus corpora correspondientes.

Los dominios de las ontologías son: inteligencia artificial (AI) y estándar e-Learning SCORM (SCORM)[6] [23].

Cada ontología contiene un número determinado de conceptos ($C$), relaciones tipo class-inclusion ($S$) y relaciones ontológicas ($R$). Los documentos ($D$) de los corpora de dominio asociados a cada ontología fueron utilizados para determinar la cantidad de palabras ($P$), el vocabulario (V) que excluye de $P$ las palabras vacías como artículos, preposiciones, etc., y el número de oraciones ($O$) (ver Tabla 1).

**Tabla 1.** Conjunto de datos

| Dominio | Ontología | | | Corpora | | | |
|---|---|---|---|---|---|---|---|
| | C | S | R | D | P | V | O |
| AI | 276 | 205 | 61 | 8 | 10,797 | 2,180 | 519 |
| SCORM | 1,461 | 1,038 | 759 | 36 | 32,572 | 2,154 | 1,779 |

[6] Las ontologías y sus corpora correspondientes están disponibles en la página http://azouaq.athabascau.ca/goldstandards.htm

### 5.2. Resultados

El algoritmo de agrupamiento CBC es capaz de obtener grupos de conceptos relacionados como los que se muestran en la Tabla 2 para el dominio de Inteligencia Artificial. Algunos conceptos relacionados con el concepto "Agent" son por ejemplo: *neural network, action*, que en la ontología son identificados como relaciones ontológicas, y *entire agent, reflex agent, abstract intelligent agent, individual agent* los cuales son identificados como relaciones tipo "class-inclusion".

**Tabla 2.** Ejemplo de conceptos agrupados por CBC.

| Concepto | Conceptos relacionados |
|---|---|
| Agent | Neural network, action, entire agent, reflex agent, abstract intelligent agent, rational agent, individual agent, planning problem, system, intelligent action, etc. |
| Artificial Intelligence | Strong ai, intelligence, field of science, human reasoning, etc. |

El total de vocabulario (palabras y conceptos) encontrado por el algoritmo LSA es de 1,537 para la ontología de Inteligencia Artificial (AI) y 2,049 para la ontología SCORM ($LSA\_V$). Los conceptos encontrados ($C\_Exactos$) en los grupos generados por el algoritmo CBC es de 222 de los 276 conceptos existentes en la ontología de AI, logrando así un 80 % de exactitud (ver Tabla 3). En el caso de la ontología SCORM, al utilizar el algoritmo LSA, sólo se logró identificar 539 conceptos de los 1,461 conceptos definidos en la ontología de dominio, obteniendo sólo el 37 % de exactitud.

**Tabla 3.** Resultados experimentales

| Dominio | C | $LSA\_V$ | $C\_Exactos$ | % |
|---|---|---|---|---|
| AI | 276 | 1,537 | 222 | 0.80 |
| SCORM | 1,461 | 2,049 | 539 | 0.37 |

En base a los resultados observamos que el método LSA logra identificar conceptos relacionados en cada ontología de dominio. El algoritmo de agrupamiento permite realizar una búsqueda más profunda de aquellos conceptos que están asociados con otros conceptos o relaciones de la ontología, por lo cual puede resultar útil en la tarea de validación.

## 6. Conclusiones

En este artículo se presenta un enfoque que permite validar conceptos de dos ontologías de dominio (inteligencia artificial y estándar e-Learning SCORM) a partir de la identificación de los mismos en un corpus de referencia asociado a

cada ontología analizada. El enfoque propuesto utiliza un algoritmo de reducción de la dimensionalidad de las características de cada concepto de las ontologías (LSA). Además se incluye un algoritmo de agrupamiento (CBC) que permite realizar una asociación más directa entre los conceptos identificados por el algoritmo LSA. En base a los resultados experimentales, se observa que la ontología de inteligencia artificial es más estable al encontrar por lo menos el 80 % del total de los conceptos. La ontología SCORM presenta una disminución en el cantidad de conceptos obtenidos por el enfoque, debido a que el algoritmo LSA sólo produce 2,049 palabras y/o conceptos de esta ontología que es una cantidad mucho menor comparada con los 1,537 que se obtienen para la ontología IA. También se observa que al analizar los conceptos asociados a cada concepto de la ontología se definen relaciones de tipo "class-inclusion" y ontológicas. Lo que permitiría extender las ontologías de dominio al realizar un análisis más profundo de cada concepto agrupado.

Como trabajo a futuro, consideramos revisar a profundidad las asociaciones existentes entre cada concepto agrupado y hacer una revisión de los conceptos que no fueron detectados para la ontología SCORM.

**Agradecimientos.** Este trabajo de investigación ha sido parcialmente financiado por el Consejo Nacional de Ciencia y Tecnología (CONACYT) con el número de becario 54371, por el Programa para el Mejoramiento del Profesorado (PROMEP) con número de convenio PROMEP/103.5/12/4962 BUAP-792 y a través del proyecto CONACYT 106625.

## Referencias

1. Beale, S., Lavoie, B., McShane, M., Nirenburg, S., Korelsky, T.: Question answering using ontological semantics. In: Proceedings of the 2Nd Workshop on Text Meaning and Interpretation. pp. 41–48. TextMean '04, Association for Computational Linguistics, Stroudsburg, PA, USA (2004)
2. Cantador, I., Ferández, M., Castells, P.: A collaborative recommendation framework for ontology evaluation and reuse. In: Actas de International Workshop on Recommender Systems, en la 17th European Conference on Artificial Intelligence (ECAI 2006), Riva del Garda, Italia. pp. 67–71 (2006)
3. Chatterjee, N., Mohan, S.: Discovering word senses from text using random indexing. In: Gelbukh, A.F. (ed.) CICLing. Lecture Notes in Computer Science, vol. 4919, pp. 299–310. Springer (2008)
4. Cimiano, P.: Ontology Learning and Population from Text: Algorithms, Evaluation and Applications. Studies in philosophy and religion, Springer (2006)
5. Cimiano, P., Hotho, A., Staab, S.: Learning concept hierarchies from text corpora using formal concept analysis. J. Artif. Int. Res. 24(1), 305–339 (Aug 2005)
6. Gruber, T.R.: Towards Principles for the Design of Ontologies Used for Knowledge Sharing. In: Guarino, N., Poli, R. (eds.) Formal Ontology in Conceptual Analysis and Knowledge Representation. Kluwer Academic Publishers, Deventer, The Netherlands (1993)

7. Ji, X.: Research on web service discovery based on domain ontology. In: Computer Science and Information Technology, 2009. ICCSIT 2009. 2nd IEEE International Conference on. pp. 65–68 (Aug 2009)
8. Jimenez Muñoz, R.J.: Un sistema de búsqueda semántica de información para su uso en el dominio de recuperación mejorada en yacimientos petroleros. Master's thesis, Fac. Ciencias de la Computación, BUAP, Puebla, Mex. (2013)
9. Krings, H. (ed.): Handbuch philosophischer Grundbegriffe. Kösel, München, studienausg. edn. (1973)
10. Landauer, T.K., Dutnais, S.T.: A solution to platoś problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. Psychological review pp. 211–240 (1997)
11. Lin, D., Pantel, P.: Concept discovery from text. In: Proceedings of the 19th International Conference on Computational Linguistics - Volume 1. pp. 1–7. COLING '02, Association for Computational Linguistics, Stroudsburg, PA, USA (2002)
12. Maedche, A., Staab, S.: Ontology learning for the semantic web. IEEE Intelligent Systems 16(2), 72–79 (Mar 2001)
13. Malucelli, A., Costa Oliveira, E.: Ontology-services to facilitate agents interoperability. In: Lee, J., Barley, M. (eds.) Intelligent Agents and Multi-Agent Systems. Lecture Notes in Computer Science, vol. 2891, pp. 170–181. Springer Berlin Heidelberg (2003)
14. Miller, G.A.: Wordnet: A lexical database for english. COMMUNICATIONS OF THE ACM 38, 39–41 (1995)
15. Nirenburg, S., Raskin, V.: Ontological Semantics. Language, speech, and communication, MIT Press (2004)
16. Pantel, P., Lin, D.: Discovering word senses from text. In: Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 613–619. KDD '02, ACM, New York, NY, USA (2002)
17. Pantel, P., Lin, D.: Document clustering with committees. In: Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. pp. 199–206. SIGIR '02, ACM, New York, NY, USA (2002)
18. Pantel, P.A.: Clustering by committee. Ph.D. thesis, University of Alberta (2003)
19. Porter, M.F.: An algorithm for suffix stripping. In: Sparck Jones, K., Willett, P. (eds.) Readings in Information Retrieval, pp. 313–316. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1997)
20. Sidorov, G.: Non-linear construction of n-grams in computational linguistics: syntactic, filtered, and generalized n-grams. Sociedad Mexicana de Inteligencia Artificial, Mexico (2013)
21. Vázquez Pérez, S.: Resolucin de la ambigedad semntica mediante mtodos basados en conocimiento y su aportacin a tareas de PLN. Ph.D. thesis, Universidad de Alicante (abril 2009)
22. Venegas V., R.: Análisis Semántico Latente: una panorámica de su desarrollo. Revista signos 36, 121 – 138 (00 2003)
23. Zouaq, A., Gasevic, D., Hatala, M.: Linguistic patterns for information extraction in ontocmaps. In: Blomqvist, E., Gangemi, A., Hammar, K., del Carmen Suárez-Figueroa, M. (eds.) WOP. CEUR Workshop Proceedings, vol. 929. CEUR-WS.org (2012)

# Enriquecimiento parcial de la consulta para la recuperación de información de alta calidad y flexibilidad

Alexander Gelbukh

Centro de Investigación en Computación (CIC),
Instituto Politécnico Nacional (IPN),
Av. Juan de Dios Bátiz s/n esq. Mendizábal,
Col. Nueva Industrial Vallejo, CP 07738, DF, México

www.gelbukh.com

**Resumen.** En la recuperación de información con alta calidad se tiene que tratar con los fenómenos de la equivalencia no literal de cadenas de letras, desde la inflexión y derivación morfológica de las palabras (*dormir*, *duermo*, *durmió*; *dormitorio*) hasta las relaciones semánticas de varios tipos entre las palabras (*computadora*, *informática*, *algoritmo*). En todos estos casos, es importante que dada una de estas palabras en la petición del usuario, se encuentren los documentos que contienen otras palabras del mismo conjunto. Por otro lado, para ciertas aplicaciones es importante que el usuario tenga buen control sobre qué palabras el sistema considera como equivalentes para cada petición específica. Técnicamente, el fenómeno se puede manejar en el momento de la indización de la base de documentos o en el momento de aplicación de la petición del usuario. En este trabajo explicamos por qué conviene tratar el problema a través de la expansión de la petición del usuario, y proponemos una técnica que drásticamente reduce el tamaño de la petición así obtenida.

## 1 Introducción

En nuestra sociedad de información, los procesos de manejo de conocimiento y en particular de recuperación de información textual juegan un papel crucial. Actualmente tenemos tanta información disponible en formato electrónico que encontrar los documentos que nos interesan se vuelve en un problema técnico muy complejo. En particular, los usuarios usualmente no saben con precisión qué palabras puede contener el documento que les puede intere-

sar, y sólo pueden formular su necesidad informática en palabras generales que describen el tema de su interés pero no un documento específico que buscan.

Consecuentemente, cualquier sistema de recuperación de información debe tratar de alguna manera el problema de la comparación aproximada entre las palabras clave de la petición de búsqueda del usuario y el documento. Por ejemplo, dado la petición computadora, el sistema debe ser capaz de recuperar (o no recuperar, ya dependiendo de los ajustes definidos por el usuario y las opciones de la petición del usuario) los documentos que contengan las cadenas de letras Computadora (con mayúscula), computadoras, computación, monitor, algoritmo, Internet, etc. Hay dos momentos en el flujo del procesamiento de datos por el sistema de recuperación de información donde este problema puede ser tratado. Por un lado, se puede tratar en el momento de la indización de los documentos, es decir, con el enriquecimiento del índice. Alternativamente, puede ser tratado en el momento del procesamiento de la consulta (así llamaremos la petición del usuario) específica, es decir, a través del enriquecimiento de la consulta.

La primera técnica se utiliza con más frecuencia debido a que los problemas de la segunda parecen prohibitivamente graves. Específicamente, en la segunda técnica, el enriquecimiento de la petición del usuario, en parecer requiere agregar a la petición millones de palabras relacionadas con las palabras indicadas por el usuario.

En este trabajo demostramos, sin embargo, que el primer método tiene sus propias desventajas, y que por otro lado los problemas del último método se pueden resolver de manera computacionalmente eficiente. Nuestra principal motivación en este trabajo ha sido el desarrollo de un sistema de recuperación de información para corpus reducidos. Específicamente, nuestro interés fue el desarrollo de la metodología de acceso computacional y recuperación de documentos del Corpus diacrónico y diatópico del español de América, denominado CORDIAM[1], el cual está en desarrollo en el marco de una colaboración entre la Academia Mexicana de la Lengua y un número considerable de instituciones lingüísticas de los países latinoamericanos. Este corpus consiste de la transcripción de los documentos históricos escritos en diferentes países y en diferentes tiempos que comprenden el período de los siglos XVI a XIX.

También hemos experimentado con el corpus de la legislación mexicana del Senado de la República Mexicana. La base del documento del Senado contiene las leyes de la República Mexicana, los proyectos de ley en estudio en las comisiones del Senado, los protocolos de las sesiones, los discursos de los senadores, etc.

[1] Véase www.CORDIAM.Gelbukh.com/acerca-de.

Sin embargo, consideramos que las técnicas expuestas en este artículo se aplican a ortos corpus y otras situaciones similares en la tarea de la recuperación de información. Otra motivación muy importante para nuestro trabajo son los sistemas de búsqueda dentro de la computadora, tales como *Microsoft Desktop Search* o el descontinuado sistema *Google Desktop Search*. Este tipo de sistemas proporcionan a los usuarios la búsqueda dentro de los archivos personales guardados en el disco duro de la computadora, documentos personales, correo y otras aplicaciones. Como ejemplo se puede considerar la función de búsqueda incluida en el programa de correo Outlook, parte de Microsoft Office. Esta función se beneficiaría mucho de las mejoras que proponemos en este artículo, ya que no proporciona nada parecido a éstas. Igual como en el caso de los corpus lingüísticos y jurídicos, la tarea de búsqueda dentro de la computadora personal muestra los rasgos que la hacen apropiada para nuestra metodología.

A saber, en el desarrollo de nuestra metodología nos guiamos por las siguientes prioridades, o sea, desiderata para el método de la recuperación de información a desarrollar [4]:

- La calidad, la fuerza expresiva y la flexibilidad de la búsqueda eran nuestras prioridades principales debido a la importancia de los resultados de la búsqueda para los usuarios, tales como los investigadores de la lingüística diacrónica o los abogados y los senadores. A diferencia de las tareas más comunes de la recuperación de información, en tales casos de uso cada documento es importante.
- Tamaño del índice reducido y razonablemente baja carga de mantenimiento en el servidor eran nuestras segundas prioridades, de acuerdo a los recursos computacionales disponibles para el sistema y el número de consultas que se espera.
- Cambios mínimos requeridos en la estructura de bases de datos de los corpus y en el mantenimiento de estas bases de datos que.
- El sistema debe estar en funcionamiento mientras que los diccionarios y los recursos lingüísticos correspondientes están en la fase de desarrollo y las mejoras y correcciones a los mismos debe surtir efecto de inmediato sin repetir el proceso de indización de toda la base de datos.
- La eficiencia computacional de procesamiento de una sola consulta es de menor prioridad ya que el número de usuarios esperado fue menos de él de los sistemas de recuperación de información en Internet.

Las propiedades características de la base de documentos para la cual se aplica la metodología propuesta son las siguientes:

- Tamaño mediano a grande, en el orden de un gigabyte, es cual puede extenderse a varios gigabytes,
- Contenidos especializados con variedad limitada de léxico y sintáctico,
- Construcciones lingüísticas sin restricciones con la posibilidad de uso ocasional de casi cualquier palabra o forma morfológica de palabra.

En este trabajo, estamos interesados en una solución flexible y computacionalmente eficiente, de implementación simple y mantenimiento fácil, al problema de la adaptación no literal de la petición de búsqueda de usuario en virtud de los requisitos y las circunstancias descritos más arriba.

El artículo está organizado de la siguiente manera. En la sección 2 damos una discusión muy breve del estado del arte en el problema abordado. En la sección 3 consideramos a detalle varios tipos de la comparación no literal de las cadenas de caracteres. En las secciones 4 y 5 discutimos dos posibles aproximaciones al problema de la comparación no literal entre la petición del usuario y el documento: la expansión del índice y la expansión de la consulta, así como sus ventajas y desventajas, llegando a la conclusión de que el enriquecimiento de la consulta tiene ventajas importantes sobre el enriquecimiento del índice. En la sección 6 proponemos un tratamiento del enriquecimiento de la consulta que subsana sus desventajas existentes. En las secciones 7 y 8 describimos la implementación de nuestro sistema y los resultados experimentales obtenidos con ella. Finalmente, en las secciones 9 y 10 discutimos el trabajo futuro —el cual consiste en la combinación de las dos aproximaciones— y presentamos las conclusiones.

## 2 Estado del arte

Existe un gran cuerpo de literatura sobre la modelación computacional del lenguaje natural [17] y en específico sobre la comparación no exacta de cadenas de caracteres. En la literatura existente se usa para esta tarea una gran variedad de estructuras de datos, tales como los denominados *tries*, los árboles B, etc. [1, 8, 11]. Estos trabajos se basan en patrones implícitos o explícitos que describen la similitud entre la cadena de origen y las cadenas con las cuales ésta se compara (por ejemplo, la distancia mínima de edición) o el conjunto de las cadenas que se buscan (por ejemplo, las expresiones regulares), a nivel de las letras individuales. Por ejemplo, un patrón de *com** (donde el asterisco indica una cadena arbitraria de caracteres) se utiliza para buscar todas las formas de la palabra española *comer*, aunque este patrón también coincidirá con al menos 172 otras palabras en español tales como *cometa*.

Sin embargo, en nuestro caso consideramos el problema de la coincidencia de conjuntos arbitrarios de palabras que pueden no compartir ningún patrón simple de letras. Por ejemplo, las palabras *dormía*, *duermo* y *durmiendo* son formas del mismo verbo español *dormir*. Las cadenas de caracteres, *iglesia*, *sacerdote* y *peregrino* representan el mismo concepto *la religión* a pesar de que no existe ningún patrón específico de letras que se pueda usar para identificar estas cadenas en el texto.

El problema de la generación y coincidencia de las formas de palabras en varios idiomas, incluyendo el español es muy bien estudiado en la lingüística. Varios métodos y estructuras de datos se han sugerido en la lingüística computacional para el manejo de los diccionarios correspondientes y las listas de morfemas [3, 12, 13]. Sin embargo, en este artículo no discutimos los problemas lingüísticos de la morfología del lenguaje natural, sino estamos interesados en la aplicación de un analizador morfológico a la tarea relacionada puramente con la gestión de bases de datos, a saber, la tarea técnica de la recuperación de una palabra clave en todas sus formas morfológicas.

En este trabajo suponemos que la lista de las formas de las palabras ya es conocida, es decir, es generada por un analizador morfológico previamente desarrollado, mientras que estas formas no necesariamente admiten un patrón simple de letras para su descripción.

Por otro lado, el uso de las llamadas ontologías y jerarquías de conceptos para el análisis de documentos es también una tarea muy bien estudiado. La tarea de análisis temático de documentos con el uso de ontologías se estudia en [9], y las técnicas correspondientes se aplican a las tareas de recuperación de información en [7]. Han sido recopilados varios tesauros jerárquicos grandes [2, 10, 14].

De manera similar al tratamiento de la coincidencia morfológica de cadenas de letras, aquí no nos interesa el manejo de los pesos estadísticos de la relación semántica entre conceptos ni la compilación del diccionario de conceptos sino la forma en que los documentos relevantes para un nodo específico de la jerarquía semántica pueden en la práctica ser encontrados en un sistema de información grande existente. Este tema no ha sido cubierto por la literatura sobre la construcción de las ontologías ni los lenguajes de descripción de las mismas.

## 3 Tipos de coincidencia aproximada de cadenas de caracteres

En esta sección analizamos con más detalle los tipos de cadenas de caracteres los cuales los usuarios necesitan coincidir en los procesos de la búsqueda

por la necesidad informática expresada de manera aproximada o general en la petición de búsqueda. Un punto importante en cada caso es el alto grado de flexibilidad necesaria para satisfacer las necesidades del usuario específico o efectuar una búsqueda específica.

### 3.1 Coincidencia entre mayúsculas y minúsculas

Es el tipo más simple de la coincidencia no literal de cadenas, y pudiera parecerse que no representa ningún problema para su tratamiento en un sistema computacional. Por ejemplo, usualmente las cadenas tales como *computadora, Computadora*, *COMPUTADORA* y *ComPuTaDoRa* deben ser consideradas como equivalentes. Los diseñadores de los sistemas de recuperación de información tienden que pensar que es obvio que antes de la indización de la base de datos todas las palabras se convertirían automáticamente a, por ejemplo, minúsculas.

Sin embargo, en determinadas circunstancias el usuario podría querer buscar una cadena específica, como *Ángel*, *Victoria*, *Gigante, PAN* (nombres de personas, topónimos, nombres de empresas), pero no *ángel*, *victoria*, *gigante, pan*, etc. O bien, un usuario puede, al revés, estar interesado en la palabra *pan* y no *PAN*.

Sin embargo, los motores de búsqueda actuales los cuales no hacen diferencia en su índice entre mayúsculas y minúsculas no proporcionan la posibilidad de especificar las mayúsculas y minúsculas en la petición de búsqueda —ni siquiera Google proporciona tal posibilidad. En nuestro caso, especialmente cuando se trata de los corpus lingüísticos tales como el CORDIAM, esta aproximación no es apropiada, dado que para los investigadores lingüistas es importante distinguir el uso de las mayúsculas y minúsculas en el corpus. Más específicamente, a veces es importante distinguirlas y a veces, al revés, esta diferencia se debe ignorar, dependiendo de las necesidades del usuario específico y la consulta específica. Con esto es claro que el tratamiento de las mayúsculas en nuestra metodología se debe posponer hasta la fase de búsqueda y no puede efectuarse en la fase de la indización de la base de datos.

### 3.2 Morfología y formas de palabras

La segunda clase importante de cadenas de caracteres las cuales con frecuencia —aunque no para cualquier búsqueda— se consideran equivalentes son las palabras formas del mismo lexema: *computamos*, *computábamos, computaremos,* o sus variantes derivativas: *computadora*, *computación, computacional, computabilidad*.

En las aplicaciones prácticas computacionales esta equivalencia se indica por el software lingüístico llamado analizador morfológico [12, 13]. Para cada palabra el analizador morfológico genera (posiblemente con ambigüedad) un identificador, por ejemplo, la primera forma, como *computar,* una base o raíz, como *comput-*, un número identificador, etc. Tal identificador es común para todas las formas morfológicas de la palabra, las cuales se consideran equivalentes para la búsqueda específica si el usuario así lo indica en las opciones de la búsqueda. Entonces la comparación aproximada de las dos cadenas de caracteres consiste en la reducción —llamada también *normalización*— de ellos a tales identificadores y luego la comparación exacta —ya no aproximada— de los resultados de tal reducción.

Existen analizadores morfológicos de diferente grado de complejidad, dependiendo de la precisión deseada y de si sólo se toma en cuenta el paradigma de inflexión dentro de un sólo lexema (*computamos / computaban*) o se debe tomar en cuenta también la derivación morfológica (*computar /computadora*) o semántica (*computadora / informática*). Algunos de los analizadores morfológicos dependen de la lengua específica de los documentos y otros son independientes del lenguaje.

En el caso simple, un analizador morfológicos utiliza una lista de terminaciones, tales como: *-ar, -amos, -ábamos, -adora, -abilidad* y una pequeña lista de excepciones, tales como *ir / fue*. Para los idiomas altamente inflexivos —como es el español— la lista de terminaciones puede ser considerablemente grande. Por ejemplo, en el sistema que hemos desarrollado en el marco de este trabajo en la actualidad se usan 3 mil 451 terminaciones.

Un analizador morfológico más sofisticado —y por lo tanto más preciso— utiliza patrones complejos y posiblemente depende de un diccionario general. Sin embargo, incluso un sistema basado en un diccionario debe contener un algoritmo heurístico para el manejo de las palabras ausentes en su diccionario, entre éstas, términos especiales, neologismos, nombres propios, etc.

Se debe tener en cuenta que los algoritmos morfológicos basados en heurísticas funcionan mucho mejor en el análisis —normalización— que en la síntesis —generación— de las formas de palabras correspondientes a una base morfológica dada. Por ejemplo, un simple algoritmo basado en lista puede normalizar *incomputabilidad* a *-comput-*, sin embargo, dado una base *-comput-*, es difícil para un sistema computacional no sofisticado elegir entre las variantes **incomputibildad*, **descomputabilidad,* etc.

Coincidencia de las formas morfológicas de la misma raíz en la aplicación de la petición de búsqueda no siempre es deseable para el usuario. Por ejemplo, el usuario puede estar interesado en las *computadoras*, pero no en la *computabilidad*. La reducción morfológica de las formas ambiguas, especialmente en

idiomas altamente inflexivos como el español, a veces produce efecto muy molesto. Por ejemplo, el verbo español *comer* tiene más de setecientas formas morfológicas, tales como *comiste* o *comiéndotelas*, una de los cuales, a saber, *como*, coincide con una conjunción muy frecuentemente utilizada *como*. Por lo tanto, si el usuario quiere encontrar los documentos con el lexema español *comer* con una precisión razonablemente alta, tendrá que sacrificar un poco el *recall* (recuerdo, el porcentaje de los documentos relevantes para el usuario que el sistema realmente encuentra) formulando la consulta como "buscar todas las formas de la palabra *comer,* pero no la forma como".

Nótese que este efecto no se puede lograr con una expresión lógica tan simple como "todos los documentos que contienen el lexema *comer* en cualquiera de sus formas morfológicas, excepto los que contienen el lexema *como*", ya que su significado no es equivalente a la búsqueda deseada. A saber, el *recall* de una consulta de este tipo sería extremadamente bajo ya que casi cualquier documento en español efectivamente contiene la cadena de caracteres *como*.

De este modo, otra vez llegamos a la conclusión de que el usuario debe poder controlar el hecho y el grado de la aplicación de la normalización morfológica que es aplicada a su consulta de búsqueda por el sistema —lo que los buscadores existentes tales como Google o *Microsoft Office Search 2013* no permiten. En particular, no será adecuado aplicar la reducción morfológica al momento de la indización de la base de documentos, como lo hacen los motores de búsqueda existentes.

### 3.3 Ontología o jerarquía de conceptos

La tercera clase de las palabras que el usuario podría necesitar que se consideren equivalentes son los sinónimos y variantes dialectales (*computadora / ordenador*), los hipónimos e hiperónimos (*computadora / dispositivo*) y posiblemente las palabras unidas por otras relaciones semánticas, tales como meronimia / holonimia, etc. Dado que ningún algoritmo puede inferir dichas relaciones entre las palabras por las correspondientes cadenas de caracteres, se utiliza un diccionario —por ejemplo, una ontología o un tesauro jerárquico— para proporcionar esta opción al usuario.

En nuestros experimentos hemos utilizado un diccionario de 33 mil palabras organizadas en una jerarquía profunda de conceptos, similar al tesauro de Roget [2] y en parte derivada de él. En cuanto a los conceptos relacionados, nos referimos no sólo a los conceptos con la relación de hiponimia (llamada *es-un*), sino también a otras palabras que son de interés para el usuario para la búsqueda de los documentos sobre un tema dado [9]. Por ejemplo, la entrada del diccionario para *religión* contiene tales palabras como *Biblia, sacerdote,*

*rezar, iglesia, peregrino,* etc. Por lo tanto, al usuario que busca los documentos sobre la *religión* el sistema le ofrece también los documentos que mencionan la *Biblia*. Opcionalmente, para medir la relevancia del documento encontrado el grado de dicha relación entre las palabras y los tópicos puede ser ponderado cuantitativamente [7].

Para este fin también se puede utilizar otros diccionarios y ontologías, siendo WordNet la primera opción que viene a la mente. Sin embargo, WordNet no contiene las relaciones semánticas tan ricas como el tesauro de Roget.

Obviamente, el usuario debe tener control total sobre el conjunto de las palabras que deben considerarse equivalentes a la palabra clave o a las palabras clave de la consulta. Las opciones mostradas en la ilustración 1 son de interés particular.

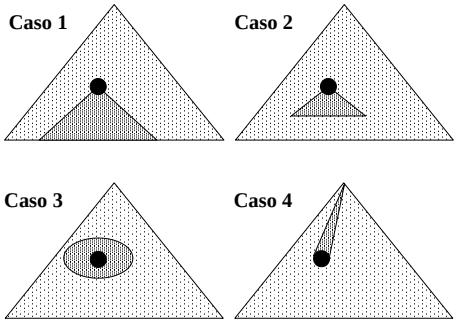



**Ilustración 1. Los tipos de vecindades en una ontología.**

En esta ilustración se muestran diferentes tipos de vecindades en una jerarquía semántica. La jerarquía se muestra en forma de triángulo, donde la punta simboliza la raíz de la jerarquía y la base simboliza los conceptos del más bajo nivel, siendo los puntos debajo de un punto dado sus hipónimos, los puntos al lado de él sus hermanos —cohipónimos— y los puntos arriba de él sus hiperónimos. Los cuatro tipos de vecindades se pueden caracterizar como sigue.

El caso 1 representa **todas las instancias** (hipónimos) de un concepto dado, es decir, todas las palabras por debajo de un nodo dado en la jerarquía. Por ejemplo, si el usuario elije este tipo de consulta, dada una consulta "¿qué hay de *matemáticas*?", el sistema recupera todos los documentos sobre *álgebra, geometría, cálculo* y dentro de estos temas, respectivamente, todo sobre la *álgebra lineal*, *teoría de grupos*, etc., *geometría diferencial*, *teoría de foliación*, etc., *cálculo diferencial*, calculo integral, etc., y finalmente recupera los documentos que mencionan las palabras *vector, variedad, diferencial,* etc. Otro ejemplo: ¿qué eventos ocurrieron en *Europa*? Esta consulta se debe interpretar

en de tal manera que se recuperen los documentos que citan *Inglaterra, Italia, Austria,* etc., *Londres, Manchester, Birmingham,* etc. En la ilustración 1, el conjunto correspondiente de palabras se muestra como todos los puntos debajo del punto indicado en la consulta, en este caso *Europa*.

El caso 2 representa los **hipónimos casi inmediatos** de un concepto dado, es decir, las palabras de abajo de un nodo dado, pero no más allá de un número $n$ de niveles. Por ejemplo, un estudiante puede querer saber lo que son las matemáticas: "Encontrar documentos sobre las *matemáticas* en general". En este caso, el sistema recuperará los documentos que mencionan las palabras como *algebra* o *geometría*, luego *ecuación*, *desigualdad*, *teorema*, pero no *isotropía* ni *semirretículo*, ya que estas últimas palabras son demasiado especializadas para ser útiles para una consulta general. Otro ejemplo: "¿Qué hay de la política de los *países europeos*?" En este caso, se deben recuperar sólo los documentos que mencionan a *Inglaterra, Italia, Austria,* etc. y, probablemente, *Londres, Roma, Viena,* pero no *Birmingham*, *Manchester*, *Wolverhampton*, etc.

El caso 3 representa los **conceptos similares,** es decir, las palabras que se encuentran en el árbol de conceptos no más de $n$ pasos de la palabra dada, ya sean los pasos para abajo (hipónimos), para arriba (hiperónimos), o en la dirección horizontal en el árbol (cohipónimos) [5]. Por ejemplo: "¿Qué disciplinas son similares a las *matemáticas*?" En este caso, las palabras relevantes son tanto *física* y *astronomía*, como *álgebra* y *geometría* o incluso *ciencia*, etc.

El caso 4 representa los **conceptos más generales,** es decir, hiperónimos: las palabras de las cuales el nodo dado es una instancia o un hipónimo. Por ejemplo: "¿En qué continente se encuentra *Morelia*?". En este caso, los documentos que podrían dar idea de este continente son los que contienen las palabras *Michoacán, México* o *América del Norte* (siendo Morelia una ciudad en el estado de Michoacán, México). Otro ejemplo: "¿Cuales derecho tiene un *Profesor Asociado*?". En este caso, los documentos útiles son los que mencionan los derechos de un *maestro*, *empleado*, *ciudadano*, *persona* o *humano*.

De hecho, las limitaciones indicadas en los casos de 1 a 4 a menudo tienen que combinarse. Por ejemplo, en una consulta de tipo 4, es útil imponer un límite sobre el número de niveles —como en la consulta de tipo 1— o sobre el nivel más general, ya que los conceptos demasiado generales aparecen en muchos documentos poco relevantes para una consulta específica y además es poco probable que provean conocimiento nuevo al usuario. O bien, las consultas de tipo 4 se pueden combinar con las restricciones de tipo 1 o 2. Por ejemplo, para la consulta "¿En qué continente se encuentra *Michoacán*?" se deben buscar tanto los conceptos más generales (como en el tipo 4) como los conceptos más específicos (como en el tipo 1).

Nótese que, al menos en la actualidad, el tipo deseado de la generalización de consultas no puede deducirse automáticamente por el sistema y debe ser elegido explícitamente por el usuario.

## 4 Enriquecimiento del índice

En la sección anterior hemos hablado de cuatro casos de la comparación aproximada de las cadenas de caracteres: con ignorar la diferencia en las mayúsculas y minúsculas, en las formas morfológicamente declinadas, en sinonimia y otras relaciones léxico-semánticas, y tomado en cuenta la estructura de un árbol de conceptos (las consultas de tipo 1 en la ilustración 1). Una aproximación ingenua —y la más utilizada— para representar los tres primeros casos de la comparación aproximada es *la reducción* del *índice*: al momento de la indización, todas las letras se reducen, por ejemplo, a minúsculas, todas las formas de las palabras o los conceptos derivados se reducen a la forma principal (*computando, computamos, computan, computación, computadora* → *computación*) y sinónimos y variantes dialectales son reemplazados por un representante elegido (*ordenador* → *computadora*). El último caso —un árbol de conceptos— puede ser manejado, además, por la indización de cada documento con los hiperónimos de las palabras las cuales este documento contiene (*PC* → *computadora, dispositivo, artefacto*). Con este método, con la consulta "*dispositivos*" se recuperará también *PC*.

Sin embargo, en este artículo proporcionamos los argumentos a favor de que esta aproximación simplista tiene serias desventajas. En primer lugar, como hemos mostrado en la sección anterior, en función de la relación de precisión vs. *recall* deseada, el usuario puede *no* querer que estas cadenas de caracteres se consideren idénticas. Por lo tanto, el proceso de indización no debe causar ninguna pérdida de información —es decir, todas las cadenas distintas de letras deben aparecer en el índice tal cual, sin cambio alguno, incluso en el caso de mayúsculas vs. minúsculas —es decir, las palabras no se deben reducir a minúsculas.

Para lograr esto, las cadenas reducidas a las letras minúsculas, reducidas morfológicamente, o bien reducidas con un diccionario de sinónimos deben aparecer en el índice además de las secuencias de letras originales y no en lugar de ellas. Por ejemplo, la palabra *Computadoras* debe causar la inclusión en el índice de las cadenas de letras *Computadoras, computadoras, computadora, dispositivo, artefacto*, etc. Dado que las nuevas palabras clave se agregan al índice en lugar de sustituir las palabras originales, este proceso se llama enriquecimiento del índice.

Para permitir cierta flexibilidad de las consultas, se pueden sugerir algunas mejoras a este esquema de indización. Por ejemplo, las palabras claves que se adicionaron se deben marcar de alguna manera para distinguirlas de las palabras originales. Por ejemplo, *Computadoras* → ORIGINAL: *Computadoras,* MINÚSCULAS: *computadoras,* MORFOLOGÍA: *computadora,* HIPERÓNIMO: *dispositivo,* HIPERÓNIMO: *artefacto*, etc.

Con esto, una consulta del usuario "exactamente *Computadoras*" se puede traducir internamente por el sistema a la consulta ORIGINAL: *Computadoras*; la consulta "la forma de palabra *computadoras*" se traduce a MINÚSCULAS: *computadoras*, lo que detecta tanto *Computadoras* como *computadoras* o *COMPUTADORAS*; la consulta "el lexema *computadora*" se traduce a MORFOLOGÍA: *computadora*, lo que detecta tanto *computadora* como *computadoras*. La consulta de tipo 1 para "*dispositivos*" se traduce a HIPERÓNIMO: *dispositivo*, lo que coincide tanto con *la unidad central* como con *impresora*. Otras mejoras serán presentadas en la sección 9 más abajo.

Sin embargo, el método de enriquecimiento del índice presenta algunos problemas, los cuales discutimos a continuación.

Primero, **falta de flexibilidad**. Sólo se pueden ejecutar los tipos de consultas para las que el índice fue diseñado específicamente. El usuario no puede elegir que las palabras de un determinado conjunto deben considerarse equivalentes, por ejemplo, "todas las formas de la palabra *comer* excepto *como* — véase la discusión de este ejemplo en la sección 3.2.

Segundo, el **índice más grande**. A diferencia de la reducción de índice, el método de enriquecimiento del índice puede aumentar el tamaño del índice muy significativamente —de dos veces a diez veces, dependiendo de la utilización de sólo morfología o también de un diccionario de sinónimos. En muchos casos, especialmente con bases de datos grandes, esta opción no es aceptable.

Tercero, **dificultad de mantenimiento**. El método del enriquecimiento del índice implica un demasiado fuerte acoplamiento del proceso de indización con el potencialmente complejo *lingware*, el software lingüístico tal como el analizador morfológico o el diccionario de sinónimos. Esto presenta al menos dos problemas técnicos y de organización.

Por un lado, el incluir el motor de búsqueda inteligente en una tecnología estable de mantenimiento de base de datos operativa que ya existe por un largo tiempo requiere cambios significativos en ésta última, lo que implica el cambio de software y la documentación existente, la formación o actualización de los ingenieros de mantenimiento, etc. A diferencia de esto, siempre es preferible conservar intacta la tecnología existente.

Por otro lado, a diferencia de los procedimientos estables de mantenimiento de las bases de datos, un *lingware* complejo basado en diccionarios tiende a

estar en constante desarrollo, al menos por un determinado período de tiempo inicial: nuevas palabras se añaden al diccionario, las tablas y algoritmos morfológicos se corrigen, se añaden nuevos enlaces al diccionario de sinónimos, etc. Ya sea con la reducción o enriquecimiento del índice, cada vez que se realiza un cambio en el *lingware*, toda la base de datos se va a volver a ser indizada. En muchos casos esto no es factible, sobre todo cuando el procesamiento lingüístico es lento y consume muchos recursos. Por otra parte, el posponer la reindización de la base durante mucho tiempo desalienta en gran medida cualquier mejora a *lingware*, desde el punto de vista tanto de los desarrolladores como los usuarios.

Estos argumentos nos indican que en el mejor caso el índice debe contener exactamente las cadenas de caracteres originales tal cual ocurren en los textos a indizar, sin alteración alguna.

## 5 Enriquecimiento de la consulta

Una alternativa viable al tratamiento de la comparación aproximada de las cadenas de caracteres al momento de la indización es su tratamiento en el momento del procesamiento de consultas de usuario.

Una aproximación ingenua a este método es la siguiente. Las cadenas de letras que se encuentran en los documentos se indizan tal cual, sin ninguna modificación. Luego, en el momento del procesamiento de la consulta del usuario, esta consulta se sustituye de forma automática por una expresión lógica apropiada. Por ejemplo, la consulta "*computar* y *matriz*" se ejecuta internamente como "(*computación* O *calcular* O *calculadas* O *informática*) Y (*matriz* O *matrices* O *matrices*)", donde O y Y son los operadores lógicos correspondientes. Este procedimiento se llamamos enriquecimiento de la consulta.

Este método no presenta ninguno de los problemas mencionados en el apartado anterior. Es decir, tiene las siguientes ventajas sobre el enriquecimiento o reducción del índice.

Primero, **flexibilidad**. El usuario puede editar la expresión resultante (por ejemplo, marcando o desmarcando las casillas de verificación junto a cada parte de la fórmula generada) para alcanzar cualquier combinación deseada. Por ejemplo, la consulta "todas las formas del verbo español *comer* excepto *como*" de forma fácil y natural puede ser expresada por el usuario y procesada por el sistema.

Segundo, el **índice más pequeño** en comparación con el enriquecimiento del índice. Sólo las cadenas literalmente presentes en el documento están presentes en el índice.

Tercero, **fácil mantenimiento**. El proceso de indización es trivial y no incluye ningún *lingware* ni depende de ningún *lingware*. No hacen falta ningunos cambios en la tecnología de indización no inteligente ya existente cuando se añade un motor de búsqueda inteligente a una base de datos ya operacional. No hace falta reindización cuando se realizan cambios en el *lingware*, y al mismo tiempo tales cambios están disponibles al usuario inmediatamente.

Sin embargo, las desventajas de este enfoque ingenuo son tan obvias que éste no se puede considerar como una alternativa viable en práctica. Los siguientes dos problemas hacen que tal método no sea factible.

Primero, **consultas demasiado grandes**. Como ya hemos mencionado, el verbo español *comer* tiene alrededor de 700 formas morfológicas, lo que resulta en la consulta enriquecida demasiado grande. Con un diccionario de sinónimos, el concepto *Europa* contribuiría a la consulta todos los países, ciudades, ríos, montañas, naciones, tipos de alimentos, etc. específicos para Europa. Además, cada una de estas cadenas debe capitalizarse de todas las maneras posibles.

Segundo, este método involucra la **generación lingüística**. Como ya hemos mencionado en la sección 3.2, la generación de todas las formas y derivadas semánticas de un lexema dado (*computar* → *computar*, *computación, incomputable*, *incomputabilidad*, etc.) es una tarea mucho más difícil que la tarea de análisis, es decir, de adivinar la forma principal correcta o la raíz de una determinada forma de la palabra (*computar*, *computación, incomputable*, *incomputabilidad* → *computar* o -comput-). En caso de un algoritmo morfológico basado en heurísticas, la cantidad de hipótesis en la generación de formas es usualmente mucho mayor que en la reducción de las palabras a la raíz morfológica.

En la siguiente sección presentamos cómo estos problemas pueden ser resueltos de manera práctica.

## 6 Enriquecimiento parcial de la consulta

La mejora que aquí sugerimos para el método de enriquecimiento de la consulta consiste en incluir en la consulta enriquecida sólo las cadenas que están presentes en al menos un documento de la base de datos sobre la cual se ejecuta la consulta. Ya que la diversidad lingüística en una base documental especializada (por ejemplo de textos jurídicos, médicos, etc.) es relativamente baja, sólo una pequeña fracción de todas las formas posibles de una palabra o subconceptos del concepto están presentes en la base de datos dada, lo que reduce en gran medida el tamaño de la consulta enriquecida. Al mismo tiem-

po, cuando se aplica a la base de datos específica, tal consulta parcialmente enriquecida es totalmente equivalente a la consulta enriquecida completa. Llamamos a esta modificación del procedimiento de enriquecimiento de enriquecimiento parcial.

El proceso de enriquecimiento parcial de consulta puede esbozarse de la siguiente manera.

Primero, se compila una lista de todas las cadenas que aparecen al menos una vez en la base de datos dada.

Segundo, esta lista relativamente pequeña es indizada como se describe en la sección 4, lo que produce una tabla de índices tales como los siguientes:

| Cadena | Identificador |
|---|---|
| *computadora* | computadora |
| *Computadora* | computadora |
| *COMPUTADORA* | computadora |
| *computación* | computar |
| *computable* | computar |
| *incomputabilidad* | computar |
| *PC* | computadora |
| *PC* | dispositivo |
| *PC* | artefacto |

Aquí, por el identificador nos referimos a una forma reducida, como en el caso de la reducción a las minúsculas, reducción morfológica a la forma principal, promoción vertical por el árbol en el diccionario de sinónimos e hiperónimos, etc., según lo descrito en la sección 3 (no mostramos en esta tabla las mejoras discutidas en las secciones 4 y 9).

Tercero, al momento de procesamiento de la consulta, cada palabra clave de la consulta es sujeto a un proceso de indización apropiado dependiendo de la opción definida por el usuario. Por ejemplo, éste puede ser la reducción morfológica a su forma principal, como en *calculable* $\rightarrow$ *calcular*, convirtiendo así la palabra a su potencial identificador. En caso de ambigüedad, se obtienen todos los identificadores potenciales.

Cuatro, los identificadores obtenidos para cada palabra clave de la consulta se buscan en la columna derecha de la tabla, y la palabra se sustituye con la lista de las cadenas correspondientes literales que se encuentran en la columna izquierda de la misma tabla.

Por ejemplo, con la tabla anterior, la pregunta "¿qué cosas son *computables*?" se transforma primero en la consulta "Identificador = computar" y después de la búsqueda en la tabla, en la consulta "*computación* O *computable* O *incomputabilidad*". Nótese que esta consulta parcialmente enriquecida no contiene tales cadenas de caracteres como *computar* o *computándose*. Ni siquiera contiene la forma de palabra *computables* de la misma consulta, ya que tal forma no se presenta en los documentos de la base de datos de nuestro ejemplo.

Para procesar una consulta compleja, tal como, por ejemplo, una consulta de tipo 3 como se discutió en la sección 3.3, el tesauro se recorre de manera correspondiente y la nueva consulta se construye primero como una disyunción de los lexemas o conceptos relevantes como se muestra en la ilustración 1. Luego la consulta de este tipo se enriquece aún más, o bien se filtra —lo cual puede incluso reducir su tamaño—, a través de la tabla de índice, como se describe más arriba.

La modificación que proponemos para el método de enriquecimiento de consulta no presenta ninguna de las desventajas del método original, y además presenta las siguientes ventajas en comparación con el enriquecimiento completo de consulta.

Primero, **consultas más pequeñas**. Sólo las palabras que realmente aparecen en la base de datos se incluyen en la consulta construida. La diferencia es especialmente sensible en el caso de las lenguas morfológicamente ricas tales como el español. Por ejemplo, de alrededor de 700 formas del verbo de uso frecuente comer, en la base de datos del Senado de la República Mexicana sólo aparecen 29, tales como *comiendose*, *comérselo*, etc. De más de 100 formas de *falsificar*, en la misma base aparecen sólo 11, tales como *falsificarla*, *falsificadas*, etc.

Segundo, uso de **sólo reducción**. El algoritmo no utiliza ningún tipo de generación sino sólo utiliza reducción —tal como la reducción a minúsculas o reducción morfológica. Esto en gran medida simplifica el *lingware*, lo que permite usar un bastante simple algoritmo del análisis morfológico basado en heurística.

Por supuesto, el método sugerido todavía tiene algunas desventajas en comparación con el enriquecimiento de consulta completo o los métodos de enriquecimiento de índice.

Primero, la **necesidad de mantener la lista de las cadenas** de caracteres. En comparación con el enriquecimiento de consulta completo, el método sugerido requiere mantener una estructura de datos adicional. Aunque en la siguiente sección vamos a demostrar que esto no presenta problemas serios de mantenimiento.

Segundo, todavía **aumenta el tamaño de la consulta**. En comparación con el enriquecimiento del índice, el tamaño de las consultas es mayor, aunque no a tal grado como con el enriquecimiento de la consulta completo.

Tercero, las **opciones pueden parecer extraño** al usuario. En comparación con el enriquecimiento de la consulta completo, la lista de las cadenas de caracteres que se presentan al usuario para la edición (véase la sección 5) puede parecer incompleta, sobre todo si el usuario no entiende cómo funciona el método y por qué algunas formas de la palabra, por ejemplo, *computan, computando* e *incomputabilidad* están presentes en la lista, mientras que otros, por ejemplo, *computar* o *computadas,* no están. Esto, sin embargo, no debería de ser un problema grave. Nótese que la lista no se puede completar con las palabras ausentes en la base de datos ya que el algoritmo morfológico basado en heurísticas que se utiliza para la reducción no está diseñado para la generación de todas las posibles formas de la palabra.

# 7 Arquitectura del sistema

En esta sección describimos la arquitectura del programa en el cual implementamos nuestra metodología para poder conducir experimentos en ella.

## 7.1 Actualización del índice

En la sección anterior, la necesidad de mantenimiento de la lista de cadenas y la tabla de índice fueron mencionadas como una fuente potencial de complicaciones o acoplamiento no deseable de la tecnología de indización con el *lingware*. Aquí vamos a mostrar cómo evitamos estos problemas en nuestra metodología. Hay dos fuentes potenciales de problemas:

- La actualización de la lista de cadenas caracteres y la tabla del índice cuando cambia la base de datos y
- actualización de la tabla del índice cuando cambia el *lingware*.

El último punto no presenta ningún problema real ya que la lista de las cadenas que se encuentran en la base de datos es muy pequeña en comparación con toda la base de datos. Así, la tabla del índice es simplemente reconstruida a partir de esta lista cada vez que el *lingware* se cambia, sin carga computacional significativa en el sistema.

El primer punto es sólo ligeramente más difícil. Para mantener la existente tecnología de mantenimiento de la base de datos independiente del módulo lingüístico que construye y utiliza la lista de las cadenas de caracteres, utili-

zamos un proceso independiente (un llamado agente computacional) el cual periódicamente, a intervalos de tiempo dependiendo de la carga actual del sistema, sincroniza la lista con la base de datos real. Hay dos maneras posibles de lograr dicha sincronización.

Con el primero método, el índice de la base de datos es accedido por el agente y enumerado en orden alfabético. El agente reconstruye la lista de palabras y la compara con la actual, así detectando que algunas palabras se han introducido y otras han desaparecido. Las entradas de las palabras desaparecidas se remueven de la tabla, mientras que las nuevas palabras se reducen —a las minúsculas, morfológicamente y con el diccionario de sinónimos— y se agregan a la tabla.

El otro método requiere una propiedad adicional del tipo booleano (lógico) del documento, "*indizado*", la cual se guarda en la base de datos junto con cada documento. Cuando se añade un nuevo documento a la base de datos, esta propiedad se establece en *falso*. El agente examina periódicamente la base de datos con la consulta "*indizado* = falso", recupera algunos de los documentos encontrados (dependiendo de la carga del sistema actual), extrae las cadenas de letras a partir de ellos, las agrega a la lista y la tabla si no están allá todavía y marca el documento como *indizado* = *verdadero*.

Los dos métodos tienen las siguientes ventajas y desventajas.

- El segundo método permite el tratamiento de los manejadores de las bases de datos como una caja negra, mientras que el primero requiere operación directa sobre sus estructuras internas.
- El primer método no requiere ningún cambio en la tecnología de mantenimiento de la base de datos utilizada antes de introducir la búsqueda inteligente, mientras que el segundo requiere un pequeño cambio en la estructura de la base de datos.
- El segundo método permite la indización de los documentos con las propiedades no relacionadas con las palabras individuales, sino más bien relacionadas con las combinaciones de palabras específicas o el conjunto de documentos completo, tales como el tema principal del documento [7, 9].
- El segundo método no proporciona una manera fácil para detectar documentos borrados y por lo tanto para eliminar del índice las palabras de tales documentos eliminados.

Esto último no es un problema grave ya que las cadenas que están presentes en la lista pero ausentes en la base de datos no afectan a los resultados de la búsqueda, aunque reducen el rendimiento del sistema. Una de las posibles soluciones a este problema es periódicamente —por ejemplo, una vez al mes— reconstruir toda la lista. Para ello, la propiedad *indizado* debe ser cambiada al

tipo de fecha en lugar de booleano. Para reconstruir la lista, si el proceso de reconstrucción se inició, por ejemplo, el 20 de mayo de 2014, el agente recupera los documentos con "*indizado* $<$ 20 de mayo de 2014", analiza los documentos encontrados y luego restablece la propiedad a "*indizado* = hoy".

### 7.2 Arquitectura general del sistema

Nuestro sistema está construido sobre la tecnología operativa existente tratada como una caja negra. El flujo de información es interceptado en los tres puntos siguientes.

Primero, la consulta del usuario es interceptada, analizada y sustituida por una consulta enriquecida mediante la técnica de enriquecimiento parcial de la consulta. La nueva consulta se presenta al usuario en un formato de uso fácil para una posible edición. Si es necesario, la consulta enriquecida se divide en una serie de consultas más pequeñas (véase más abajo).

Segundo, la respuesta del manejador de la base de datos es interceptada, analizada y —en caso de una consulta dividida en partes— una respuesta se compila de varios resultados de las consultas parciales.

Tercero, en los momentos de baja carga del sistema, el agente analiza periódicamente la base de datos para actualizar la lista de palabras y por lo tanto la tabla de sustitución utilizada para el enriquecimiento parcial de la consulta.

### 7.3 Enriquecimiento gradual

Para mejorar el rendimiento en los casos cuando la consulta enriquecida es demasiado grande, la consulta se enriquece sólo parcialmente con las palabras que están más estrechamente relacionadas con la consulta original del usuario. Por ejemplo, en la consulta del tipo 1 presentado en la sección 3.3, primero se efectúe la reducción a minúsculas y luego la reducción morfológica. Sólo en el caso de que una consulta así parcialmente enriquecida no resulta en un número suficiente de documentos encontrados, según lo especificado por el usuario, se efectúe el enriquecimiento de tipo 2 de la ilustración 1. Si esta consulta no es suficiente, se lleva a cabo el enriquecimiento completo de tipo 1.

Sin embargo tan pronto como la consulta parcial resulta en un número suficiente de los documentos recuperados (por ejemplo, diez) éstos se ordenan por relevancia y se presentan al usuario. Se lleva a cabo búsqueda adicional sólo si el usuario solicita más resultados. Con esta técnica, en la mayoría de los casos las consultas mucho más pequeños han demostrado ser suficientes.

Esta técnica se basa en la presuposición de que los documentos que contienen las palabras más cercanas a las palabras clave originales de la consulta del usuario (en la secuencia de minúsculas luego morfología luego ontología) son siempre más relevantes para el usuario. En realidad, el usuario debe poder controlar el orden exacto de la aplicación de las consultas parciales. Por ejemplo, en algunos casos la declinación morfológica podría ser más importante que la reducción a minúsculas: el *Estado* puede ser considerado más cercano a los *Estados* que a *estado* (cf. también el apartado 8.2).

## 8 Resultados experimentales

Hemos aplicado nuestra metodología a dos proyectos distintos: el buscador para el CORDIAM, Corpus diacrónico y diatópico del español de América, en el marco de colaboración con la Academia Mexicana de la Lengua, y el buscador de los textos legislativos para el Senado de la República Mexicana.

En particular, investigamos a detalle un subconjunto de 200 megabytes de la base de datos del Senado mexicano que contiene una mezcla representativa de los discursos de los senadores, leyes y otros documentos de trabajo del Senado. El corpus contiene 21 millón de palabras, de las cuales sólo 174 mil son diferentes (0.8%). Hemos reducido estas cadenas de caracteres de diversas maneras. Obviamente, la tasa de dicha reducción es igual a la tasa de expansión de la consulta cuando se usa enriquecimiento parcial de la consulta.

Primero, la reducción a minúsculas dio 102 mil cadenas diferentes, lo cual demuestra que con sólo tomar en cuenta la equivalencia de mayúsculas y minúsculas, el tamaño de la consulta se aumenta de manera no significativa, a saber, menos de dos veces. Los resultados para el enriquecimiento morfológico y el enriquecimiento basado en diccionario de sinónimos se discuten en las siguientes subsecciones.

### 8.1 Enriquecimiento morfológico de la consulta

Para nuestros experimentos hemos utilizado un procedimiento morfológico muy simple basado en una lista de todas las posibles cadenas de sufijos (sufijos morfológicos, terminaciones y clíticos) potencialmente usados en español, un total de 3 mil 578, por ejemplo: *-a*, *-aba*, *-abais*, *-abamos*, *-aban*, *-andoselas*, ..., *-eandoselo*, *-eandoselos*, *-eandoseme*, *-eandosenos*, ..., *-ismo*, *-ismos*, *-ista*, *-istas*, ..., etc. La reducción de una palabra consiste simplemente en remover tal sufijo. En caso de ambigüedad, todas las variantes posibles de reducción son considerados: *hablaba* → *hablab-* y *habl-*. Nótese que nuestra

reducción involucra sufijos significativas, por ejemplo, *comunismo*, *comunista*, *comunes* → *comun-*, lo que aumenta la tasa de enriquecimiento. Nuestra intención fue aumentar el *recall* con un método sencillo y robusto, sin usar diccionarios grandes.

Obviamente el método tan simplista que usamos produce cierto número de raíces incorrectas y a veces erróneamente considera diferentes palabras como si tuvieran una raíz común, por ejemplo, a *démosle* y *día* se les asigna una raíz común *d-*, véase más abajo. En las etapas posteriores del desarrollo de nuestro sistema, se usará un analizador morfológico basado en el diccionario [6] y además los sufijos significativos serán tratados en la ontología. El método basado en sólo en la lista de sufijos todavía se aplicará a las palabras ausentes en el diccionario morfológico. Esto mejorará aún más la proporción de expansión de consultas y la precisión de la búsqueda.

La reducción morfológica con nuestro método simple demostró que la base de datos utiliza 55 mil raíces diferentes. Por lo tanto, el número promedio de cadenas de caracteres por raíz —es decir, la tasa promedia del enriquecimiento parcial de consulta utilizando sólo la morfología— are aproximadamente de cuatro veces. En este proceso distinguimos las letras minúsculas y mayúsculas. Por ejemplo, la raíz *cultiv-* fue representada por tres cadenas de caracteres: *Cultiva, cultivo* y *cultivaron.*

Nótese que las "raíces" que forma nuestro método automático no necesariamente corresponden a las raíces lingüísticas de las palabras; además, las "palabras" que se encuentran en la base de datos no necesariamente son palabras correctas en español sino pueden ser erróneas o con errores de dedo. Con esto, el mayor número de cadenas (incluyendo erróneas) por raíz fue de 279 (la "raíz" *d-* según nuestro método simplista: *D, Dádme, Dé, Démos, Démosle, Démosles, Dénnos, Día, Días, Díza, DA, DADO, ..., duelo, duelos)*, el segundo mayor número fue 201 (la "raíz" *s-* según nuestro método simplista: *S, Sán, Sé, Sí, SA, SADAS, SAL, SALA, SALAS, SALES, SAN, ..., suelo, suelos*), luego 200 (*v-*), 190 (*c-*), 172 (*m-*), 171 (*p-*), 150 (*t-*), 140 (*r-*), 131 (*l-*), 125 (*est-: éstó, ésta, éstan, éstas, éste, éster, ésto, éstos, ESTA, ESTABLE, ESTADO, ESTADOS, ..., estira, esto, estos*), siendo este último el primer elemento de la lista que no fue de una sola letra. Para 183 raíces, es decir, sólo el 0.3% del total de las raíces representadas en la base de datos, el número de cadenas de letras por raíz fue mayor o igual a 50. El número total de cadenas correspondientes a estas raíces fue 12 mil, es decir, 7% del número total de diferentes cadenas de letras en la base de datos.

Como se puede ver, las palabras que causan una alta tasa de enriquecimiento de la consulta son las palabras cortas, en su mayoría formas de los verbos auxiliares, palabras con significado muy amplio o palabras incorrectamente

identificados por nuestro procedimiento como si fuera que tengan la misma raíz. Por lo tanto, aunque la tasa media del enriquecimiento parcial de la consulta con el procedimiento morfológico calculada por las cadenas de letras en nuestra base de datos es de 4, con la exclusión de las palabras con el significado muy amplio que no se utilizan en las consultas reales y la mejora del procedimiento morfológico esta cifra se disminuiría aún más. De hecho, en las consultas de los usuarios reales la tasa media que se observó (con nuestro procedimiento morfológico simplista basado en una lista de sufijos) fue alrededor de 3, que es un resultado muy alentador.

### 8.2 Enriquecimiento de consulta basado en ontología

Presentamos aquí dos ejemplos de consulta de enriquecimiento de la consulta a base de diccionario de sinónimos de tipo 1 según la ilustración 1. En el diccionario que usamos, el concepto *una ciudad mexicana* se compone de 2 mil 413 nombres. Cuando el nombre de la ciudad consiste de varias palabras, por ejemplo, *La Paz*, consideramos las dos cadenas de forma independiente, como si la lista incluyera ambas palabras *La* y *Paz*, lo que se tradujo en 2 mil 129 cadenas de letras (debido a las repeticiones de partes de los nombres). Sin embargo, la base de datos sólo menciona 1,130 de esas palabras si ignoramos la diferencia de mayúsculas y minúsculas, o bien 1,780 cadenas si distinguimos las mayúsculas y minúsculas.

De hecho sólo 1,077 de ellos eran los nombres de ciudades y el resto eran palabras que coinciden accidentalmente con el nombre de una ciudad o una parte de tal nombre, debido a la reducción a las minúsculas. Por ejemplo, la palabra *paz* coincide con una parte del nombre de la ciudad *La Paz*.

Este ejemplo demuestra una vez más la importancia del control por parte del usuario sobre los tipos de reducción que se aplican a la consulta: en este caso, la reducción basada en tesauro se debe aplicar sin reducción a minúsculas, a pesar de que esta última se percibe como más "básica" y usualmente se aplica de manera rutinaria sin siquiera ofrecer al usuario una opción de deshabilitarla. Con la técnica de enriquecimiento del índice, la decisión sobre el orden de aplicación de los tipos de reducción se toma en el momento de la indización —a pesar de que se puede hacer de una manera inteligente para cada palabra individualmente— y no puede entonces ser cambiada por el usuario.

El concepto *partes del cuerpo humano* en nuestro diccionario está representado por 97 palabras: *barba*, *barbilla*, ..., *mejilla*, *torso*, ..., *tripa*, etc. La base de datos sólo menciona 55 de ellos, o sea 86 si distinguimos mayúsculas y minúsculas. La mayoría de estas palabras fueron mencionadas en los discursos de

los senadores de estilo sonoro, por ejemplo: "¿Y será que ahora ponemos la otra mejilla?", "¡No vamos a caer de rodillas!".

Por lo tanto, con el enriquecimiento de la consulta a base de un diccionario de sinónimos, la tasa de expansión de la consulta es relativamente alta y podría no ser práctico para un sistema real. Sin embargo, como ya hemos mencionado, debido a la naturaleza de un diccionario de sinónimos que refleja el conocimiento "general" que a menudo resulta inadecuado para un usuario en particular, así como debido a la relativamente baja calidad de los diccionarios existentes, este tipo de enriquecimiento más que otros necesita el grado de control de usuario que no puede ser proporcionado por el método del enriquecimiento del índice.

Entonces, consideramos que la desventaja del método de enriquecimiento de la consulta es puramente técnica, es decir, temporal y fácilmente se subsana con mayor memoria y velocidad de las computadoras, mientras que su ventaja —mayor grado de control por parte del usuario— es fundamental y se refleja directamente en la calidad de los resultados de búsqueda. Nótese que a medida de que se está elaborando y expandiendo el diccionario la tasa de expansión de la consulta no se aumentará de manera significativa ya que las nuevas palabras que se añaden al diccionario son de baja frecuencia en los textos. En la siguiente sección, discutiremos una posible solución al problema de alta tasa de expansión de la consulta.

## 9 Trabajo futuro: una técnica combinada

A pesar de sus ventajas e incluso con la técnica propuesta, el enriquecimiento de consultas aún ralentiza el sistema debido a la inflación de la consulta del usuario, sobre todo en caso de enriquecimiento de la consulta a base de una ontología o un diccionario de sinónimos como se muestra en la ilustración 1. Por otro lado, el método del enriquecimiento del índice tiene otras ventajas, por ejemplo, la ventaja de poder utilizar el contexto de la palabra de una de las siguientes maneras.

Primero, las expresiones compuestas de varias palabras y las frases idiomáticas presentes en el diccionario de sinónimos, tales como *a duras penas* o *ser pan comido*, se pueden manejar de forma más natural en la etapa de la indización del texto completo del documento. Como una variante de las expresiones compuestas, planeamos usar el concepto de las llamadas n-gramas sintácticos [15], los cuales han mostrado buen comportamiento en otras tareas del procesamiento de texto [16]. Los detalles sobre la construcción de tales n-gramas se puede encontrar en el libro [18].

Segundo, las palabras se pueden desambiguar en el contexto: por ejemplo, con la consulta "*sobres*", el texto *la carta está en un sobre* se debe encontrar mientras que *la carta está sobre la mesa* no se debe proporcionar al usuario, siendo el contexto lo que da chance de desambiguar la categoría gramatical de la palabra.

Tercero, puede ser tomada en cuenta la estructura del documento. Por ejemplo, las palabras en el título o en el resumen de un artículo científico se pueden indizar de manera diferente a las palabras del texto principal.

Cuarto, se puede utilizar para la indización las propiedades globales del documento no relacionadas con ninguna palabra en específico en su texto, tales como el tema principal del documento [7, 9].

Para proporcionar estas mejoras sin sacrificar la flexibilidad del lenguaje de consulta, el enriquecimiento del índice puede ser utilizado en combinación con el enriquecimiento de la consulta. El primer paso para dicha combinación es el siguiente. Ambos métodos se implementan en el sistema; en particular, los documentos están indizados con el enriquecimiento del índice tal como se explica en la sección 4. Las consultas de los usuarios de los tipos estándares, tales como la reducción morfológica completa o una consulta basada en el diccionario de sinónimos de tipo 1 completo (véase sección 3.3), se procesan rápidamente con el índice enriquecido sin ningún enriquecimiento de la consulta. Por otro lado, en los casos relativamente poco frecuentes cuando el usuario modifica de alguna manera la lista de cadenas de caracteres las cuales para los efectos de esta consulta específica deben considerarse como equivalentes, se utiliza el enriquecimiento de la consulta.

La división del trabajo entre los dos métodos se puede optimizar. Por ejemplo, sólo los niveles profundos del tesauro pueden ser considerados para el enriquecimiento del índice (*matriz*, *ecuación*, desigualdad, etc. → HIPERÓNIMO: *matemáticas*, véase la sección 4), mientras que la jerarquía de nivel superior, si el usuario así lo indica, puede ser tomada en cuenta en forma del enriquecimiento de la consulta (*ciencia* → HIPÓNIMO: *matemáticas* O HIPÓNIMO: *física* O HIPÓNIMO: *química*).

Más aún, la forma en que los usuarios con mayor frecuencia modifican sus consultas puede ser aprendida y aplicada en el enriquecimiento del índice de forma automática, semiautomática o manual. Por ejemplo, si los usuarios con frecuencia no incluyen la forma *como* del paradigma morfológico del verbo español *comer* (véase la sección 3.2), entonces esta forma debe ser excluida del paradigma de este verbo en la fase del enriquecimiento del índice, mientras que la consulta con el paradigma completo (con la forma *como* incluida) será internamente —y de manera transparente para el usuario— tratado a través del enriquecimiento de la consulta como "MORFOLOGÍA: *comer* O MINÚSCULAS:

*como*" (obviamente aquí el "paradigma" morfológico del verbo *comer* no incluye la forma *como*; no estamos hablando del paradigma lingüístico real sino de una solución puramente técnica).

Además, el enriquecimiento de índice puede ser mejorado aún más cuando se utiliza en combinación con el enriquecimiento de la consulta. En la sección 4 hemos presentado las marcas para las palabras clave añadidas al índice durante el enriquecimiento, tales como MORFOLOGÍA, MINÚSCULAS, etc. Para permitir una mayor flexibilidad necesaria para las consultas de los tipos 2 y 3 basadas en un diccionario de sinónimos o una ontología (véase la ilustración 1), la distancia (en términos de los niveles) desde la palabra de origen hasta el concepto generalizado también se debe indicar en el índice.

Con esto, el ejemplo de la sección 4 se reescribe de la siguiente manera: *Computadoras* → ..., HIPERÓNIMO-1: *dispositivo,* HIPERÓNIMO-2: *artefacto*, HIPERÓNIMO-3: *objeto,* donde la notación HIPERÓNIMO-3 significa que la palabra *objeto* está a tres pasos en la ontología de la palabra original, en este caso siendo un hiperónimo del hiperónimo del hiperónimo. Ahora las consultas de tipo 2 "*dispositivos,* pero no más de 2 niveles hacia abajo" se implementan como un enriquecimiento de la consulta "HIPERÓNIMO-1: *dispositivo* O HIPERÓNIMO-1: *dispositivo*" y por lo tanto el sistema encontrará los documentos que contiene la palabra *Computadoras*, pero no *Compaq T100*, ya que éste último es un dispositivo demasiado específico y el usuario indicó que no está interesado en los hipónimos demasiado específicos.

Las consultas de los otros tres tipos se implementan de manera similar por la enumeración de los nodos correspondientes. Incluso si el usuario excluye algunas palabras o nodos de la subjerarquía, lo que genera una consulta no estándar, los nodos que mantienen intactos se pueden enumerar en la notación HIPERÓNIMO: ... en lugar de enumerar todas las palabras clave como se sugirió en las secciones 5 y 6 y como está implementado en nuestro sistema actual. Las palabras clave originales tales como HIPERÓNIMO: *computadora* se pueden mantener en el índice para ser utilizados sólo en el caso de las consultas del tipo más frecuente, es decir, tipo 1.

La técnica combinada aliviaría en gran medida el problema de la inflación de la consulta causada por el enriquecimiento de consultas, especialmente en el caso de las consultas basadas en el enriquecimiento a través del diccionario de sinónimos u ontología. Tiene la ventaja de mejor rendimiento debido a consultas más pequeñas, sin sacrificar la flexibilidad del lenguaje de consulta ni la calidad de las respuestas.

Obviamente, esto implica tanto ventajas y desventajas del enriquecimiento de índice. Específicamente, se da la ventaja de la posibilidad de considerar el contexto. Por otro lado, se introduce las desventajas metodológicas y técnicas

del enriquecimiento del índice mencionadas en la sección 4, por ejemplo los problemas de mantenimiento y acoplamiento indeseable del manejador de la base de datos con el *lingware*. Una investigación más profunda de la técnica combinada será la dirección de nuestro trabajo futuro.

## 10 Conclusiones

En este trabajo hemos considerado el problema de recuperación de información de las bases de tamaño mediano (no de tamaño de la web abierta) en la circunstancias que requieren alta calidad de la respuesta y consecuentemente gran flexibilidad del lenguaje de consulta. Para esto, analizamos dos aproximaciones al problema de comparación no literal de las palabras clave de la petición del usuario y del documento: el enriquecimiento del índice y el enriquecimiento de la consulta.

Hemos demostrado que la técnica usada de manera tradicional para tal tarea —la técnica llamada el enriquecimiento del índice— tiene desventajas inherentes importantes, y hemos propuesto una nueva técnica —el enriquecimiento parcial de la consulta— la cual permite una mayor flexibilidad de las consultas, mejor arquitectura general del sistema y un mantenimiento más sencillo del mismo.

Hemos probado nuestra metodología en el marco de dos proyectos: el buscador de los textos legislativos para el Senado de la República Mexicana y el buscador para el CORDIAM, Corpus diacrónico y diatópico del español de América, en el marco de una colaboración con la Academia Mexicana de la Lengua.

Nuestro método todavía tiene al menos dos problemas. Primero, las consultas enriquecidas construidas en algunos casos resultan significativamente más grandes y por lo tanto funcionan más lento. Segundo, no es obvio cómo en nuestro método se puede tomar en cuenta el contexto de la palabra clave para su desambiguación en el documento. Como una posible solución a estos dos problemas, hemos discutido la posibilidad de combinar los dos métodos: el enriquecimiento de la consulta y el enriquecimiento del índice, lo cual será nuestro trabajo futuro.

**Agradecimientos**. Este trabajo fue parcialmente apoyado por el Gobierno de México (SNI, SIP-IPN proyecto 20144534). En el trabajo se usaron los recursos léxicos y la experiencia de los proyecto apoyados por la Academia Mexicana de la Lengua (proyecto CORDIAM) y por el Senado de la República Mexicana.

## Referencias


1. Aho, Alfred V. *Algorithms for finding patterns in strings*. En: J. van Leeuwen (ed.), Handbook of Theoretical Computer Science, capítulo 5, pp. 254–300. Elsevier Science Publishers B. V., 1990.
2. Cassidy P. *An Investigation of the Semantic Relations in the Roget's Thesaurus: Preliminary Results*. En: A. Gelbukh (ed.), Computational Linguistics and Intelligent Text Processing, Proc. of CICLing 2000, February 2000, IPN, Mexico City, 2000.
3. Gelbukh, A. *Exact and approximate prefix search under access locality requirements for morphological analysis and spelling correction*. Computación y Sistemas, Vol. 6, N 3, 2003, pp. 167–182.
4. Gelbukh, A. *Lazy query expansion*. Computación y Sistemas, Vol. 6, No. 1, July-September 2002, p. 13–24.
5. Gelbukh, A. *Ontology-based Semantic Relatedness Measures: Applications and Calculation*. Research in Computing Science, Vol. 47, 2012, pp. 117–138.
6. Gelbukh, A., G. Sidorov. *Approach to construction of automatic morphological analysis systems for inflective languages with little effort*. En: Computational Linguistics and Intelligent Text Processing, Proc. of CICLing 2003. Lecture Notes in Computer Science, N 2588, Springer, pp. 215–220.
7. Gelbukh, A., G. Sidorov, A. Guzmán-Arenas. *Use of a Weighted Topic Hierarchy for Document Classification*. En: Text, Speech, Dialogue. Lecture Notes in Artificial Intelligence, N 1692, Springer, 1999.
8. Gusfield, Dan. *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology*. Cambridge University Press, 1997.
9. Guzmán-Arenas, A. *Finding the main themes in a Spanish document*. Expert Systems with Applications, Vol. 14, No. 1/2. Enero-febrero de 1998, pp. 139–148.
10. Fellbaum, C. (ed.) *WordNet as Electronic Lexical Database*. MIT Press, 1998.
11. Frakes, W., R. Baeza-Yates, editores. *Information Retrieval: Data Structures and Algorithms*. Prentice-Hall, 1992.
12. Hausser, Roland. *Three principled methods of automatic word form recognition*. En: Proc. of VEXTAL: Venecia per il Tratamento Automatico delle Lingue. Venice, Italy, Sept. 1999.

13. Koskenniemi, Kimmo. *Two-level Morphology: A General Computational Model for Word-Form Recognition and Production*. University of Helsinki Publications, N 11, 1983.
14. Lenat, D. B., R. V. Guha. *Building Large Knowledge Based Systems*. Reading, Massachusetts: Addison Wesley, 1990.
15. Sidorov, G. *N-gramas sintácticos no-continuos*. Polibits, vol. 48, pp. 67–75, 2013.
16. Sidorov, G. *Syntactic dependency based n-grams in rule based automatic English as second language grammar correction*. International Journal of Computational Linguistics and Applications, 4(2), 2013, pp. 169–188.
17. Sidorov, G. *Modelos formales en la lingüística computacional*. Universitat Autònoma de Barcelona, 2013, 220 pp.
18. Sidorov, G. *Construcción no lineal de n-gramas en la lingüística computacional: n-gramas sintácticos, filtrados y generalizados*. Sociedad Mexicana de Inteligencia Artificial, 2013, 166 pp.

# Herramienta de apoyo en la detección de reutilización de código fuente

Raymundo Picazo-Alvarez, Esaú Villatoro-Tello,
Wulfrano A. Luna-Ramírez, Carlos R. Jaimez-González

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

207363142@alumnos.cua.uam.mx,
{evillatoro, wluna, cjaimez}@correo.cua.uam.mx

**Resumen.** El acto de tomar parcial o totalmente contenidos generados por otras personas, y presentarlos como propios, sin dar el crédito correspondiente a los autores, es una forma indebida de reutilización de contenidos, considerada como plagio. Desafortunamente, en la actualidad, dada la amplia disponibilidad de contenidos a través de Internet, esta práctica se ha incrementado. La gran mayoría de los contenidos disponibles en la Web son materiales multimedia, aplicaciones y sobre todo textos, y todos ellos son susceptibles de plagio. En este artículo se hace énfasis en una clase de textos en particular: los programas escritos en algún lenguaje de programación, denominados código fuente. Dada la facilidad de acceso y las prácticas de reutilización de contenidos sin citar las fuentes (el abuso de la posibilidad de "*Copiar y Pegar*", derivado de deficiencias metodológicas o bien como acción deliberada), surge la necesidad de contar con herramientas para combatir el plagio, en especial, de código fuente. En el presente trabajo se propone una herramienta orientada a detectar la reutilización de código fuente en programas escritos en un mismo lenguaje de programación. Las técnicas aplicadas se basan en la detección de la similitud entre dos programas, a través del uso de su **Frecuencia de Términos** (TF) y su **Frecuencia Inversa** (TF-IDF), considerando como términos conjuntos de n-gramas de caracteres presentes en cada uno de ellos.

**Palabras clave:** $n$-gramas de caracteres, representación vectorial, similitud de documentos, reutilización de código fuente, procesamiento del lenguaje natural.

## 1. Introducción

La disponibilidad de grandes cantidades de información a través del acceso a Internet permite a millones de usuarios consultar información y materiales muy diversos. La cantidad de información accesible está en constante crecimiento, y se ha acelerado con la denominada Web 2.0, que permite a los usuarios la producción y publicación de materiales de distinta naturaleza. Esto ha sido posible, entre otras cosas, por la facilidad de reproducir y reutilizar contenidos ya existentes en formato digital. Sin embargo, muchas de estas reproducciones

son copias no autorizadas y la reutilización de una parte o su totalidad, frecuentemente conduce a prácticas de plagio y violación de las leyes de derechos de autor[1], pues no se citan las fuentes ni se toman en cuenta, pese a su existencia, las restricciones de autoría. Por esta razón, se han desarrollado distintas herramientas que intentan identificar la autoría de los materiales y, como consecuencia, el posible plagio de la información.

La tarea antes descrita enfrenta múltiples dificultades; en primer lugar, existen muchas definiciones de plagio, aunque es difícil nombrar a una como la más acertada [1]; sin embargo, coinciden en señalar que plagio es tomar ideas de otros y presentarlas como propias sin dar el crédito correspondiente al autor [1]. Un ejemplo de acciones de esta índole en el área de Tecnologías de la Información (TI), es el caso de la demanda entre empresas interpuesta por Oracle en contra de Google por plagio de código en el sistema operativo Android para su producto Smartphone[2].

En el ámbito universitario, en particular en las carreras de TI, merece especial atención, pues las prácticas de plagio son altamente lesivas para la actividad académica y debido a la abundancia de información disponible aunada a la facilidad de su reutilización, requiere el desarrollo de aplicaciones y procedimientos que las eviten. Lo anterior, dada su recurrencia y la afectación que provoca, motivó la realización de una herramienta académica auxiliar para la detección de la reutilización de código fuente monolingüe, es decir, dentro de un mismo lenguaje de programación (susceptible de ser evaluado como plagio a juicio del usuario experto, en este caso, un profesor de programación).

Diferentes autores han planteado variadas estrategias para detectar la reutilización de código fuente, como Faidhi y Robinson [2] en los que se basa parte del presente trabajo. Este último realizó una clasificación de seis niveles y/o tipos de plagio, según la dificultad que representa para el programador ocultar que está reutilizando código fuente. En la Figura 1 se muestran los diferentes niveles de plagio.

- **Nivel 0**. Es la copia exacta del código fuente.
- **Nivel 1**. Es la inserción, borrado, modificación de comentarios junto a la indentación, saltos de línea y espacios en blanco.
- **Nivel 2**. Se centra en los identificadores, ya sea cuando se cambian los nombres o se ponen en mayúsculas.
- **Nivel 3**. Consiste en cambiar de posición declaraciones, ya sean variables o funciones de una parte del código a otra; también añadir variables o funciones que no se usarán.
- **Nivel 4**. Resulta de la combinación y división de funciones.
- **Nivel 5**. Consiste en el cambio de estructuras de control del código fuente.
- **Nivel 6**. Realiza cambios en las expresiones contenidas en el programa.

La herramienta propuesta en este trabajo es capaz de identificar reutilización de código hasta el nivel 3. En este tenor, debe señalarse también que hay poca

[1] Disponible en: http://www.diputados.gob.mx/LeyesBiblio/pdf/122.pdf

[2] Disponible en: http://www.informationweek.com/software/operating-systems/oracle-appeals-android-lawsuit/d/d-id/1106694

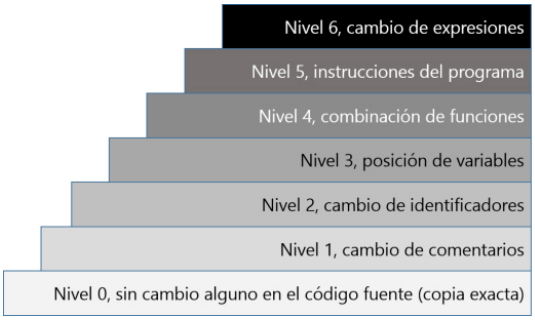



**Fig. 1.** Niveles de complejidad en la detección de plagio propuesta por Faidhi and Robinson [2].

información sobre desarrollos de herramientas en México para la detección de reutilización de textos en general y mucho menos de código fuente. De este modo, se propone contar con una herramienta desarrollada en nuestro país y contribuir así a su desarrollo tecnológico.

El resto del artículo está organizado de la siguiente manera. En la sección 2 se presentan algunos antecedentes y el estado del arte; la sección 3 describe el método propuesto en nuestra herramienta; la arquitectura de la herramienta se muestra en la sección 4. Finalmente, en la sección 5 se presentan las conclusiones, y el trabajo futuro.

## 2. Antecedentes y estado del arte

Los sistemas de detección automática de reutilización de código (que potencialmente es plagio) [1,3] nacieron debido a la imposibilidad de evitar el plagio por parte de organismos, tales como comisiones éticas en empresas y universidades. Por este motivo se desarrollaron varios modelos para la detección automática de reutilización de textos y de código fuente, los cuales pueden distinguirse en dos clases: la detección de plagio monolingüe, la cual detecta el plagio entre documentos pertenecientes al mismo lenguaje; y la detección de plagio translingüe, la cual es capaz de detectar el plagio entre varios lenguajes [4].

Estos modelos van más allá del análisis de códigos fuente completos para determinar si han sido escritos por un autor determinado, al analizar sus fragmentos, para intentar identificar que realmente fue escrito por el autor que lo presenta como propio [5].

En la literatura referente a la detección de reutilización existen dos enfoques que se han usado ampliamente para la detección de plagio: el **análisis intrínseco** y el **análisis extrínseco**. Los sistemas intrínsecos, tratan de identificar qué partes de un mismo código pertenecen a otro autor sin la necesidad de recurrir a fuentes externas. La idea intuitiva de estos sistemas es que cada autor/programador tiene estilos muy particulares, entonces, donde hay un cambio de estilo de programación se pre supone la existencia de un bloque que podría

pertenecer a otro autor. Por otro lado, los sistemas extrínsecos cuentan con una colección de códigos fuente confiables contra la cual se compara el código sospechoso. De esta manera, tratan de detectar si alguno de los códigos fuente confiables se han reutilizado o incluso si ha sido reutilizado el código completo de alguno o varios de ellos [6,7].

### 2.1. Técnicas para la detección de reutilización de código fuente

Existen dos técnicas principales en la comunidad científica para el análisis en la detección de reutilización de código fuente, las cuales se basan en ideas extraídas del área de Procesamiento de Lenguaje Natural, la cual es una sub disciplina de la Inteligencia Artificial. La primera de ellas, se basa en la comparación de características del propio código fuente, como son: el número de líneas, el número de palabras y caracteres, las líneas indentadas, los saltos de línea, la cantidad y tipos de *tokens*[3] de un código fuente, entre otras [6,7].

La segunda técnica, consiste en comparar la estructura del código fuente mediante un análisis sintáctico del documento. Algunos de los elementos del análisis son: las instrucciones, expresiones, asignaciones, identificadores, etc. Esta estructura se representa generalmente en forma de un árbol de ejecución; el cual se recorre en post orden, es decir, representando los nodos terminales (hojas) como 0 y los nodos internos (ramas) como 1. Este recorrido genera una cadena binaria que representa un perfil del árbol de ejecución; posteriormente con algoritmos de búsqueda de subcadenas comunes, se pueden identificar rápidamente partes en coincidentes entre dos árboles de ejecución representados de esta forma [6,7]. Esta técnica detecta la reutilización de código fuente a pesar de que se intente engañar al detector mediante técnicas de paráfrasis (la reformulación del fragmento reutilizado) o bien si se ha producido un resumen [4].

**Técnica basada en Bolsa de Palabras.**

Esta técnica utiliza vectores de características de código fuente, las cuales pueden ser palabras o $n$-gramas de palabras (tuplas de palabras o caracteres según su secuencia de aparición en el texto o en la palabra respectivamente). Una de las técnicas basadas en bolsa de palabras es el cociente de Jaccard, el cual consiste en dividir el tamaño de la intersección de dos vectores de características, entre el tamaño de su unión, como se muestra la Fórmula 1.

$$\mathcal{J}(A, B) = \frac{A \cap B}{A \cup B} \tag{1}$$

Donde $\mathcal{J}$ representa la probabilidad de reutilización del documento, $A$ al documento original y $B$ el documento sospechoso. $\mathcal{J}$ está acotado en [0, 1] siendo 1 cuando $A$ y $B$ son idénticos [4].

[3] Un token es una cadena de caracteres que tiene un significado coherente en cierto lenguaje de programación, por ejemplo podría ser una palabra clave como: *while, print, if, for.*

**Técnica basada en Huella Digital.**
El concepto de huella digital, consiste en generar, a partir de fragmentos de un documento, una representación a modo de resumen de lo que éste contiene. Un ejemplo es el uso de una función *hash*, la cual a partir de una secuencia de caracteres, obtiene un número único y que al realizar la mínima modificación de la secuencia, la función devuelve un número distinto [6,7]. A continuación se describen los pasos para obtener la huella digital de acuerdo al trabajo propuesto en [9].

1. Se divide el documento en $n$-gramas.
2. A cada $n$-gama, se le aplica una función que permita obtener un valor único. Para ello, generalmente se aplica una función *hash*.
3. De todo el conjunto de valores *hash* obtenidos, se selecciona un pequeño grupo que representa a todo el documento en el proceso de similitud. Cada valor del pequeño grupo seleccionado se vuelve una huella digital del documento.
4. Se elabora un índice invertido con las huellas digitales obtenidas. Para poder encontrar todos los documentos que poseen similitud con un documento $d_j$, primero se leen todas las listas invertidas de la huella digital del documento $d_j$; posteriormente se mezclan estas listas, y finalmente se aplica una regla de verificación especificada.

Como se puede observar, el primer paso es muy importante pues consiste en seleccionar $n$, es decir el tamaño de los $n$-gramas, nótese que un número muy grande, como el de todo el documento, permitiría únicamente detectar copias exactas; mientras que un tamaño más reducido, por ejemplo de un solo $n$-grama, terminaría indicando que todos los documentos son copias entre sí. En el segundo paso, se selecciona el subconjunto de valores *hash* que sean lo suficiente representativos de todo el documento [9], lo cual tampoco es una tarea trivial.

La técnica de huella digital es eficiente en el sentido que permite almacenar una pequeña porción del documento para el proceso de comparación, a la vez que, al no guardar el documento en sí, evita que los sistemas que lo implementen puedan ser empleados como fuente de plagio en sí mismas. Sin embargo, posee la desventaja de ser susceptible a pequeños cambios en la estructura de los códigos, por ejemplo remplazar un *for* por un *while* [2].

## 2.2. Herramientas para la detección de reutilización de código fuente

Existen algunas herramientas para identificar la detección de reutilización de código fuente [10], desarrolladas en otros países, pero muchas de ellas son poco amigables con el usuario, además de que no cuentan con su respectivo manual de usuario y su instalación suele ser compleja. Algunas de estas herramientas se revisan a continuación.

**JPlag**[4]. Es una herramienta no comercial, capaz de detectar reutilización multilingüe entre códigos fuente. Se empezó a desarrollar en el año 1996 mediante un

[4] http://plagiostop.wordpress.com/gratuito/jplag/

proyecto de investigación de la universidad alemana Karisruhe. En el año 2005 surgió como servicio web; para poder usar esta aplicación es preciso completar un registro y justificar que será utilizada por un profesor o investigador de alguna universidad o institución educativa.

**Sherlock**[5]**.** Es una herramienta de código abierto capaz de detectar reutilización entre documentos o códigos fuente en los lenguajes de programación Java y C. Fue desarrollada en la Universidad de Sidney; está implementada en el lenguaje de programación C; y utiliza firmas digitales para encontrar los fragmentos similares en el código fuente. Una firma digital es un número que está formado por varias palabras. Algunas desventajas es que no cuenta con interfaz gráfica; y al no utilizar el documento en su totalidad, no se puede afirmar que los documentos sean iguales.

**Simian**[6]. Es una herramienta de uso comercial, capaz de detectar reutilización multilingüe entre códigos fuente, tales como: Java, C, C++, COBOL, Ruby, JSP, ASP, HTML, XML, Visual Basic y texto natural. Fue desarrollado por una consultora de Australia llamada REDHILL. La herramienta esta implementada en el lenguaje de programación Java. No cuenta con interfaz gráfica.

**Tester SIM**[7]**.** Es una herramienta de código abierto, capaz de detectar reutilización multilingüe, algunos de los lenguajes que soporta: C, Java, Lisp, Modula2, Pascal y texto natural. La herramienta detecta fragmentos del código fuente potencialmente duplicados en grandes proyectos de software, no cuenta con interfaz gráfica y no muestra las partes reutilizadas del código fuente. Fue desarrollado por la Universidad de Ámsterdam.

**Moss**[8]**.** Es una herramienta no comercial, capaz de detectar reutilización entre documentos de texto y código fuente de varios lenguajes de programación, tales como: C, C++, Java, C#, Python, Visual Basic, JavaScript, Fortran, ML, Haskell, Lisp, Scheme, Pascal, Modula2, Ada, Perl, TCL, Matlab, VHDL, Verilog, Spice, MIPS assembly, a8086 assembly, a8086 assembly, MIPS assembly, HCL2. La herramienta utiliza la técnica de huella digital para encontrar la reutilización de código fuente; fue desarrollada en 1994; se puede utilizar a través de internet una vez estando registrado. Una desventaja es que no muestra las partes reutilizadas del código fuente.

[5] http://sydney.edu.au/engineering/it/ scilect/sherlock/
[6] http://www.harukizaemon.com/simian/index.html
[7] http://www.cs.vu.nl/pub/dick/similarity_tester/README.1ST
[8] http://theory.stanford.edu/ aiken/moss/

## 3. Método de detección propuesto

En esta sección se comenta el método de detección de reutilización de código fuente propuesto, iniciando con las fases de preprocesamiento y representación de documentos de código fuente, concluyendo con la descripción de la medida de similitud.

### 3.1. Preprocesamiento de los documentos de código fuente

El preprocesamiento consiste en eliminar del código fuente espacios en blanco, saltos de línea, la capitalización de las letras, convirtiendo todos los caracteres en minúsculas. La cadena resultante se divide en $n$-gramas de caracteres, ya que al dividirlo se pierde la localización espacial, y no importa si un programador sitúa una función que estaba al principio del código fuente original, al final o en cualquier parte del mismo; así, el conjunto de $n$-gramas de un mismo código o de un fragmento de él permanece igual pese a estos cambios de posición. El uso de $n$-gramas de caracteres ha mostrado ser una forma de representación útil para identificar el estilo de cada autor [8]. En este sentido, nuestro trabajo se basó entonces en las ideas propuestas por [6,7] con la finalidad de tener una representación que sea capaz de identificar estilos de programación.

### 3.2. Representación de los documentos de código fuente

La representación más comúnmente utilizada para representar cada documento es un vector con términos ponderados como entradas, concepto tomado del modelo de espacio vectorial usado en Recuperación de Información [11]. Es decir, un texto $d_j$ es representado como el vector $\overrightarrow{d}_j = \langle w_{kj}, \ldots, w_{|\tau|j} \rangle$, donde $\tau$ es el *diccionario*, *i.e.*, el conjunto de términos que ocurren al menos una vez en algún documento de $Tr$, mientras que $w_{kj}$ representa la importancia del término $t_k$ dentro del contenido del documento $d_j$. En ocasiones $\tau$ es el resultado de filtrar las palabras del vocabulario, *i.e.,* resultado de un preprocesamiento (sección 3.1). Una vez que hemos hecho los filtrados necesarios, el diccionario $\tau$ puede definirse de acuerdo a diferentes criterios. El criterio que se empleó en esta propuesta corresponde a Bolsa de Palabras (conocido como BOW del inglés Bag of Words), que es la forma tradicionalmente utilizada para representar documentos [12]. Este método de representación utiliza a las palabras simples como los elementos del vector de términos.

Con respecto al peso (*i.e.,* la importancia) $w_{kj}$, se tienen diferentes formas de calcularlo, entre las más usadas en la comunidad científica se tienen el ponderado booleano, ponderado por **frecuencia de término** y el ponderado por **frecuencia relativa de términos**. Una breve descripción se presenta a continuación:

- *Ponderado Booleano:* Consiste en asignar el peso de 1 si la palabra ocurre en el documento y 0 en otro caso.

$$w_{kj} = \begin{cases} 1, \text{ si } t_k \in d_j \\ 0, \text{ en otro caso} \end{cases} \quad (2)$$

- *Ponderado por frecuencia de termino (TF):* En este caso el valor asignado es el número de veces que el término $t_k$ ocurre en el documento $d_j$.

$$w_{kj} = f_{kj} \tag{3}$$

- *Ponderado por frecuencia relativa (TF-IDF):* Este tipo de ponderado es una variación del tipo anterior y se calcula de la siguiente forma:

$$w_{kj} = TF(t_k) \times IDF(t_k) \tag{4}$$

donde $TF(t_k) = f_{kj}$, es decir, la frecuencia del termino $t_k$ en el documento $d_j$. IDF es conocido como la "frecuencia inversa" del termino $t_k$ dentro del documento $d_j$. El valor de IDF es una manera de medir la "rareza" del termino $t_k$. Para calcular el valor de IDF se utiliza la siguiente formula:

$$IDF(t_k) = \log \frac{|D|}{\{d_j \in D : t_k \in d_j\}} \tag{5}$$

donde $D$ es la colección de documentos que está siendo representada en su forma vectorial.

### 3.3. Medida de similitud

Para el calculo de similitud entre dos documentos ($\overrightarrow{d_i}$ y $\overrightarrow{d_j}$) se han propuesto varias métricas que permiten determinar el parecido de éstos [13]. El objetivo de estas métricas es contar con un valor numérico al cual llamaremos *coeficiente de similitud SC*, el cual nos dirá cuán parecidos son los documentos $\overrightarrow{d_i}$ y $\overrightarrow{d_j}$. Una de las medidas ampliamente utilizadas en el campo de recuperación de información que permiten determinar la similitud entre documentos es la medida *cosenoidal*, la cual se describe a continuación.

**M**edida Cosenoidal. La idea básica de ésta es medir el ángulo entre el vector de $\overrightarrow{d_i}$ y de $\overrightarrow{d_j}$, para hacerlo, calculamos:

$$SC(\overrightarrow{d_i}, \overrightarrow{d_j}) = \frac{\sum_{k=1}^{t} w_{ik} w_{jk}}{\sqrt{\sum_{k=1}^{t} (w_{jk})^2 \sum_{k=1}^{t} (w_{ik})^2}} \tag{6}$$

Note que $k$ va de 1 a el número total de términos del vocabulario $\tau$, $w_{ik}$ indica la importancia del término $k$ en el documento $\overrightarrow{d_i}$ mientras que $w_{jk}$ la importancia del término $k$ en el documento $\overrightarrow{d_j}$.

## 4. Arquitectura de la herramienta

En esta sección se describe la arquitectura de la herramienta propuesta, la cual se muestra en la Figura 2. Durante el desarrollo de la herramienta, se utilizaron las siguientes tecnologías y lenguajes de programación: HTML, MySql, JSP, Java, CSS, Ajax y JavaScript. La herramienta desarrollada tiene un alcance de detección de reutilización de código hasta el nivel 3 (Sección 1) de complejidad. En las siguientes subsecciones de describen los tres módulos que componen la herramienta propuesta

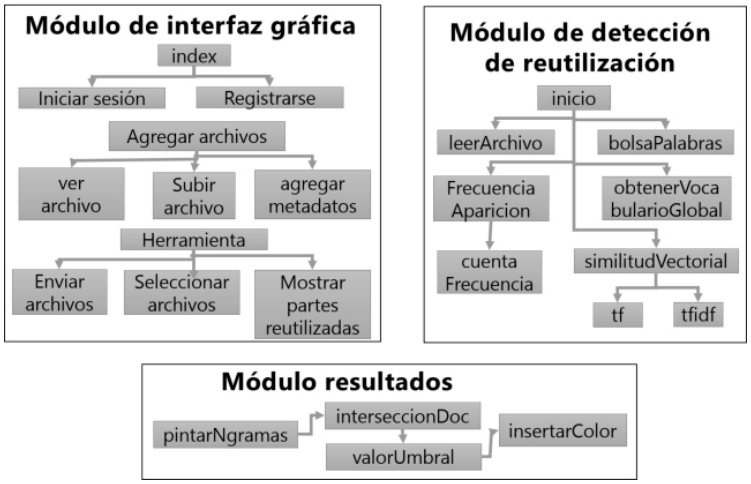



**Fig. 2.** Arquitectura del sistema propuesto. Note el diseño altamente modular, lo que permitirá en un futuro hacer modificaciones y/o extensiones de forma sencilla al sistema desarrollado hasta el momento.

### 4.1. Módulo de interfaz gráfica

Este módulo permite la interacción entre el usuario de la herramienta. Mediante éste, el usuario puede registrarse para realizar análisis de detección de reutilización de código fuente. También permite agregar los archivos que se quieran comparar y guardarlos en una base de datos para una futura utilización. La Figura 3 muestra un ejemplo de cómo seleccionar los archivos que se desean comparar.

### 4.2. Módulo de detección de reutilización de código fuente

Este módulo realiza la detección de reutilización de código fuente, tomando como entrada un archivo base y un archivo sospechoso que el usuario le proporciona al sistema mediante la interfaz gráfica (Sección 4.1). Una vez que el módulo recibe los archivos, realiza los pasos descritos en la Sección 3, y envía el resultado al Módulo de Interfaz Gráfica, para que sea mostrado al usuario, tal como se aprecia en la Figura 4.

### 4.3. Módulo de visualización de porciones reutilizadas

Este módulo se implementó para visualizar las partes reutilizadas entre un documento de código fuente base y un documento de código fuente sospechoso. Las partes del código fuente reutilizadas se verán resaltadas en tres colores diferentes que indican la frecuencia de los $n$-gramas encontrados, como se muestra en la Figura 5.

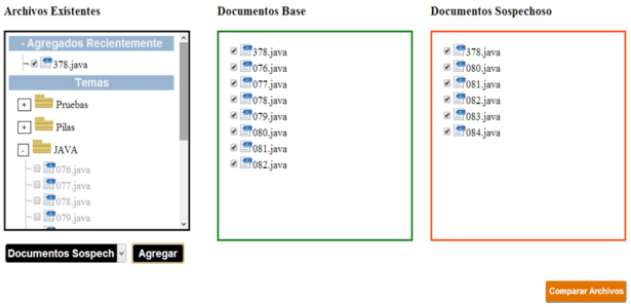



**Fig. 3.** Interfaz gráfica principal de la herramienta propuesta. En el recuadro más a la izquierda el usuario puede seleccionar de uno o varios directorios los archivos que desea analizar. Al momento de hacer esta selección el sistema pregunta al usuario cuáles son los archivos sospechosos de plagio y cuáles son los archivos originales, colocando los sospechosos en la ventana de la extrema derecha y los originales en la ventana de en medio.

| Archivos Sospechoso | Archivos Base | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 378.java | 378.java 100.0% | 076.java 36.0% | 077.java 51.0% | 078.java 29.0% | 079.java 43.0% | 080.java 49.0% | 081.java 21.0% | 082.java 21.0% |
| 080.java | 378.java 49.0% | 076.java 44.0% | 077.java 63.0% | 078.java 42.0% | 079.java 57.0% | 080.java 100.0% | 081.java 19.0% | 082.java 38.0% |
| 081.java | 378.java 21.0% | 076.java 12.0% | 077.java 18.0% | 078.java 13.0% | 079.java 14.0% | 080.java 19.0% | 081.java 100.0% | 082.java 14.0% |
| 082.java | 378.java 21.0% | 076.java 25.0% | 077.java 31.0% | 078.java 26.0% | 079.java 28.0% | 080.java 38.0% | 081.java 14.0% | 082.java 100.0% |
| 083.java | 378.java 45.0% | 076.java 44.0% | 077.java 65.0% | 078.java 39.0% | 079.java 62.0% | 080.java 61.0% | 081.java 17.0% | 082.java 45.0% |
| 084.java | 378.java 39.0% | 076.java 29.0% | 077.java 46.0% | 078.java 30.0% | 079.java 33.0% | 080.java 51.0% | 081.java 24.0% | 082.java 27.0% |

**Fig. 4.** Resultados del módulo de detección de reutilización de código fuente. Con la intensión de facilitar al usuario la tarea de identificar códigos reutilizados, los resultados se muestran en forma de una matriz de similitudes, siendo las filas los archivos sospechosos y las columnas los archivos marcados como originales en la interfaz principal (Figura 3). Note que en las celdas de la matriz aparece el nombre del archivo junto con su porcentaje de similitud, así entonces porcentajes altos significa que hay una alta similitud entre los archivos correspondientes.


**Fig. 5.** Código de colores empleado para resaltar las partes reutilizadas. El color más intenso (rojo) significa una coincidencia alta, mientras que el color menos intenso (amarillo) representa una reutilización baja.

Una vez que el usuario selecciona cualquiera de los archivos de la matriz de resultados (Figura 4), la herramienta le muestra el código base y el código sospechoso resaltados de acuerdo al código de colores explicado en la Figura 5. La Figura 6 ilustra un ejemplo del resaltado de secciones reutilizadas entre un par de códigos.

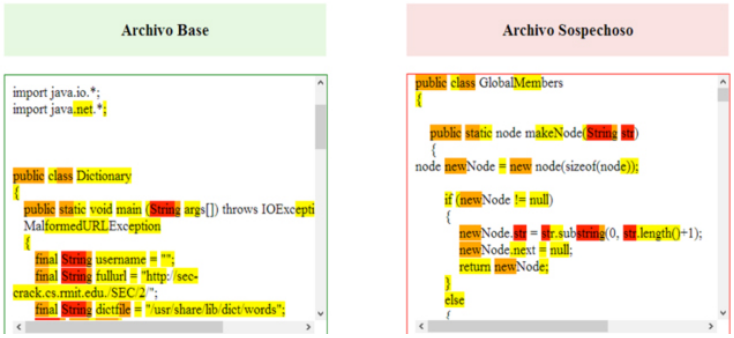



**Fig. 6.** Ejemplo de la visualización de secciones reutilizadas empleando el código de colores mostrado en la Figura 5. Note que las palabras reservadas tienden a ser las secciones del código que suelen tener altas coincidencias. Sin embargo, este tipo de visualización ayuda en gran medida al experto a identificar de manera rápida las posibles secciones reutilizadas, proporcionándole más elementos al momento de emitir un juicio de plagio o no-plagio entre uno más códigos fuente.

Es importante mencionar que para definir cuando un $n$-grama es de frecuencia alta, intermedia o baja no se utilizó un umbral fijo de frecuencia. En su lugar, el sistema propuesto calcula al vuelo cuando una frecuencia es alta, intermedia o baja tomando en cuenta el peso TF-IDF de los $n$-gramas que conforman el vocabulario del par de códigos que se están comparando en determinado momento.

## 5. Conclusiones y trabajo futuro

En este trabajo se presenta una herramienta para la detección de reutilización de código fuente, sin precedente en nuestro país. La herramienta aún necesita probarse con un mayor número de usuarios y códigos fuente.

Hasta el momento se ha observado que los comentarios, palabras reservadas, caracteres repetitivos (como puntos, comas, y los signos propios de los lenguajes de programación) interfieren en la detección al elevar el porcentaje de similitud con la medida adoptada, sin embargo, al hacer pruebas cualitativas contra una de las herramientas más populares (*i.e.,* JPLAG) notamos que casos de reutilización alta no son detectados por Jplag y si por el sistema propuesto. Con la intención de validar el sistema desarrollado se planea como parte del trabajo futuro conseguir un conjunto de colecciones estándar con las cuales sea posible determinar el grado de efectividad de la herramienta desarrollada.

Agregado a lo anterior, como parte del trabajo a futuro también se prevee extender el enfoque que realice un análisis de código con base en $n$-gramas de palabras, además de probar otras técnicas de comparación de código y emplear diversas medidas de similitud. Finalmente, es conveniente mencionar que la herramienta desarrollada en este trabajo estará disponible en el sitio del Grupo de Lenguaje y Razonamiento de la UAM-C cuya página es: http://lyr.cua.uam.mx

**Agradecimientos.** Agradecemos el apoyo otorgado por la Universidad Autónoma Metropolitana Unidad Cuajimalpa y el SNI-CONACyT.

## Referencias

1. Sánchez Vega, J.F.: Detección automática de plagio basada en la distinción de fragmentación del texto reutilizado. Tesis de Maestría. Instituto Nacional de Astrofísica Óptica y Electrónica (2011)
2. Faidhi,J.A.W., Robinson, S.: An empirical approach for detecting program similarity and plagiarism within a university programming environment. Computers and Education, 11(1), pp.11–19 (1987)
3. Sánchez-Vega, J.F., Villatoro-Tello, E., Montes-y-Gómez, M., Villaseñor-Pineda, L., Rosso, P.: Determining and characterizing the reused Text for Plagiarism Detection. Expert Systems with Applications, 40(5), pp. 1804–1813 (2013)
4. Franco Salvador, M.: Detección de plagio translingüe utilizando una red semántica multilingüe. Tesis de Maestría, México D.F., México (2013)
5. Barrón-Cedeño, A.: Detección automática de Plagio en Texto. Tesis de Doctorado. Valencia, España (2012)
6. Flores, E., Barrón-Cedeño, A., Rosso, P., Moreno, L.: Towards the detection of cross-language source code reuse. In: Proceedings of the 16th International conference on Applications of Natural Language to Information Systems (NLDB), pp. 250–253 (2011)
7. Flores, E., Barrón-Cedeño, A., Rosso, P., Moreno, L.: DeSoCoRe: Detecting source code re-use across programming languages. In: Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstration Session, NAACL, pp. 1–4 (2012)

8. Frantzeskou, G., Stamatatos, E., Gritzails, S., Katsikas, S.K.: Source Code Author Identification Based On N-gram Author Profiles. pp. 508–515 (2006)
9. Alva Manchego, F.E.: Sistema de información de detección de plagio en documentos digitales usando el método document fingerprinting. Tesis de Maestría. Peru (2010)
10. Herramienta. Sitio web describiendo varias herramientas para la detección de plagio en Código Fuente. (Noviembre 2013). http://www.linti.unlp.edu.ar/uploads/docs/herramientas_para_la_deteccion_de_plagio_de_software_un_caso_de_estudio_en_trabajos_de_catedra.%20Un%20caso%20de%20estudio%20Anhi.pdf
11. Baeza-Yates, R., y Ribeiro-Neto, B. Modern Information Retrival. Addison Wesley (1999)
12. Sebastiani, F.: Machine Learning in Automated Text Categorization. In: ACM Computing Surveys, 34(1), pp. 1–47 (2002)
13. Grossman, D.A., Frieder, O.: Information Retrieval, Algorithms and Heuristics. Springer (2004)

# API de Google Maps para un mapa de conocimiento de los asesores especializados de un Centro de Desarrollo Empresarial

José Sergio Ruiz Castilla, José Antonio Díaz García, Jair Cervantes Canales

Centro Universitario UAEM Texcoco,
Universidad Autónoma del Estado de México,
Estado de México, México

jsergioruizc@gmail.com, ittonzs@gmail.com, chazarra17@gmail.com

**Resumen.** Existen sitios dedicados a asesorar a emprendedores y micro empresarios en temas de compras, producción, innovación, ventas, finanzas, etc. Tanto en México, como en otros países, existen los CDE (Centros de Desarrollo Empresarial) otros denominados Incubadoras de Empresas, donde se brinda apoyo a los emprendedores que inician una nueva empresa. En la Universidad Autónoma del Estado de México, se creó una red de Incubadoras de Empresas en los Centros Universitarios, Una de estas unidades se encuentra en el Centro Universitario UAEM Texcoco, donde se enfocó este proyecto. Una de las misiones de los CDE es ofrecer y llevar a cabo asesorías ante los problemas y retos de las empresas nacientes. Para lo anterior se creó una aplicación Web para administrar el conocimiento a través de un Mapa de Conocimiento. Los usuarios pueden consultar la existencia de conocimiento requerido; si existe, lo podrán visualizar en un mapa con ayuda de la API (Interfaz de Programación de Aplicaciones) de Google Maps. Para su uso, basta un dispositivo con conexión a internet. Esta aplicación busca apoyar a emprendedores y empresarios que tiene las preguntas: *¿El CDE Tiene el conocimiento qué requiero? ¿Quién tiene el conocimiento? ¿Dónde están los portadores del conocimiento? ¿Cómo y dónde los contacto*? De existir el conocimiento en el mapa bastará contactar a algún portador y establecer una modalidad para la asesoría requerida. Se parte, de la búsqueda de conocimiento, su existencia, ubicación y forma de contactar al portador del conocimiento. Para lo anterior se logró una aplicación capaz de filtrar un tipo de conocimiento y el mapeo sobre la plataforma de la API de Google Maps.

**Palabras clave:** Conocimiento, mapa de conocimiento, Google Maps, emprendedores, asesoría especializada.

## 1. Introducción

Las IE (Incubadoras de Empresas) y CDE tiene la misión de apoyar a emprendedores y microempresarios que desean que sus empresas tengan crecimiento o mejora de utilidades, procesos, calidad u otros propósitos.

Es preciso contar con expertos que una vez registrados, forman parte de una red de AE (Asesores Especializados). De cada AE se conoce su nombre o razón social, alguna dirección o solo ubicación, así como forma de contacto: correo electrónico, página Web, teléfono celular o fijo u otro medio que establezca.

Una vez registrada la red de AE; se requiere un IC (Inventario de Conocimientos). El IC contiene los conocimientos de los AE que puede ser actualizado en medida que se adquieren nuevos conocimientos.

Para la administración del conocimiento se define un catálogo de conocimientos, que a través de una base de datos, es posible su almacenamiento y consulta.

Por otro lado la tecnología nos permite otras nuevas formas de consultar la información y la comunicación entre personas e instituciones. Con un simple clic, se puede navegar por el mundo sin siquiera moverse de la oficina, accediendo a las bases de datos de algunos servidores que existen en el planeta.

Ya existen sitios capaces de buscar una ubicación, una ruta, un taxi, un restaurante, un café, museos. ¿Por qué no, sitios, para buscar conocimiento?

Este sistema permite conocer: ¿qué conocimiento existe?, ¿quién lo posee?, ¿dónde están los portadores? y ¿cómo acceder al conocimiento?

La API (Interfaz de Programación de Aplicaciones) de Google Maps[1] puede incorporarse a las aplicaciones ofreciendo mostrar vistas de calles o satélite. Además es posible: insertar ubicaciones e información personalizada. Así se muestra en los resultados.

## 2. El conocimiento

La historia de la GC (Gestión del Conocimiento) ha venido evolucionando hasta el punto que aún no se encuentra una definición clara y completa a este término; a lo que muchos autores han tratado de formular su propia definición, como Nonaka y Takeuchi; planteando que "La Gestión del Conocimiento implica llevar los conocimientos correctos a las personas que lo necesitan "just in time" con el objeto que puedan resolver el problema que deseen con prontitud y eficacia" [7].

Otra definición sería la de "Tratar de extraer lo mejor de las personas de la organización utilizando sistemas que permiten que la información disponible se convierta en conocimiento" [1].

Se puede afirmar que al gestionar el conocimiento facilita:

- La localización de fuentes de conocimiento.
- La reutilización de experiencias.

[1] Google Maps en una aplicación de la Empresa Google Inc. Sistema de mapas digitales lanzado en febrero de 2005. Disponible en www.maps.google.com.

- La mejora de los procesos de negocios.
- La reutilización de artefactos del proceso.

Existiendo experiencia en organizaciones que demuestran que se puede gestionar el conocimiento y se empiezan a adoptar arquitecturas Fig. 1 para dicha gestión, donde éstas, se puede considerar como el nexo que une a las actividades de producción diarias con las actividades de mejora y los objetos de negocio.

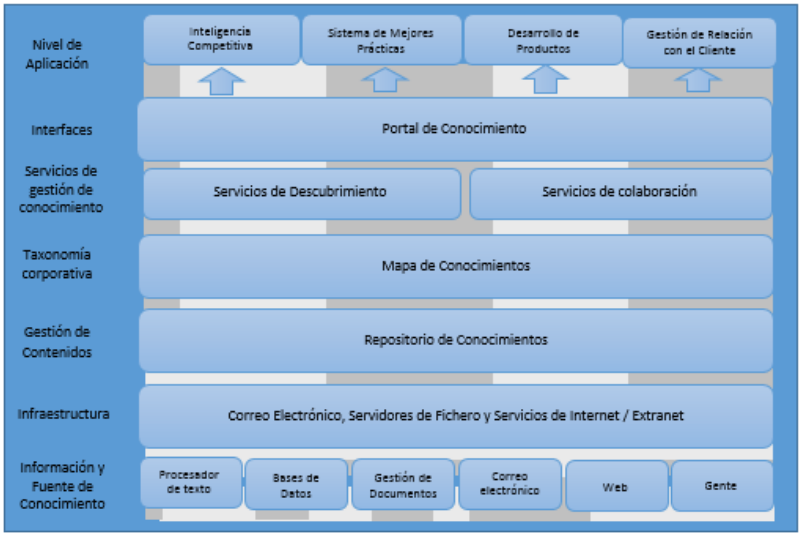



**Fig. 1.** Arquitectura de gestión de conocimiento (Piattini, 2007).

### 2.1. Asesores de negocios

Un asesor de negocios es un consultor que ofrece a los propietarios de empresas conocimiento que les ayude a manejar sus negocios más eficazmente. Estos pueden ser consultados para ayudar a simplificar o mejorar un negocio, o pueden jugar un papel más activo, que ofrece sesiones de asesoramiento recurrente a sus clientes.

Estos asesores pueden ayudar a diagnosticar dónde y por qué una empresa tiene mal desempeño, para revertir una tendencia negativa, además de ayudar a las personas con el desarrollo de un nuevo negocio o una nueva diversificación en un negocio existente.

### 2.2. Centro de Desarrollo Empresarial

Es un proceso dinámico para la estimulación de nuevos y pequeños proyectos empresariales. Por ellos las incubadoras otorgan a las empresas y proyectos jóvenes, dentro del campo competitivo, apoyo durante sus etapas de desarrollo, ya que en estas son más vulnerables.

### 2.3. Historia de la Incubadora de empresas

La existencia de incubadoras de empresas nació hace casi 50 años en los Estados Unidos.

La gran mayoría de las empresas que inician, no llegan a desarrollarse, y es aquí donde apoyan las incubadoras, ya que ayuda a reducir la mortalidad de dichas empresas.

La Asociación Mexicana de Centros para el Desarrollo de la Pequeña Empresa surge ante la necesidad de contar con un organismo que apoye a los centros de desarrollo de las PyMEs que han surgido en nuestro país, en sus esfuerzos para el desarrollo de la pequeña empresa, representando el interés colectivo de dichos centros ante entidades gubernamentales y privadas, nacionales o extranjeras [2].

### 2.4. Atención a emprendedores

A través de dos departamentos, el de incubación de empresas y el de capacitación empresarial, el CDE, brinda a los nuevos emprendedores servicios indispensables para ellos, como son: tutoría y asesoría, en donde se desarrollan las propuestas de mejora, comercial, técnica, financiera y administrativa para empresas en operación.

En el desarrollo de negocios, se hace un acompañamiento y soporte del emprendedor a través de las etapas de diagnóstico, gestación e incubación de un proyecto.

El albergue, donde en los primeros meses de operación, el emprendedor/empresario podrá contar con el uso de un espacio de oficina con sus servicios administrativos necesarios [3].

### 2.5. Google Maps

Es una aplicación de Google Inc. Es un servidor de aplicaciones de mapas en la Web que ofrece imágenes de mapas desplazables, así como fotos satelitales del mundo, e incluso la ruta entre diferentes ubicaciones o imágenes a pie (utilizando la API de Google Street View)[2].

Es idéntico a Google Earth[3], una aplicación que ofrece vistas del globo terráqueo, sea de día o de noche, pero que no es fácil de integrar a páginas Web. Está disponible para Android[4] y Java ME[5].

Google Maps ofrece la capacidad de hacer acercamientos o alejamientos para mostrar el mapa. El usuario puede controlar el mapa con el mouse o las teclas de

[2] Aplicación de Google Inc. Que permite visualizar calles de las principales ciudades, como si lo hiciera a pie.

[3] Aplicación de Google Inc. Permite ver la tierra en forma de mapa de carreteras e imágenes satelitales.

[4] Sistema Operativo creado por Google Inc. Actualmente integrado en la mayoría de tabletas y Smartphones.

[5] Plataforma para el lenguaje de Java bajo la dirección de la Empresa Sun Microsystems. Disponible en: http://www.oracle.com/us/sun/index.htm.

dirección para moverse a la ubicación que se desee. Los usuarios pueden ingresar una dirección, una intersección o un área en general para buscar en el mapa. Los resultados de la búsqueda pueden ser restringidos a una zona, gracias a Google Local. Por ejemplo, si alguien quiere consultar por "Waffles in Ottawa", para encontrar restaurantes que sirven este alimento cerca de la ciudad. Las búsquedas pueden encontrar una amplia gama de restaurantes, hoteles, teatros y negocios generales.

Las coordenadas de Google Maps se encuentran en el sistema WGS84[6], el cual mostrará la latitud y la longitud, positiva para Norte y Este, negativa para Sur y Oeste. Hay varias formas de obtenerlas, una vez que se ha localizado el lugar que nos interesa.

### 2.6. Actividad de compartir el conocimiento en México.

Las lecciones aprendidas se obtienen al culminar una tarea, actividad o un proyecto, las cuales se deben documentar para una consulta posterior por todos los integrantes de la organización [12, 13]. En México el modelo MoProSoft (Modelo de procesos para el desarrollo de software) recomienda almacenar la documentación de los procesos en una base de conocimiento de acceso libre para los integrantes de la organización [13].

Compartir el conocimiento en las organizaciones es una ventaja competitiva que permite agregar valor a los productos y servicios y a la organización misma. Sin embargo para compartir el conocimiento es necesario un ambiente, herramientas y procesos definidos para dicha tarea [4].

El conocimiento se genera y se acumula a partir de lo que sé, de lo que aprendo de otros y lo que aprendo haciendo. Una vez que se posee se debe compartir con los demás logrando un conocimiento colectivo u organizacional [15].

Compartir el conocimiento de persona a persona requiere de un ambiente adecuado, pero en organizaciones distribuidas o grandes es necesarias herramientas donde es necesario trasformar el conocimiento tácito a explícito y de explícito a tácito[7] [7].

### 2.7. Transferencia del conocimiento en México

En México existe transferencia de conocimiento; pero, ¿cómo se transfiere el conocimiento? podría ser cara a cara, con cursos de capacitación o desde internet; sin embargo ¿cómo lo hacen? Podría ser informalmente, porque no existen procesos y políticas que lo incluyan como actividad en la creación de nuevas empresas.

---

[6] Es un sistema de coordenadas geográficas mundial que permite localizar cualquier punto de la Tierra (sin necesitar otro de referencia) por medio de tres unidades dadas. WGS84 son las siglas en inglés de World Geodetic System 84 (que significa Sistema Geodésico Mundial 1984).

[7] Se considera la obra de Nonaka y Takeuchi, a pesar de su antigüedad, por ser la obra clásica que fundamenta el conocimiento tácito y explícito así como el proceso de la transferencia del conocimiento; que a pesar del año de publicación sigue siendo vigente.

Otra forma es a través de textos, repositorios de documentos, la intranet, groupware y SGC (Sistema de Gestión del conocimiento). Y una forma más consiste en aprender viendo, se dispone de un experto y un aprendiz que al observar cómo se desarrollan las actividades aprende [5, 6].

La socialización es el medio más eficaz para la transferencia del conocimiento tácito-tácito, por eso en Japón se acostumbra que las personas socialicen, de hecho prefieren la transferencia del conocimiento tácito [11].

### 2.8. Almacenamiento del conocimiento

Es importante obtener un inventario de activos de conocimiento, que corresponden al conjunto de conocimientos, que posee una persona o plasmado en un medio y que guarda de forma cohesiva la solución de un problema específico, dentro de un dominio del conocimiento.

En la empresa se guardan todos los activos de conocimiento explícito en una base de conocimiento; dichos activos de conocimiento explícito aumentarán en medida que se introduzcan a la base de conocimiento.

### 2.9. Inventario del conocimiento

Para el almacenamiento del conocimiento es necesario un repositorio de conocimiento denominado base de conocimiento donde se almacenan el conocimiento que poseen los AE. El conocimiento almacenado forma parte de un inventario de conocimiento. La unidad son componentes de conocimientos llamados activos de conocimiento [5, 10].

## 3. Método utilizado

### 3.1. Identificación de los Asesores Especializados

Los AE son docentes o personal de apoyo que se encuentran dentro o fuera del Centro Universitario UAEM Texcoco, los cuales sirven de soporte para los emprendedores que se acerquen al CDE.

Los AE son registrados generando un catálogo con su información de contacto personal, como su nombre, forma de contacto, dirección de correo electrónico, teléfono de contacto, y algún otro dato que este quiera brindar.

### 3.2. Categorías de conocimientos en el CDE

Debido a que los emprendedores que se acerquen al CDE, vendrán con la idea de un negocio, o la mejora del mismo; se debe tener una Base de Datos en la cual se cuente con un amplio repertorio de los conocimientos que tiene a su alcance el CDE.

Para que el usuario, que será el responsable de atender a los emprendedores en el CDE, no se le complique la toma de decisión al momento de elegir entre los temas de

asesoramiento que tiene dicho centro de desarrollo, se ha hecho una categorización de conocimientos. En vez de poner la palabra "Ingeniería", como una categoría la cual abarca subtemas en los cuales se tiene en muchas ocasiones más especialistas; se optó por poner específicamente categorías como "desarrollador web", "administrador de una base de datos", "mercadotecnia", "asesor de impuestos", etc.

### 3.3. Alcance geográfico del CDE

Los CDE se encuentran, en los Centros Universitarios. Los emprendedores recurren a los CDE, porque tienen la seguridad de que cuentan con el personal necesario y experimentado para que los logre a cumplir sus propósitos.

El alcance principal, en donde se lanza este sistema, es únicamente en la zona de Texcoco y municipios aledaños. Lo anterior porque se considera la movilidad de los AE y de los emprendedores.

### 3.4. Diseño de la arquitectura de la aplicación

Se optó por adaptar una arquitectura en capas basada en la arquitectura de Piattini [14]. Se incluyen 5 capas desde la obtención de la información hasta la aplicación que interactúa con el usuario, Fig. 2.

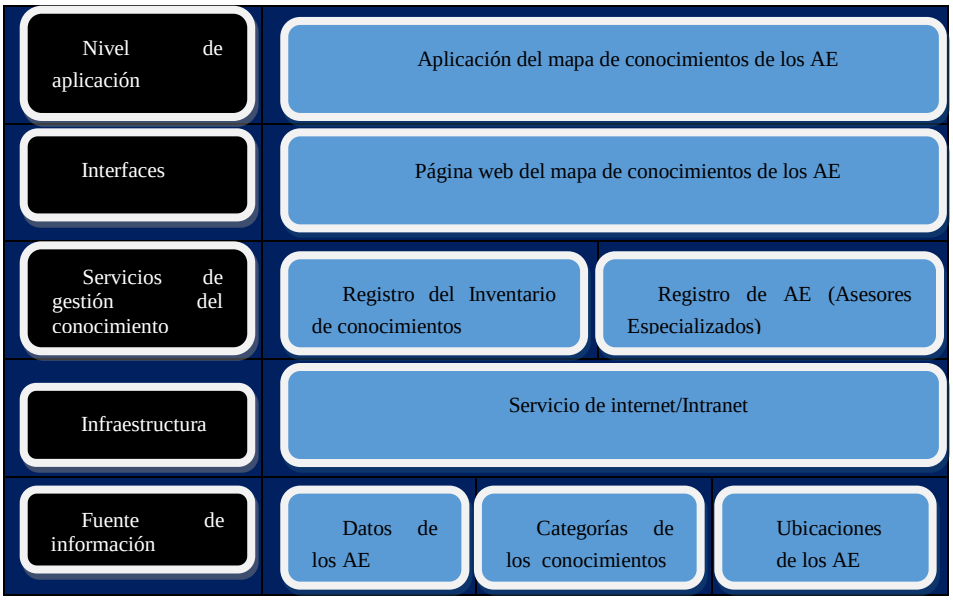



**Fig. 2.** Arquitectura de la aplicación del Mapa de Conocimientos (Adaptación de [14]).

Cabe precisar que no se trata de un SBC (Sistema Basado en el Conocimiento), sino más bien una plataforma que permite generar un mapa de conocimiento que apoya en la búsqueda del conocimiento y a los portadores.

### 3.5. Diseño de la Base de Datos

En el diseño de la BD (Base de Datos), se usan tres tablas que son creadas desde la línea de comandos de MySQL Console[8] con el apoyo de la suite de WampServer[9], ya que dicha suite es gratuita y compatible con Windows, el sistema operativo que se está utilizando para este proyecto. La tabla 1se muestra a continuación:

**Tabla 1.** Tabla del catálogo de Asesores especializados.

| asesores_esp | | | | | |
|---|---|---|---|---|---|
| **Id_AE** | **Nombre_AE** | **Direccion_AE** | **Telefono** | **Correo** | **Id_ubicacion** |
| 1 | Ing. Antonio Diaz | Plata s/n, Arenal 2, Chicoloapan. | 5531494882 | ittonzs@gmail.com | 1 |
| 2 | Lic. Oscar Muro | CU UAEM Texcoco | 5531494882 | ittonzs@gmail.com | 2 |
| 3 | Ing. Francisco Nava | CU UAEM Texcoco | 5531494882 | ittonzs@gmail.com | 3 |

La tabla de Asesores Especializados, donde se incluirá la información de contacto de cada asesor, como su nombre, la dirección que el asesor quiera mostrar como contacto si es que tanto como el asesor y emprendedor requieran citas fueras del horario de operación de los CDE, de igual forma el teléfono y dirección de correo electrónico, para poner en contacto al Emprendedor y al AE.

La tabla de Asesorías, permite alojar nombres de posibles asesorías la y clave de identificación de cada una. Se muestran los nombres de los AE en el sistema, y que el usuario podrá elegir para conocer quien posee el conocimiento de dicha asesoría. Es posible examinar la tabla 2 en seguida.

**Tabla 2.** Registro de asesorías.

| Registro de Asesorías | | |
|---|---|---|
| **Id_ase** | **Id_AE** | **Nombre_ase** |
| 1 | 1 | Base de Datos |
| 2 | 2 | Desarrollo web |
| 2 | 3 | Redes |

En esta tabla se hacen las posibles combinaciones entre las Asesorías y Asesores, considerando que un Asesor puede poseer el conocimiento en dos o más temas a consultar.

La tabla 3 de Ubicaciones, es donde se establecen los lugares donde están los AE (podría ser más de uno), donde estén almacenadas las direcciones o ubicaciones.

[8] Interfaz no gráfica de MySQL. MySQL es un sistema de gestión de bases de datos relacional lanzada por Sun Microsystems y actualmente por Oracle Inc.

[9] Paquete de instalación que contiene un servidor (Apache), el gestor de base de datos MySQL y PHP como lenguaje de programación; para el Sistema Operativo Windows.

**Tabla 3.** Dirección o ubicación de los AE.

| Direcciones de los AE | | |
|---|---|---|
| **Id_ubicacion** | **latitud** | **longitud** |
| 1 | 19.417154 | 98.897545 |
| 2 | 19.434422 | 98.916420 |
| 3 | 19.434437 | 98.917503 |

Estas ubicaciones están almacenadas con su latitud y longitud, puesto que la API de Google Maps usa las coordenadas para poder ubicar a nuestros AE en el mapa.

Para la obtención de dichas coordenadas, se requirió hacer la conversión de dirección física a coordenadas, lo que Google le llama Geocode Simple.

Pero gracias a la política de OpenSourceCode que Google maneja, se pueden encontrar aplicaciones que hacen esto, en donde se muestran los marcadores y al dar clic en alguno de ellos muestra las coordenadas necesarias para el sistema.

### 3.6. Diseño de la interfaz del sistema

Para el diseño de la interfaz se optó por la elaboración de una página web, para que en un futuro sea parte de la página principal de la institución.

Dicha interfaz es muy amigable con el usuario, solamente se tiene que elegir entre las distintas asesorías que cuenta el CDE, para saber dónde y quién posee dicho conocimiento que le será de ayuda al emprendedor. La interfaz se puede apreciar en las vistas en las Fig. 2, 3 y 4 de la sección de resultados.

Para hacer uso del mapa de Google, ofrece su API la cual es una línea de código que se incrusta dentro del <head> (cabecera) de nuestro sitio. La versión más reciente de la API de Google Maps es la versión 3, la cual y está disponible en la página oficial de Google.

### 3.7. Programación requerida

Para la construcción del sitio web (únicamente la interfaz), se hizo usando HTML5[10], la versión de HTML más reciente, ayuda a la compatibilidad con los navegadores que existen actualmente.

De igual manera, como otros sitios web, se hacen utilizando JavaScript[11], un lenguaje de programación para páginas y sitios web. Las sentencias que se usan son parecidas a programar en java, diferenciando un poco aquí pues se usan unas "etiquetas" para identificar el comienzo y final de una "instrucción". Por ejemplo para incluir la API de Google Maps se usa la etiqueta <script> en la que se declara la API, la cual manda a llamar a los servidores de Google que permita hacer uso del

[10] Es la versión 5 del lenguaje básico de la world wide web. Bajo la regulación del consorcio W3C.

[11] Lenguaje de programación interpretado orientado a objetos. Se incrusta en el código HTML para que los sitios sean dinámicos.

mapa, que cabe mencionar, que Google nos permite modificar y hacer sus mapas hasta cierto punto a nuestras necesidades, pues cuenta con varias opciones para adecuarse a nuestros proyectos; como pueden ser los markers, opciones del zoom, la ubicación para mostrar en primera instancia el mapa, por ejemplo nuestro sitio posicionarlo en un área donde quede centrada la zona de Texcoco, Estado de México. En el código siguiente se establece el mapa a mostrar.

```
function initialize () {
   var mapOptions = {
      zoom: 8,
      center: new google.maps.latLng(-34.397, 150.644)
};
Var map = new google.maps.Map(document.getElementById('map-canvas'),
mapOptions);
}
Google.maps.event.addDomListener(window,  'load',  initialize);
```

Dentro de la función principal se declara una variable, que es "mapOptions", la cual tiene dos opciones:

Zoom: nos permite mostrar el nivel de acercamiento visto desde el cielo para ver cierta zona del mapa.

Center: con esta opción nosotros le indicamos al mapa en que zona del mapa queremos posicionarnos a través de unas coordenadas de latitud y longitud.

Como el sistema que se desarrolló hace una consulta a través del combolist este debe de enviar dichas consultas a la base de datos y así muestra el resultado, pues lo que se hace es usar el método "post", que nos permite enviar la selección de nuestra consulta y arrojar un resultado sin que el usuario lo pueda ver, como usar el método "get", con el que se muestran todos los resultados al usuario y que para el sitio no es el caso.

Para usar el método "post" se declara en otro archivo, donde también se tiene que incluir el anterior archivo, al que se le ha llamado "puntosDados.php", y se incluye en el archivo de la misma manera que se incluyó el archivo de conexión de la BD. Se declaró a "post" como una variable global " $_POST ", para que la podamos usar en cualquier momento y con cualquier función sin tener que declararla para cada una de ellas.

Para ello en el archivo que se llamará "index.html", el cual es el principal para que pueda correr en un navegador, puesto que los navegadores buscan este nombre para poder mostrar algo en pantalla. Este archivo es donde se había declarado el "<!DOCTYPE hmtl>"  y donde estaba el "<script>" de la API de Google Maps [8].

### 3.8. GEOCODE SIMPLE (Codificación Geográfica)

Es el proceso de transformar direcciones (como "1600 Amphitheatre Parkway, Mountain View, CA") en coordenadas geográficas (como 37.423021 de latitud y -122.083739 de longitud), que se pueden utilizar para colocar marcadores o situar el mapa. El API de codificación geográfica de Google proporciona una forma directa de

acceder a un geocoder mediante solicitudes HTTP. Además, el servicio permite realizar la operación contraria (convertir coordenadas en direcciones); este proceso se conoce con el nombre de "codificación geográfica inversa".

## 4. Resultados

El resultado obtenido de este trabajo consiste en una aplicación web capaz de funcionar en dispositivos con un navegador y conexión a internet. Un administrador del sistema cargó un catálogo de categorías de conocimiento de acuerdo al Inventario de Conocimientos que tiene el CDE a través de los AE. También fue necesario registrar los AE con sus datos de ubicación y formas de contacto. Para cada AE se debió determinar su ubicación geográfica para poder mapear su existencia. La aplicación filtra los AE de acuerdo a la categoría elegida, y solamente muestra los AE existentes. Cuando se elige una categoría y no se tienen ningún experto, no se muestra ninguna marca. La página principal se puede observar en la figura 3.


**Fig. 3.** Vista de la aplicación para la elección de categoría de conocimiento.

Una vez que se elige una categoría de conocimiento, la aplicación filtra los AE que poseen ese tipo de conocimiento y se muestran marcas en el mapa.

El usuario puede dar un clic en la marca para desplegar los datos del AE, como se muestra en la Fig. 5. El sistema no emite automáticamente el aviso a los AE, pero es posible, agregar la función; por ahora toma los datos de contacto y podrá de forma tradicional contactar y acordar detalles de la asesoría requerida.

La aplicación se ha implementado en un servidor del CDE del Centro Universitario UAEM Texcoco, se están registrando el inventario de conocimiento y los AE, posteriormente se valorará su impacto.


**Fig. 4**. Marcas de AE que poseen el conocimiento elegido.

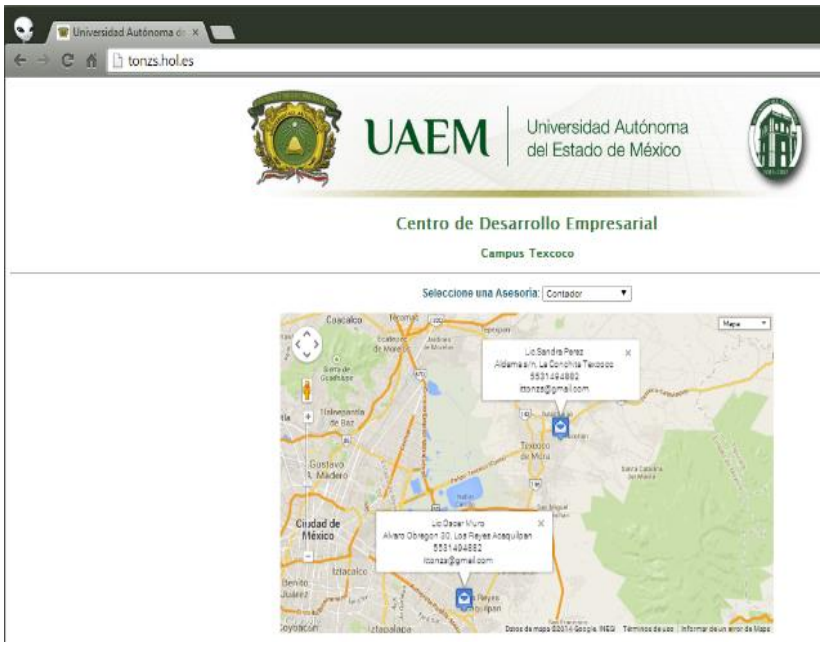



**Fig. 5.** Mapa con los datos de los AE desplegados.

## 5. Conclusiones y trabajos futuros

Después de haber llevado a cabo pruebas con diferentes navegadores, se observó que no existe variación alguna en cuanto al comportamiento de los elementos del sitio web, así como tampoco en los resultados que arrojaron las consultas hechas, y esto se debe a que desde que se empezó a programar y diseñar el sitio, se contempló utilizar las sentencias que indicaban ser las versiones más recientes y estables de HTML como de la API de Google Maps.

Desde una primera instancia se tenía como objetivo que el sistema funcionara arrojando los resultados de las consultas en un mapa. Al final la aplicación cumple el cometido y se logra el objetivo principal.

La aplicación resuelve el problema del CDE porque permite que los usuarios accedan a la aplicación elijan una categoría de conocimiento existente y obtengan un mapa que muestra la existencia de conocimiento; así como la ubicación y a los portadores.

Durante la implementación fue fácil y amigable; además de encontrar la gran gama de posibilidades para otras soluciones para mapear no solo conocimiento sino otros aspectos; como; seguridad, riesgos, mercados, etc.

Se hicieron pruebas por parte de usuarios, personal del CDE y fuera de la región de Texcoco arrojando los resultados esperados.

Se planean dos trabajos futuros: El primero consiste en lograr que el sistema permita la programación, registro, monitoreo y estadísticas de las asesorías a los emprendedores y empresarios por parte de los AE. La segunda consiste en agregar información más específica del conocimiento, AE y ubicación con el fin de agregar un algoritmo genético para determinar cuál es el AE que más conviene contactar.

## Referencias

1. Carrión, J.: Gestión del conocimiento. Recuperado en Mayo, 2002, vol. 4, p. 2007.
2. Centro Universitario UAEM Texcoco. Centro de Desarrollo Empresarial. http://incemtex.mx/
3. Centro Universitario UAEM Texcoco. Centro de Desarrollo Empresarial.
4. Collison, C., Parcell, G.: La gestión del conocimiento, Lecciones prácticas de una empresa líder. Buenos Aires: Paidós (2003)
5. Davenport, T., Prusak, L.: Conocimiento en acción, Como las organizaciones manejan lo que saben. Buenos Aires: Prentice Hall (2001)
6. Davenport, T., et al.: Building Successful Knowledge Management Projects. [En línea]. 16-04-13, Disponible en Internet: http://www.providersedge.com/docs/km_articles/Building_Successful_KM_Projects.pdf
7. De la Espriella Fortoul, L.M, Pineda Pinzon, D.C.: Gestión del conocimiento. Soluciones efectivas S.A. Articulo. [en línea][23-04-13]. Disponible en internet: http://hdl.handle.net/10818/2012
8. Google Maps. Obtenido el 23 de abril de 2013 de https://developers.google.com/maps/web/
9. http://incemtex.mx/

10. Martínez, Martínez, A., & Corrales, M. (2010). *Administración de conocimiento y desarrollo basado en conocimiento, redes e innovación.* México D. F.: CENGAGE Learning.
11. Nonaka, S., & Takeuchi, N. (1999). *La organización creadora del conocimiento.* México: Oxford.
12. Normalización y Certificación Electrónica A. C. (2007). *Guía práctica de implantación de los requisitos de la NMX-I-059-NYCE-2005 (MoPRoSoft).* México D. F.: Normalización y Certificación Electrónica, A. C.
13. NYCE A. C:. (2007). *Guía práctica de implantación de los requisitos de la NMX-I-059-NYCE-2005 (MoProSoft)* (Primera ed.). México D. F.: Normalización y Certificación A. C.
14. Piattini, M.; García, F. y Caballero, I. *Calidad de Sistemas Informáticos.* México, DF: Alfa Omega Grupo Editor, S.A de C. V., 2007. 416p. ISBN: 978-970-15-1267-8.

Ruiz, S., Ledeneva, Y., & Morales, R. (2012). Base de conocimiento de los procesos de desarrollo de software a través de un modelo de un sistema de gestión del conocimiento. *Research in Computing Science, Avances en inteligencia artificial, 55*, 113-123.

# Evocación de hábitos en personajes virtuales mediante Mapas Cognitivos Difusos y técnicas de videojuegos

J.Carlos Conde-Ramírez, Abraham Sánchez-López

Facultad de Ciencias de la Computación,
BeneméritaUniversidad Autónoma de Puebla, México

{juanc.conde, asanchez}@cs.buap.mx

**Resumen.** Los últimos avances tecnológicos en realismo de personajes virtuales y sus ambientes han vuelto necesaria la adaptación autónoma de dichos personajes a situaciones particulares en tiempo real. Esta investigación propone un proceso simple para crear agentes virtuales que aparenten mayor inteligencia a través de sus comportamientos. Se verifica como el modelado cognitivo de personajes permite alcanzar respuestas apropiadas a estímulos externos y deseos. Como prueba se implementó un mundo submarino donde los comportamientos reportados están basados en una arquitectura cognitiva propia (MoCAMG) combinada con técnicas comunes de videojuegos y Mapas Cognitivos Difusos no aumentados. La definición e identificación de patrones de comportamiento, reportados al final, justifican la importancia de la arquitectura cognitiva utilizada. Ésto sirve como base para investigaciones futuras en fenómenos detallados relacionados con la memoria.

**Palabras clave:** Modelado cognitivo, comportamiento, adaptación autónoma, personajes virtuales, agentes autónomos, videojuegos.

## 1. Introducción

Para que las aplicaciones gráficas modernas puedan generar procesos mecánicos realistas en personajes, basados en elementos cognitivos, existen estudios en seres vivos e información (psicológica, anatómica, etológica, etc.) de sobra que puede ser de gran utilidad. Como se menciona en [19], un modelo cognitivo adecuado puede proporcionar los datos necesarios para generar simulaciones validadas por modelos teóricos. Hoy en día aplicaciones de este estilo, como auto-animaciones y videojuegos, necesitan implementar modelos cognitivos que le permita a un personaje reflejar un comportamiento congruente con su entorno y situación particular.

En específico, este trabajo muestra cómo a partir de un conjunto de requerimientos obtenidos en tiempo real (percepción), es posible *pre-diseñar* agentes o personajes autónomos sin tener que especificar cada uno de los detalles de su comportamiento (control de alto nivel). Como evidencia se reproduce el comportamiento peces inmersos en un ambiente submarino. Se enfatiza la sencillez

de utilizar Mapas Cognitivos Difusos (FCM's) independientes o no aumentados, ni anidados, para obtener comportamientos congruentes con patrones de comportamiento o hábitos. Estos patrones se obtienen combinando técnicas de sensado usadas en videojuegos y la definición *off-line* de un di-grafo tratado de forma difusa. Se destaca el hecho de que el uso de FCM's no excluye a los personajes de enfrentarse a situaciones imprevistas, pero integra acciones primarias o primitivas en acciones motivadas por factores dinámicos.

A continuación se describe cómo está estructurado este documento. La *Sección 2* describe cuáles son los componentes de comportamiento en los seres vivos más relevantes para este trabajo. Algunos de los enfoques más utilizados para representación del conocimiento en este tipo de trabajos se presentan en la *Sección 3*. La *Sección 4* resume algunos de los trabajos relacionados con el área de animación y videojuegos. Por su parte, la Sección 5 contiene las especificaciones de la arquitectura utilizada y una metodología para el modelado cognitivo de personajes virtuales. En la Sección 6, se muestra la implementación de un sistema que simula un comportamiento básico (supervivencia) en peces, con el objetivo de probar la utilidad de la arquitectura y la metodología propuesta, utilizando FCM's simples. Como resultado, la Sección 7 especifica todos los posibles patrones de comportamiento obtenidos de la ejecución del sistema implementado. Para concluir, la *Sección 8* destaca la importancia de este trabajo y la dirección que tomará el trabajo futuro.

## 2. Elementos comportamentales

Las prioridades u objetivos primordiales para todos los seres vivos, tal como sobrevivir o reproducirse, son objetivos que pueden descomponerse en otros más inmediatos.

Además, es sabido que los seres vivos enfocan su atención de dos maneras: utilizando órganos de percepción especializados y enfocando su atención de forma cognitiva. Son características importantes que mejoran el procesamiento de la información sensorial. Por lo tanto, cualquier sistema que intente simular tales características debe considerar los siguientes factores [20]:

- El *ambiente*, los *estímulos externos* y los *deseos*.
- El proceso requerido para seleccionar una acción.
- La animación comportamental derivada.

Aquí el problema crítico es seleccionar una de todas las posibles acciones. Ésto conduce a un problema de diseño que puede ser resuelto de la siguiente manera:

1. Identificando los principios por los cuales un ser vivo selecciona sus acciones.
2. Representando y organizando el conocimiento de manera eficiente.

La selección de acciones basada en el razonamiento involucra el uso de técnicas de Inteligencia Artificial llamadas *planificación de movimiento a nivel de*

*tarea*. Por su parte, las secuencia de acciones que le permiten a un personaje artificial ser autónomo y "sobrevivir" en ambientes dinámicos corresponde a un comportamiento básico conocido como comportamiento reactivo o adaptativo [16,17].

## 3. Representación del conocimiento

La representación del conocimiento y el razonamiento es un área de la inteligencia artificial cuyo objetivo fundamental es representar el conocimiento de una manera que facilite la inferencia (sacar conclusiones) a partir de dicho conocimiento. Junto a una teoría de interpretación, dan significado a las frases en la lógica.

En este caso, un personaje realiza una acción de "percepción" (sensado) para actualizar su información y entonces hacer posible la replaneación de sus acciones. Por eso es importante que la arquitectura utilizada en la implementación de modelos cognitivos esté armada con un enfoque viable para representar la incertidumbre.

### 3.1. Arboles de decisión

Según la literatura, usando *axiomas de precondición* es posible definir sentencias que especifiquen cuál es el estado del mundo antes de realizar alguna acción. Los efectos de una acción están dados por los *axiomas de efecto*. *Acciones*, *axiomas de efecto* y *axiomas de precondición* pueden ser expresados en forma de árbol, donde:

- Los nodos representan las situaciones.
- Los axiomas de efecto describen las características de cada situación (la raíz del árbol corresponde a la situación inicial $s_0$).
- Los axiomas de precondición permiten saber cuáles son las secuencias de acciones permitidas.

Además cada ruta sobre el árbol puede representar una posible secuencia de acciones. Definiendo algunas situaciones (nodos) como "objetivos" es posible utilizar programación basada en lógica convencional y realizar búsquedas para hallar una secuencia de acciones que conduzcan al personaje hacia su objetivo [9].

### 3.2. Mapas Cognitivos Difusos

Un ambiente virtual cambia con el tiempo conforme el actor se mueve, por lo que existe la posibilidad de que el actor modifique su entorno o que su entorno lo cambie a él. De acuerdo a Julie A. Dickerson y Bart Kosko en [6], los cambios en un mundo virtual son causales y a su vez esta causalidad retroalimenta otros cambios.

Modelos matemáticos tales como las ecuaciones de Navier-Stokes, las ecuaciones de cinemática inversa o las ecuaciones diferenciales acopladas, son modelos

que pueden ayudar a resolver cambios en un mundo virtual. La desventaja es que son difíciles de encontrar, difíciles de resolver y difíciles de ejecutar en tiempo real. Usualmente los árboles de decisión "facilitan" la elección de acciones en agentes artificiales a través de búsquedas que retroalimentan el conocimiento del personaje. Sin embargo, cada inferencia utiliza sólo una pequeña parte del conocimiento almacenado, sin mencionar que el diseñador debe anticipar qué acción podría ser seleccionada bajo todas las posibles condiciones.

En contraste, los Mapas Cognitivos Difusos (FCM's) [14] son redes causales representados por un grafo dirigido difuso, donde los nodos representan conceptos, acciones o deseos, y los ejes causales establecen las reglas difusas entre dichos conceptos. Además cada entrada dispara todas las reglas, en cierto grado, para modelar una "causalidad circular" en mundos virtuales realistas.

Por si mismos, los FCM's actúan como un sistema dinámico no lineal (tal como en una Red Neuronal) donde las entradas son mapeadas con estados de equilibrio como salida. Por lo tanto, en un FCM simple la ruta concluye en un punto fijo o en un ciclo limitado.

## 4. Trabajo relacionado

Los problemas relacionados con la creación de personajes virtuales con mejores esqueletos y cuerpos deformables más realistas son considerados en la mayoría de trabajos recientes en el campo de gráficos y animación por computadora. No obstante, la problemática de brindar un nivel de comportamiento creíble recae en el reto de generar movimientos libres y suaves dentro de ambientes utilizando capacidades de "percepción" realistas. Hasta ahora, un área de investigación poco abordada en comparación con otras.

Uno de los trabajos importantes en el área es el de N. Magnenant-Thalmann y D. Thalman, quienes presentan los resultados de crear humanos virtuales realistas. La interacción y comportamientos grupales son parámetros importantes, por lo que se concluye que la *percepción realista* y el *estado interno* del "actor" debe verse reflejado en su comportamiento [15].

Por su parte, estudios en *emociones artificiales* demuestran que la implementación de *modelos cognitivos* en aplicaciones gráficas, como videojuegos, generan una interacción con los usuarios más notable. En [13], los autores diseñan y prueban modelos de emoción artificial en un personaje. El trabajo consiste en una herramienta de pre-visualización que puede ser utilizada para planear niveles en videojuegos. En el mismo sentido, los autores de [21] proponen un sistema basado en modelos de comportamiento que revelan la interrelación jugador-ambiente.

Un área de oportunidad puede ser la mejora en motores de juegos. Un enfoque como éste es propuesto por E. Hudlicka en [12], en donde se establecen los requerimientos a satisfacer por parte de los trabajos y aplicaciones de este tipo. El objetivo es mejorar el realismo social y afectivo de los personajes identificando "emociones" en los personajes.

Finalmente, un análisis crítico realizado por T. Giang en [11] revela los enfoques de alto nivel empeñados en identificar las áreas clave de interés para in-

vestigaciones futuras relacionadas con las *Técnicas de Animación de Personajes en Tiempo-Real.*

## 5. Propuesta teórica

La arquitectura MoCAMG (Movis's Cognitive Architecture for Modeling Game-characters) mostrada en la Figura 1, combina algunos componentes de dos arquitecturas bastante conocidas; ACT-R and Soar [1,18].

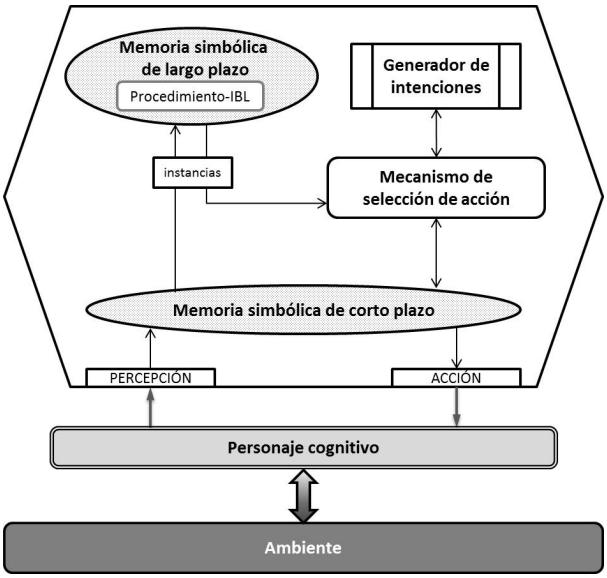



**Fig. 1.** Vista de alto nivel de la arquitectura MoCAMG.

La arquitectura propuesta posee dos tipos de memoria: declarativa (datos) y procedural (reglas). La memoria declarativa está asociada con objetivos y es removible, por lo que es considerada como una *memoria de corto plazo*. Por su parte, en una memoria procedural las "reglas" pueden ser actualizadas o agregadas, pero no removidas en tiempo de ejecución tal como en una *memoria de largo plazo*.

En este trabajo nos enfocaremos en el módulo nombrado Generador de Intenciones, que es el componente clave para obtener patrones de comportamientos. Funciona como arbitrador de comportamientos cuya activación depende del estado interno y externo de la entidad cognitiva. En este caso, el Mecanismo de Selección de Acciones combina comportamientos primitivos en comportamientos motivados.

Los detalles relativos a los demás componentes se describen en [4]. La única observación importante de este trabajo es que se actualizó el tipo de conexión

entre el Generador de Intenciones y el Mecanismo de Selección de Acciones en el diagrama de la arquitectura, ya que en la práctica se comprobó que existe un flujo de trabajo bidireccional entre estos dos componentes (Figura 1).

### 5.1. Metodología para el modelado cognitivo en personajes virtuales

Esta metodología (Figura 2) está basada en la jerarquía de modelado en animación comportamental, la cual ha dado como resultado personajes virtuales auto-animados dotados de comportamientos realistas [10,9,8,5].

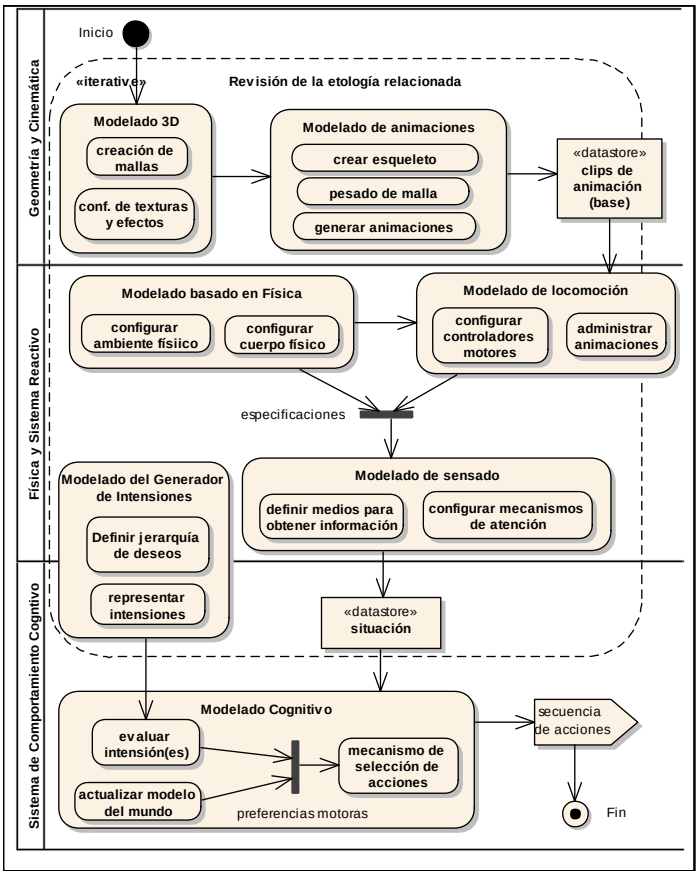



**Fig. 2.** Descripción de las actividades para el modelado cognitivo de personajes virtuales.

Esta metodología considera tres niveles de modelado (1) modelado de apariencia realista, (2) modelado de movimientos realistas, suaves y flexibles, (3) modelado de comportamientos realistas de alto nivel.

De acuerdo a lo que le atañe a este trabajo, por ahora sólo es necesario definir que un deseo es una influencia potencial para una intención. Un personaje puede tener múltiples deseos, pero solo aquel que sea más fuerte ó jerárquicamente más importante será considerado como una intención. La información obtenida del ambiente se representa como *recomendaciones cualitativas* para que una acción se realice. Dichas recomendaciones son conocidas como *preferencias motoras*.

### 5.2. Detalles de funcionamiento

El sistema reactivo es responsable de ejecutar acciones primarias (incluyendo acciones de sensado), pero este debe trabajar con el Generador de Intenciones y el Mecanismo de Selección de Acción para generar comportamientos inducidos. Con el objetivo de caracterizar conceptos difusos (deseos e intenciones) de forma que no se obtengan comportamientos 100 % deterministas, es recomendable que el generador de intenciones esté implementado con técnicas no clásicas para la toma de decisiones.

## 6. Modelado cognitivo de personajes

Se seleccionó un mundo submarino con el objetivo de analizar comportamientos elaborados relacionados con el paradigma de *vida artificial*. La implementación consiste de presas (peces) y un depredador (tiburón), plantas y diferentes obstáculos para probar las capacidades cognitivas inherentes a un pez. Este mundo fue implementado con la ayuda del motor de juegos ShiVa3D.

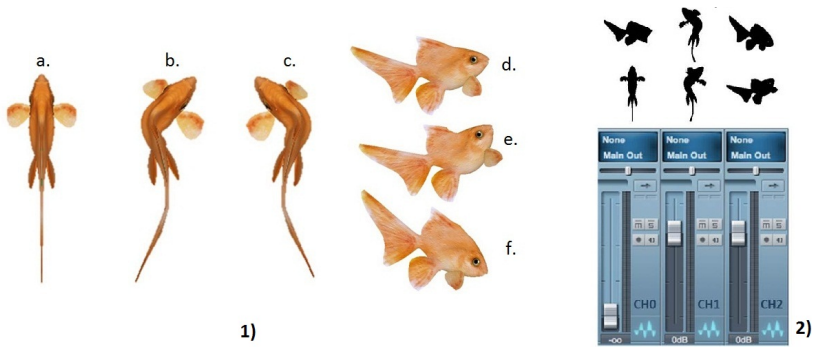



**Fig. 3.** 1) Movimientos atómicos (idle, derecha, izquierda, frenar, arriba, abajo). 2) Esquema propuesto para programar un administrador de animaciones efectivo.

En base al enfoque de H. Barthel en [2], esta implementación enfatiza la combinación de movimientos atómicos para generar secuencias de movimientos realistas. Se almacenan 6 animaciones simples (Figura 3, inciso 1) dentro de

una base de datos de movimientos donde más de uno puede ser ejecutado simultáneamente para generar movimientos compuestos.

Se define un enfoque factible que obtiene una combinación efectiva de animaciones. La idea es tener dos tipos de acciones: *exclusivas* y *no-exclusivas* (Figura 3, inciso 2). En este caso las animaciones mutuamente exclusivas no pueden reproducirse simultáneamente, por lo que se reproducen en el mismo canal de animación.

El sistema de percepción o sensado propuesto considera cuatro modelos:

1. Evadir colisión con el terreno.
2. Evadir colisión con obstáculos fijos.
3. Evadir colisión con obstáculos dinámicos.
4. Contacto físico.

La capacidad para *evadir de colisiones con el terreno* fue diseñada para ser activada sólo cuando la entidad se está moviendo. Debido a las características intrínsecas del terreno submarino, la implementación utiliza un método que traza rayos. Este método toma como referencia las coordenadas inferiores del *bounding-box* correspondiente al modelo 3D del personaje. Cuando el personaje avanza los rayos son trazados con respecto a las coordenadas delanteras y cuando retrocede los rayos son trazados con respecto a las coordenadas traseras.

La *evasión de colisión con obstáculos fijos* es utilizada cuando no se tiene una idea precisa de la forma, altura y posición de los mismos. Formas difíciles de reconocer son difíciles de procesar. Un pez real utiliza *puntos de referencia* para generar mapas mentales de relaciones geométricas [3]. Así que el cálculo del campo potencial atractivo define un objetivo en el espacio 2D definido por los ejes X y Z. Por su parte, el cálculo del campo potencial repulsivo es una opción viable para evadir colisiones con obstáculos fijos conocidos y que tienen formas demasiado irregulares (arboles, ramas, etc.).

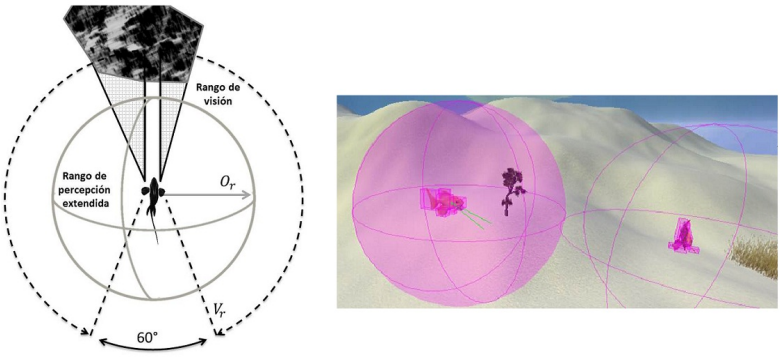



**Fig. 4.** Modelo de sensado que intenta simular habilidades especiales de percepción en un pez; rango de visión y rango de percepción extendida.

No obstante, la mayoría de los peces poseen órganos de percepción altamente desarrollados, como los *quimiorreceptores* o los receptores que forman el sistema de *linea lateral*, mediante los cuales detectan comida, suaves corrientes y vibraciones. De aquí la necesidad de modelar capacidades sensoriales similares. Capacidades como estas son útiles para *evadir la colisión con obstáculos dinámicos* e incluso para mantener el seguimiento de otros objetos (comida, peces, etc.). En la Figura 4 se muestra la propuesta para esta implementación, donde el rango de percepción extendida es un sensor esférico que detecta sensores similares.

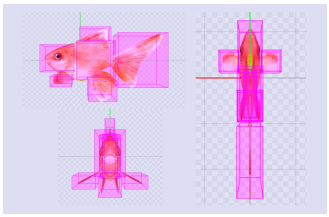


**Fig. 5.** Ragdoll del pez para simular la percepción del contacto físico.

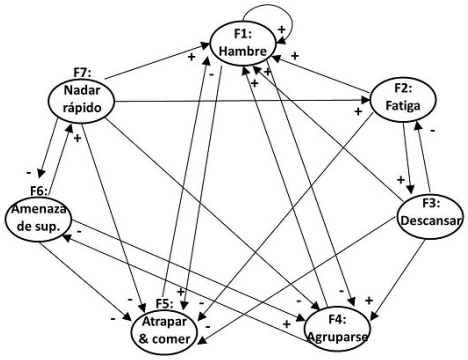
Por consiguiente es necesario simular el contacto físico con el objetivo de definir comportamientos más complejos o incluso reacciones simples. En este caso se utiliza un *ragdoll* (Figura 5) para poder simular la captura de comida y de peces, y para simular un comportamiento grupal libre de colisiones.

La jerarquía de intenciones considerada en la literatura [20] proporciona los fundamentos teóricos para proponer el siguiente esquema de esta implementación:

Intención de *evadir colisión*
⇓
Intención de *nadar rápido* para alejarse del depredador
⇓
Intención de *comer* / Intención de *descansar*
⇓
Intención de *agruparse*
⇓
Intención de *vagar* / Intención de *irse*

Sin embargo, existen otras variables relacionadas con estas intenciones consideradas como deseos o *influencias potenciales de intención* tales como: el *hambre*, la *fatiga* o la *amenaza de supervivencia*, los cuales cambian de manera impredecible con respecto al tiempo. Por lo tanto el generador de intenciones esta implementado con una técnica no clásica como lo es los Mapas Cognitivos Difusos (FCM). El FCM para modelar el generador de intenciones del pez es tomado del trabajo de Julie A. Dickerson y Bart Kosko en [6], el cual es mostrado en la Fig. 6. La red causal del Mapa Cognitivo Difuso esta formado por las reglas o aristas

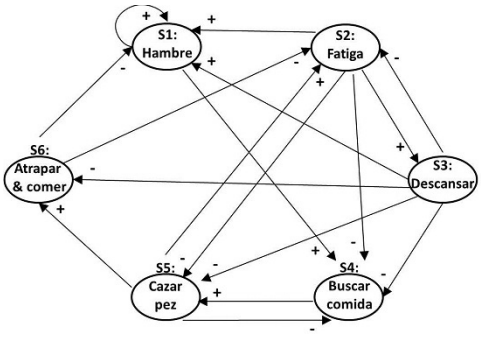
que conectan a los nodos o conceptos causales dentro de una matriz de conexión. Esta matriz puede contener valores en $\{-1, 0, 1\}$.




| | $F_1$ | $F_2$ | $F_3$ | $F_4$ | $F_5$ | $F_6$ | $F_7$ |
|---|---|---|---|---|---|---|---|
| $F_1$ | 1 | 0 | 0 | -1 | 1 | 0 | 0 |
| $F_2$ | 1 | 0 | 1 | 0 | -1 | 0 | 0 |
| $F_3$ | 1 | -1 | 0 | 1 | -1 | 0 | 0 |
| $F_4$ | 1 | 0 | 0 | 0 | 0 | -1 | 0 |
| $F_5$ | -1 | 0 | 0 | 0 | 0 | 0 | 0 |
| $F_6$ | 0 | 0 | 0 | 1 | -1 | 0 | 1 |
| $F_7$ | 1 | 1 | 0 | -1 | -1 | -1 | 0 |

**Fig. 6.** Mapa Cognitivo Difuso trivalente del generador de intenciones para un pez.

El FCM para modelar el generador de intenciones de un tiburón, está basado en el FCM propuesto en [6]. Sin embargo, cuando se dispara algún estado que hace referencia a los conceptos *buscar comida*, *cazar pez* o *atrapar & comer*, el flujo causal se estanca o se cicla sobre estos tres nodos. En base a los fundamentos teóricos, se modificó este modelo para obtener puntos de atracción más equilibrados sobre el FCM del tiburón. La Fig. 7 muestra el FCM propuesto y la matriz que representa la red causal del FCM de un tiburón.




| | $S_1$ | $S_2$ | $S_3$ | $S_4$ | $S_5$ | $S_6$ |
|---|---|---|---|---|---|---|
| $S_1$ | 1 | 0 | 0 | 1 | 0 | 0 |
| $S_2$ | 1 | 0 | 1 | -1 | -1 | 0 |
| $S_3$ | 1 | -1 | 0 | -1 | -1 | -1 |
| $S_4$ | 0 | 0 | 0 | 0 | 1 | 0 |
| $S_5$ | 0 | 1 | 0 | -1 | 0 | 1 |
| $S_6$ | -1 | -1 | 0 | 0 | 0 | 0 |

**Fig. 7.** Mapa Cognitivo Difuso trivalente propuesto para un tiburón.

A diferencia del uso que se le da a los FCM en [6], donde se utilizan *FCM's aumentados*, esta implementación propone utilizar FCM's simples, es decir, mapas cognitivos que no estén vinculados en una sola red causal de forma explícita.

Esto es posible gracias al soporte de *percepción-acción* que brinda la arquitectura MoCAMG. Además de que esto es congruente con el objetivo de alcanzar la autonomía. De lo contrario se estaría dotando a los personajes con la capacidad de saber explícitamente el estado mental de otro personaje. En general, el sistema de comportamiento está implementado como muestra el Algoritmo 1. Este procedimiento lo ejecuta tanto un pez como un tiburón, haciendo diferencias relativas dentro de cada uno de los métodos implementados. Este algoritmo sólo mencionan aquellos procesos vitales para el funcionamiento.

**Algorithm 1** Comportamiento Artificial

```
Inicializar variables globales
Cargar animaciones
Cargar Mapa Cognitivo Difuso
Cargar campo potencial repulsivo
Inicializar y activar sensores ragdoll
Inicializar sensor de percepción extendida
Obtener datos del Bounding-Box del modelo 3D
Obtener datos de rotación actual (ángulo en Y)
Inicializar variable Fatiga aleatoriamente
Inicializar variable Momento con el tiempo (frame/s)
for Cada frame do
    if Locomoción activa then
        Activar percepción extendida
        Actualizar información sensorial
        Evadir colisión con obstáculos
        Evadir colisión con terreno
        Actualizar movimiento                ▷ frenado, giro, desplazamiento
        Dirigirse hacia meta
    else
        Activar flotado
        Desactivar percepción extendida
        Decrementar variable Fatiga          ▷ factor alto de recuperación
    end if
    Generador de Intenciones                 ▷ arbitrar comportamientos
end for
```

## 7. Patrones de comportamiento obtenidos

Como resultado se observa cómo es que los Mapas Cognitivos Difusos de un pez y un tiburón se comportan. Se trató de evaluar de manera no formal el comportamiento de cada rol dentro del mundo virtual submarino implementado. Para un pez los deseos que evocan un comportamiento particular son *hambre*, *fatiga* y *amenaza de supervivencia*. Si un pez tiene *hambre* se dispara:

$C_1$ = [ 1 0 0 0 0 0 0 ] F1:hambre

$\rightarrow C_2 =$ [ 1 0 0 0 1 0 0 ] F1:hambre, F5:atrapar & comer
$\rightarrow C_3 =$ [ 0 0 0 0 1 0 0 ] F5:atrapar & comer
$\rightarrow C_4 =$ [ 0 0 0 0 0 0 0 ] vagar

Si un pez entra en un estado de *fatiga* el generador de intenciones dispara:

$C_1 =$ [ 0 1 0 0 0 0 0 ] F2:fatiga

$\rightarrow C_2 =$ [ 1 0 1 0 0 0 0 ] F1:hambre, F3:descansar
$\rightarrow C_3 =$ [ 1 0 0 0 0 0 0 ] F1:hambre
$\rightarrow C_4 =$ [ 1 0 0 0 1 0 0 ] F1:hambre, F5:atrapar&comer
$\rightarrow C_5 =$ [ 0 0 0 0 1 0 0 ] F5:atrapar & comer
$\rightarrow C_6 =$ [ 0 0 0 0 0 0 0 ] vagar

Si un pez entra en un estado de alerta por *amenaza de supervivencia* el generador de intenciones dispara:

$C_1 =$ [ 0 0 0 0 0 1 0 ] F6:amenaza de supervivencia

$\rightarrow C_2 =$ [ 0 0 0 1 0 0 1 ] F4:agruparse, F7:nadar rápido
$\rightarrow C_3 =$ [ 1 1 0 0 0 0 0 ] F1:hambre, F2:fatiga
$\rightarrow C_4 =$ [ 1 0 1 0 0 0 0 ] F1:hambre, F3:descansar
$\rightarrow C_5 =$ [ 1 0 0 0 0 0 0 ] F1:hambre
$\rightarrow C_6 =$ [ 1 0 0 0 1 0 0 ] F1:hambre, F5:atrapar & comer
$\rightarrow C_7 =$ [ 0 0 0 0 1 0 0 ] F5:atrapar & comer
$\rightarrow C_8 =$ [ 0 0 0 0 0 0 0 ] vagar

Tal como lo menciona la teoría, es posible observar que existen ciclos limitados que muestran patrones ocultos de comportamiento. Incluso algunos ciclos están contenidos dentro de otros al disparar un estado diferente. A simple vista se puede asegurar que estos patrones son congruentes con el comportamiento de un pez. Es importante decir que la intención de *vagar* esta representado por un estado vacío (sólo ceros) y este determina el punto de paro del ciclo.

Para el caso de un tiburón los deseos que evocan un comportamiento particular solamente son *hambre* y *fatiga*. Si un tiburón tiene *hambre* el generador de intenciones dispara:

$C_1 =$ [ 1 0 0 0 0 0 ] S1:hambre

$\rightarrow C_2 =$ [ 1 0 0 1 0 0 ] S1:hambre, S4:buscar comida
$\rightarrow C_3 =$ [ 1 0 0 1 1 0 ] S1:hambre, S4:buscar comida, S5:casar pez
$\rightarrow C_4 =$ [ 1 1 0 0 1 1 ] S1:hambre, S2:fatiga, S5:casar pez, S6:atrapar & comer
$\rightarrow C_5 =$ [ 1 0 1 0 0 1 ] S1:hambre, S3:descansar, S6:atrapar & comer
$\rightarrow C_6 =$ [ 1 0 0 0 0 0 ] S1:hambre

Si el tiburón entra en estado de *fatiga* el generador de intenciones dispara:

$C_1 =$ [ 0 1 0 0 0 0 ] S2:fatiga

$\rightarrow C_2 =$ [ 1 0 1 0 0 0 ] S1:hambre, S3:descansar
$\rightarrow C_3 =$ [ 1 0 0 0 0 0 ] S1:hambre
$\rightarrow C_4 =$ [ 1 0 0 1 0 0 ] S1:hambre, S4:buscar comida
$\rightarrow C_5 =$ [ 1 0 0 1 1 0 ] S1:hambre, S4:buscar comida, S5:casar pez
$\rightarrow C_6 =$ [ 1 1 0 0 1 1 ] S1:hambre, S2:fatiga, S5:casar pez, S6:atrapar & comer
$\rightarrow C_7 =$ [ 1 0 1 0 0 1 ] S1:hambre, S3:descansar, S6:atrapar & comer
$\rightarrow C_8 =$ [ 1 0 0 0 0 0 ] S1:hambre

A diferencia de un pez este ciclo no está limitado por un estado vacío. De hecho el deseo *hambre* siempre está presente. Por lo tanto se puede inferir que para un tiburón el estado [100100] (S1:hambre, S4:buscar comida) es equivalente a la intención de vagar de un pez. Además cada patrón de comportamiento lo describe un ciclo limitado por el estado simple [100100] (S1:hambre). Esto es comprensible hasta cierto punto, ya que algo que caracteriza a un tiburón es mantener siempre un comportamiento agresivo. Nótese que la *jerarquía de intenciones* definida previamente es un factor clave para generar un mecanismo de selección de acción efectivo. Principalmente cuando se genera un estado con 2 o más intenciones simultaneas.

## 8. Conclusiones y trabajo futuro

Apoyados en la arquitectura MoCAMG, una representación del conocimiento sencilla pero útil (FCM's simples), y las herramientas de sensado y física que proporciona un motor de juegos se logró comprobar que [a menudo] *es posible descomponer patrones complejos de comportamiento en unidades más pequeñas, algunas de las cuales son inmediatamente comparables con reflejos. Y que frecuentemente el comportamiento es organizado de una forma jerárquica; así que los sistemas de control superior deben competir por el control de los reflejos más que los reflejos competir por el control de los músculos.* Tal como lo dice A. Manning en [16].

Cabe mencionar que hay pocas arquitecturas cognitivas enfocadas en videojuegos, la mayoría están enfocadas en procesos detallados relacionados con la psicología humana. Sin embargo, la arquitectura y la metodología utilizadas probaron ser útiles para este enfoque. Además, el ambiente virtual implementado es una base importante para probar procesos cognitivos más complejos.

Cómo trabajo futuro, se planea que la situación de una entidad cognitiva (estado interno y externo) y una respuesta apropiada (acción) sea almacenada como *instancia* en la memoria de largo plazo, procesada por el módulo nombrado Procedimiento-IBL (Figura 1). De forma similar a como lo realiza la IBLTool de V. Dutt y C. González en [7]. La correcta descripción de estas instancias tiene el objetivo de lograr un mejor rendimiento y respuestas más ponderadas por parte del mecanismo de selección de acción. En pocas palabras, evolucionar hacia un sistema de razonamiento.

## Referencias


1. Anderson, J.: The Architecture of Cognition. Cognitive science series, Lawrence Erlbaum Associates (1996)
2. Barthel, H., Dannenmann, P., Hagen, H.: Towards a General Framework for Animating Cognitive Characters. In: Proceedings of the 3rd IASTED International Conference on Visualization, Imaging, and Image Processing. VIIP'03, Benalmádena, Spain (2003)
3. Chung, S.: Appropriate maze methodology to study learning in fish. University of Toronto Journal of Undergraduate Life Sciences 2(1) (2009)
4. Conde Ramírez, J.C., Sánchez López, A., Sánchez Flores, A.: An architecture for cognitive modeling to support real-time adaptation and motivational responses in video games. In: MICAI (1). Lecture Notes in Computer Science, vol. 8265, pp. 144–156. Springer, Mexico City, Mexico (2013)
5. Dannenmann, P., Barthel, H., Hagen, H.: Multi level Control of Cognitive Characters in Virtual Environments. In: Proceedings of the 14th IEEE Visualization 2003 (VIS'03). pp. 92–. VIS '03, IEEE Computer Society, Washington, DC, USA (2003)
6. Dickerson, J.A., Kosko, B.: Virtual Worlds as Fuzzy Cognitive Maps. Presence 3(2), 173–189 (1994)
7. Dutt, V., Gonzalez, C.: Making Instance-based Learning Theory usable and understandable: The Instance-based Learning Tool. Comput. Hum. Behavior 28(4), 1227–1240 (Jul 2012)
8. Funge, J.: Cognitive modeling for games and animation. Commun. ACM 43(7), 40–48 (Jul 2000)
9. Funge, J., Tu, X., Terzopoulos, D.: Cognitive modeling: knowledge, reasoning and planning for intelligent characters. In: Proceedings of the 26th annual conference on Computer graphics and interactive techniques. pp. 29–38. SIGGRAPH '99, ACM Press/Addison-Wesley Publishing Co., New York, USA (1999)
10. Funge, J.D.: Making them behave: cognitive models for computer animation. Ph.D. thesis, University of Toronto, Toronto, Ont., Canada (1998)
11. Giang, T., Mooney, R., Peters, C.: Real-time character animation techniques. Tech. rep., Image Synthesis Group Trinity College Dublin (2000)
12. Hudlicka, E.: Affective game engines: motivation and requirements. In: Proceedings of the 4th International Conference on Foundations of Digital Games. pp. 299–306. FDG '09, ACM, New York, NY, USA (2009)
13. Kim, M.: The artificial emotion model of game character through analysis of cognitive situation. In: Proceedings of the 2009 Fourth International Conference on Computer Sciences and Convergence Information Technology. pp. 489–493. ICCIT '09, IEEE Computer Society, Washington, DC, USA (2009)
14. Kosko, B.: Fuzzy cognitive maps. International Journal of Man-Machine Studies (24), 65–75 (1986)
15. Magnenat-Thalmann, N., Thalmann, D.: Virtual humans: thirty years of research, what next? The Visual Computer 21(12), 997–1015 (2005)
16. Manning, A., Dawkins, M.: An Introduction to Animal Behaviour. Cambridge University Press (1998)
17. McFarland, D.: Animal behaviour: psychobiology, ethology, and evolution. Longman Scientific and Technical (1993)
18. Ritter, F.E.: Two cognitive modeling frontiers Emotions and usability. Transactions of the Japanese Society for Artificial Intelligence 24(2), 241–249 (2009)

19. Sun, R.: Theoretical status of computational cognitive modeling. Cogn. Syst. Res. 10(2), 124–140 (Jun 2009)
20. Tu, X.: Artificial animals for computer animation: biomechanics, locomotion, perception, and behavior. Springer-Verlag, Berlin, Heidelberg (1999)
21. Yan, W., Kalay, Y.: Geometric, cognitive and behavioral modeling of environmental users. In: GERO, J. (ed.) Design Computing and Cognition ́06, pp. 61–79. Springer Netherlands (2006)

# Comunicando de manera efectiva un ambiente virtual de aprendizaje y algoritmos de planificación de itinerarios formativos: una propuesta arquitectónica basada en agentes

Lluvia Morales, José Figueroa, Marvelia Gizé Jiménez Guzmán,
Felipe Trujillo-Romero, Óscar Pérez

Universidad Tecnológica de la Mixteca,
México

lluviamorales@mixteco.utm.mx, jfigueroa@mixteco.utm.mx, marvgize@gmail.com,
ftrujillo@mixteco.utm.mx, oscaruriel@gmail.com

**Resumen.** En el contexto de E-Learning existe un número sumamente importante de propuestas en las que los agentes inteligentes se utilizan para diversas tareas; desde virtualizar a los estudiantes para analizar su comportamiento, hasta proponerles nuevas actividades a realizar dependiendo de su interacción con el sistema, entre otras. Este artículo describe una arquitectura de comunicación entre un ambiente virtual de aprendizaje y un conjunto de servicios Web de Planificación y Scheduling, con el fin de crear itinerarios formativos adaptados a diferentes perfiles de estudiante de un curso. El puente de comunicación entre estos dos componentes es un conjunto de agentes que permiten: (1) modularizar la información que se pueda recabar del ambiente virtual, (2) comunicar la información al planificador inteligente con la estructura apropiada y (3) recuperar los itinerarios formativos dados por el planificador o los problemas del dominio encontrados por los agentes, y enviarlos de manera clara a los usuarios interesados.

**Palabras clave:** Arquitectura de agentes, planificación y scheduling, e-learning adaptativo, ambientes virtuales de aprendizaje.

## 1. Introducción

Los agentes inteligentes han sido utilizados ampliamente como apoyo en diversas tareas necesarias durante el proceso de aprendizaje en línea. Los problemas menos atacados hasta el momento según [9] han sido los de generación de itinerarios formativos adaptados al usuario y el uso de funcionalidades externas dentro del proceso de aprendizaje. Sin embargo, la gestión de información del usuario e inferencia acerca de la misma, es el tema prioritario de los mismos; pero bajo un enfoque de simulación de usuarios o administración de recursos inherentes a los mismos.

Desde nuestro punto de vista, la información que habitualmente gestionan y procesan los agentes en la educación, debería ser utilizada por aplicaciones externas que permitan llevar a cabo con mayor eficacia y rapidez el proceso de generación de itinerarios formativos adaptados a las diferentes necesidades y características de los estudiantes.

Es así como en este artículo se propone una arquitectura de agentes que permita, a un ambiente virtual de aprendizaje, acceder a un conjunto de servicios Web de planificación que permiten la generación en tiempo real de itinerarios formativos adaptados a diversos perfiles de estudiantes. Los agentes no solo se encargan de (1) procesar la información dinámica que se genera durante la interacción de los usuarios con el ambiente virtual de aprendizaje, sino también de (2) obtener inferencias en base a la misma y decidir si se solicita o no un itinerario formativo nuevo o modificado. También se encargan de (3) compilar la información recibida desde el Ambiente Virtual de Aprendizaje (VLE, por sus siglas en inglés) de manera que, en caso de ser necesario un itinerario adaptado, sea comprendida por los servicios Web de planificación inteligente. Y, finalmente, (4) comunicar los posibles resultados de la planificación o necesidades/incongruencia en la información recibida por las diferentes fuentes de manera que los itinerarios formativos sean del agrado de los estudiantes.

En las siguientes secciones se explica primero que nada el trabajo relacionado con el nuestro en cuestión tecnológica y de resolución del problema particular. Enseguida se sitúa al usuario en el contexto general de aplicación de los agentes en la educación y posteriormente en nuestro caso particular sobre el cual justificamos nuestra propuesta. Siguiendo en esa línea, la propuesta general de arquitectura de agentes, incluyendo los diferentes módulos de agentes y los componentes que se interconectan, se describe en la sección 4, la cual se ve reflejada en unas pruebas iniciales de aplicación en la sección 5 y, finalmente, se resumen nuestras conclusiones y trabajo futuro en la sección 6.

## 2. Trabajo relacionado

Como primera parte de este trabajo, se plantea una arquitectura distribuida de agentes, había que encontrar la manera de comunicar a los diferentes elementos de la arquitectura, es así como en [4] podemos observar un excelente ejemplo de uso de servicios Web, XML y RDF como elementos de comunicación entre agentes móviles y en [6] se utiliza XML para transmitir y procesar información entre el ambiente de aprendizaje y los agentes. Sin embargo, a nosotros no nos interesa que los agentes sean móviles, ya que no es necesario que trabajen directamente sobre la interfaz del usuario si no con la información que se transmite a la interfaz a partir de lo almacenado en el LMS.

Tampoco nos interesa que el problema de planificación de itinerarios formativos se resuelva por entero a través del uso de agentes, como ya se propuso en [7] o en [6] donde se apoyan de una ontología que representa un flujo de trabajo o tareas para facilitarse el trabajo. Según el paradigma de agentes, no se propone que el agente

resuelva todo si no que sea una herramienta de apoyo para la comunicación entre éstos resolvedores y el usuario de manera que los mismos agentes hagan uso de herramientas especializadas de inteligencia artificial, en este caso planificación inteligente, pues éstas ya resuelven el grueso del problema de los itinerarios formativos y no queremos reinventar la rueda, si no mejorar lo que ya existe.

Pero, para eso, necesitamos asegurarnos que los agentes sean realmente inteligentes, ya que como se describe en [1] los agentes inteligentes son programas de computadora que aplican un alto grado de inteligencia para poder llevar a cabo tareas de manera independiente sin ninguna supervisión.

Existen, por otro lado, varios ejemplos de agentes que permiten reutilizar información de la web para personalizar el material mostrado al estudiante. Algunos se basan en las necesidades detectadas en el curso a través de las acciones de profesores y alumnos sobre el mismo [13], otros no solo utilizan a los agentes para recuperar el contenido si no que emplean otros agentes especializados para saber cómo mostrárselo a los usuarios [12].

Otros agentes permiten inferir información como sus niveles de conocimiento o actividad del estudiante a partir de actividad durante un período de tiempo sobre algún tópico particular [3] y agentes como el de [10] también consideran las reacciones o gestos del alumno frete a distintas actividades para determinar las actividades más convenientes para algún alumno.

Existen agentes que permiten recuperar el perfil del usuario en tiempo real e ir sugiriendo actividades una a una [2] lo cual no empata con la tarea que realiza una planificador del tipo que utilizaremos. Sin embargo, si con la recuperación del perfil y avances del usuario sobre su secuencia de actividades pre-planificada en tiempo real. Pero la mayoría de los agentes que podemos encontrar en la bibliografía, como es el caso de [14] trabajan sobre un curso específico y reglas pedagógicas predeterminadas para controlar la dinámica del curso y, cada vez es más común ver que los mismos trabajan sobre la Web.

## 3. Problema y justificación

El presente capítulo pretende resumir y plantear la problemática que se tiene a pesar del gran trabajo realizado a lo largo de tantos años. Gran parte de la información mostrada en el mismo se ha extraído del trabajo previo de [9]. Así, según Steffen Mencke y Reiner R. Dumke, es posible y beneficioso integrar agentes en un sistema cuando:

- Existen subsistemas y componentes de los subsistemas formando el sistema.
- Existe un gran número de interacciones entre un subsistema y sus componentes en términos de tamaño y complejidad.
- Las interrelaciones cambian a lo largo del tiempo.

Las características anteriores son inherentes a un sistema de e-learning que permite interactuar a diferentes usuarios (estudiantes, profesores y administradores), diferentes tipos de usuarios (tipos de estudiantes, diferentes métodos de enseñanza de

los profesores) que interactúan de formas distintas y diferente interacción y gusto por el sistema dependiendo del tema que se pretende aprender, respectivamente, según las características anteriores, por decir algunas.

La mayoría de los agentes se aplican en e-learning para apoyar/mejorar principalmente los siguientes aspectos del sistema:

– Relacionados con **el usuario**
– De funcionalidad interna
– **De funcionalidad externa**
– De **datos internos**

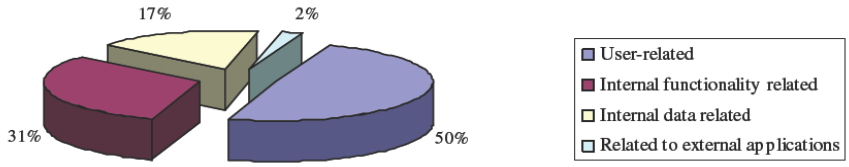



**Fig. 1.** Diferentes actividades llevadas a cabo por los agentes en sistemas de e-learning (tomado de [9]).

Como se puede observar en la ilustración anterior, la mayor parte de las implementaciones de agentes se centran en el usuario, mientras que no pretenden comunicarse con aplicaciones externas para mejorar la experiencia del mismo porque, como dijimos anteriormente estos sistemas suelen estar hechos a la medida para cursos y LMSs específicos.

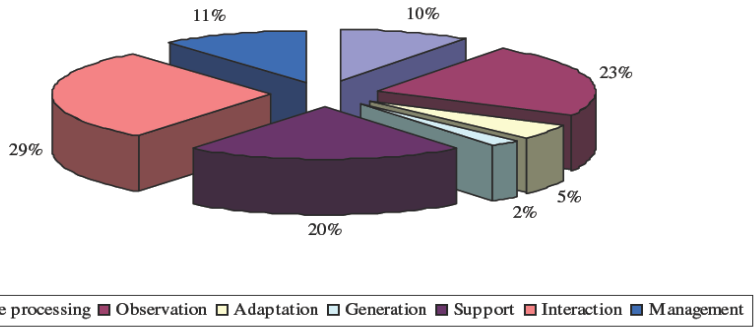



**Fig. 2.** Funcionalidades que cubren los agentes en los sistemas e-learning (tomado de [9])

Los agentes suelen apoyar a los ambientes de aprendizaje a través de las siguientes acciones:

– *Procesamiento de información/conocimiento*: unidades de aprendizaje o evaluación de datos.
– *Observación*: de usuarios, objetos de aprendizaje, recursos de conocimiento y/o artefactos del sistema.

- *Adaptación*: de secuencias de aprendizaje.
- *Generación:* de secuencias de aprendizaje.
- *Soporte*: toma de decisiones, tutoría, recomendaciones, capacidades de búsqueda.
- *Interacción:* mostrar conocimiento, notificar, motivar y los objetivos que tengan las interfaces humano-computadora.

Como podemos ver en la ilustración anterior, los agentes casi no tienden a adaptar ni generar secuencias de aprendizaje ya que no es el fuerte de los mismos, se dedican en su mayoría a observar, interactuar y dar soporte a las actividades que tienen que ver directamente con los usuarios, y menor medida a procesar y gestionar conocimiento.

Así, nos encontramos con que un sistema en el que se pretende generar y adaptar secuencias de actividades de aprendizaje para un curso genérico, debiera estar integrado por una gran variedad de agentes que no necesariamente estén relacionados con la generación y adaptación que ya de por sí se lleva a cabo por algoritmos especializados, si no con la recuperación y gestión de conocimiento, así como la interacción apropiada con la interfaz del ambiente de aprendizaje, que es algo que se ha descuidado en trabajos previos dentro de entornos "libres de agentes" y que no consideran la usabilidad [11] e iniciativa mixta [5] como un punto clave durante este proceso.

Para que quede más claro y que sea posible justificar la complejidad del mismo, es preciso describir algunos casos representativos en los que se hace patente la necesidad de "algo" que dé soporte a la actividad planteada.

### 3.1. Caso Uno: planificación

Dentro de la arquitectura original para generar itinerarios formativos adaptados, una vez que el profesor ha terminado de describir y relacionar todas las actividades de un curso y que todos los estudiantes están registrados y han rellenado su perfil dentro de la plataforma, es tiempo de preguntar al planificador inteligente que genere un nuevo plan adaptado para todos los estudiantes.

Si durante la planificación, el planificador inteligente no hubiera podido generar los planes, el sistema simplemente muestra un mensaje de error. Sin embargo, son varias las razones que podrían llevar a esta situación y que, hasta ahora, han tenido que ser resueltas por expertos humanos:

- Que las relaciones entre las actividades del curso generen un ciclo al momento de requerirse unas a otras.
- Que existan actividades que tengan prerrequisitos que los estudiantes no pueden cumplir o no han expresado claramente en su perfil.
- Que existan actividades con el mismo nombre y/o características.

Estas y otras problemáticas han tenido que ser detectadas por expertos en planificación quienes solicitan al profesor que cambie las relaciones o los nombres de las actividades o que genere/busque nuevas actividades que puedan ser más accesibles a cierto grupo de estudiantes con características que no se habían considerado

inicialmente. Una vez resueltas estas problemáticas, se ha solicitado de nuevo la generación de estas secuencias, pero no sin antes haber gastado una gran cantidad de horas-hombre en resolver este problema "técnico".

Después de una planificación "exitosa", los planes se muestran a los estudiantes. Sin embargo, en ocasiones existen planes con los cuales el profesor no está de acuerdo, o incluso el mismo estudiante, y no existe una vía para comunicar esta situación al sistema y cambiar el plan propuesto por el planificador inteligente, o revisarlo antes de ser ejecutado.

Un conjunto de agentes que detectaran dichas ambigüedades en la especificación previa del curso y otros que mostraran y/o permitieran al profesor y estudiante comunicar sus inconformidades con respecto al plan propuesto inicialmente, evitarían un gasto en tiempo y dinero innecesarios y facilitarían la independencia del profesor en el uso del sistema inteligente.

### 3.2. Caso Dos: monitorizando a los estudiantes

Durante la ejecución de su itinerario de aprendizaje adaptado, un estudiante puede sufrir diversos cambios tanto en su perfil como en su rendimiento con respecto a un tema que es posible que invaliden el itinerario formativo personalizado que se había propuesto inicialmente para él, por ejemplo:

- Su nivel de idiomas puede haber aumentado drásticamente a la mitad del curso, sobre todo en el caso de un curso de idiomas.
- No logró obtener un resultado suficientemente satisfactorio en una actividad de evaluación y, por tanto, no puede llevar a cabo las actividades que dependen de la misma.
- Tuvo que entrar a trabajar.
- Se enfermó durante un período considerable de tiempo.
- Consiguió acceso a dispositivos electrónicos tipo altavoces o micrófono.

En este caso podemos ver como problemas temporales, técnicos o de falta/sobra de conocimientos, también se hacen necesaria la intervención de una entidad que detecte estas situaciones y que, si las mismas pudieran afectar notablemente el itinerario del estudiante, se encarguen de modificarlo y comunicárselo al estudiante antes y después de hacerlo; pero sin afectar las actividades ya realizadas.

### 3.3. Caso Tres: monitorizando las sugerencias del profesor

Después de la generación de los itinerarios formativos adaptados para cada estudiante, el profesor debería tener la oportunidad (como ya dijimos anteriormente) de validar dichos itinerarios personalizados, es decir, que manifieste su agrado con respecto a lo propuesto por el sistema inteligente y se asegure de que se le proporcionen al estudiante las actividades suficientes y necesarias para que le sea posible aprender todos los temas del curso.

También se puede dar el caso de que al analizar esos itinerarios, es posible que el profesor decida que no le parecen adecuados la mayoría de los itinerarios, en este caso habría de cambiar las características del curso y/o agregar y/o borrar actividades, según considere pertinente. Por lo tanto, después de estas modificaciones y antes de iniciar el curso, deberá mandar a generar itinerarios adaptados de nuevo.

Por último, las modificaciones a las características del curso, tanto por cambiar propiedades o jerarquías de las actividades como por borrar o agregar nuevas actividades o temas, pueden ocurrir también durante la ejecución de los itinerarios formativos. En este caso, es necesaria una entidad (agente) que revise el avance de cada estudiante en cada una de sus actividades propuestas, detecte los cambios sugeridos por el profesor y se asegure de que el planificador inteligente genere nuevos itinerarios que consideren esos planes, pero que también consideren una variación mínima en las porciones de itinerarios adaptados ya ejecutadas.

Es así como, dados los problemas planteados anteriormente se pone de manifiesto la necesidad de un ente o conjunto de entidades que detecten inconsistencias en la generación y ejecución de los planes y sus usuarios, de manera que mantenga la consistencia adaptativa de los mismos durante el desarrollo de todo el curso y teniendo siempre en cuenta el visto bueno del profesor.

## 4. Arquitectura de agentes

En esta sección describiremos de manera general la arquitectura, tipos de agentes y comunicación que debe existir entre ellos para resolver el problema de generación de itinerarios formativos adaptados utilizando un planificador inteligente. Recordemos que el principal objetivo de esta arquitectura es que se considere la correcta y suficiente comunicación con los usuarios, además de, sus opiniones y sugerencias acerca de los itinerarios, así como los cambios de las características del curso y usuarios a lo largo de la ejecución de las actividades de los mismos.

En las sub-secciones correspondientes a cada subconjunto de agentes, se describirá un poco más a detalle el comportamiento y propósito de cada agente, así como las técnicas de inteligencia artificial que habrán de requerir (en su caso) para aprender lo necesario para su correcto funcionamiento.

En la Fig. 3 podemos observar la arquitectura general de agentes que pretendemos comuniquen el Sistema Gestor de Aprendizaje con el Servidor de Inteligencia Artificial que contiene las diferentes funciones de planificación inteligentes necesarias para crear itinerarios formativos personalizados.

Primero que nada, cabe mencionar que el servidor del ambiente virtual de aprendizaje puede ser cualquier Sistema Gestor de Aprendizaje (LMS del inglés Learning Management System) o VLE (Virtual Learning Environment) que solicite el uso de los agentes y pueda proporcionarles la información suficiente en el formato especificado como para que se puedan llevar a cabo los procesos de inferencia y planificación (los agentes del módulo de Procesamiento de información en tiempo real se encargan de proporcionar el formato de envío y de mencionar las necesidades mínimas de información al ambiente). También será necesario, por parte del VLE,

que sea capaz de desplegar, en el formato que considere apropiado, los itinerarios formativos resultantes y preguntar al usuario su grado de satisfacción respecto a los itinerarios propuestos.

Por otro lado, el servidor Web de planificación deberá permitir elegir entre la gran variedad de planificadores, re-planificadores y planificadores basados en casos que utilizan el formato estándar de PDDL (Planning Domain Defnition Language) [8] para su funcionamiento de manera que a alguno de ellos podamos solicitar el o los itinerarios formativos necesarios y de proporcionar un acceso al validador de problemas de dicho estándar.

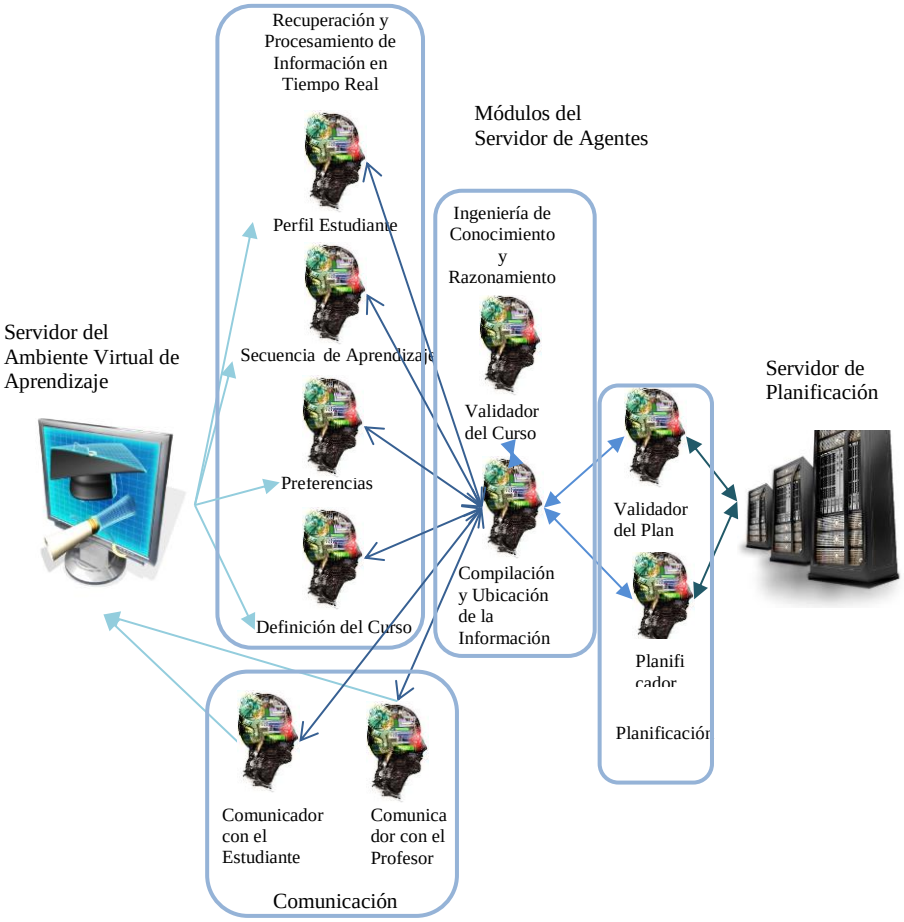



**Fig. 3.** Arquitectura General de nuestra propuesta.

En lo que concierne a la arquitectura de agentes propuesta en este artículo, la misma se divide en cuatro subconjuntos de agentes:

- *Agentes que Recuperan Información del LMS en Tiempo Real*.

Se comunican con el VLE para recibir información acerca de:
- El perfil de los estudiantes y sus cambios durante el desarrollo del curso,
- Los cambios que haga el profesor al curso,
- La forma de actuar de cada estudiante en cada una de las actividades asignadas dentro del VLE,
- Cada una de las actividades del itinerario formativo personalizado de cada estudiante,
- La conformidad o no del profesor respecto a los planes o itinerarios formativos propuestos por los agentes, entre otras.

También se encarga de comunicar dicha información en forma de lógica temporal de primer orden a los agentes de ingeniería de conocimiento.

- *Agentes para la Ingeniería de Conocimiento y Razonamiento*:
  Se encargan de procesar y distribuir información en tiempo real entre los otros agentes de la arquitectura. El procesamiento de información corresponde a la ordenación en uno o varios archivos de los predicados recibidos a partir de los agentes del conjunto anterior y al envío de dicha información a los agentes de planificación. También recibe los planes y/o mensajes de los agentes de planificación y los compila, de manera que puedan ser interpretados y procesados por los agentes de comunicación.
- *Agentes de Planificación*:
  Se encargan de analizar la información proporcionada por los dos grupos de agentes anteriores para revisar si es válida, proporcionar dicha información al servidor de inteligencia artificial y recibir los planes y/o respuestas del servidor para validarlos y enviarlos a los agentes de ingeniería de conocimiento.
- *Agentes de Comunicación*:
  Se encargan de comunicar la información sobre la posible generación o no de los itinerarios formativos a los diferentes usuarios del sistema. En caso de que los planes no hayan sido creados por alguna razón, también se encargan de comunicar dicha razón de la manera más apropiada y solicitar la solución de esta situación en caso de que al usuario le corresponda intervenir.

A continuación describiremos más a detalle los módulos mencionados en los puntos anteriores.

### 4.1. Módulo de agentes de recuperación de información en tiempo real

Este conjunto de agentes está compuesto por cuatro elementos que reciben documentos en XML y traducen la información encontrada en estos documentos a predicados en lógica de primer orden:

- El agente que *recupera información del perfil del estudiante* lo hace al inicio del curso, pero también durante la ejecución de sus actividades personalizadas.
  El comportamiento inteligente de este agente radica en que cada estudiante puede tener características distintas, a pesar de que los campos del perfil del estudiante pudieran ser estáticos, la interfaz permite al estudiante y al profesor definir nueva información que podría resultar interesante dentro de su perfil y el agente tiene que

aprender a traducirla a un predicado. Además, permite recuperar la opinión del estudiante acerca de su itinerario personalizado.

- El agente *Secuencia de Aprendizaje* se encarga de guardar las secuencias de actividades de aprendizaje para cada estudiante y de revisar constantemente el progreso de cada uno de ellos en las mismas. Si el progreso es negativo lo comunica al agente de ingeniería de conocimiento.
- El agente de *preferencias del profesor*, permite descartar itinerarios que el profesor considere inviables o pobres, o recomendar itinerarios que en pasadas ocasiones le hayan parecido muy buenos a ese profesor en particular y para ese curso específico. Es decir, guarda las preferencias de cada profesor para cada itinerario de cada curso que imparta.
- El agente de *definición del curso*, permite recuperar toda la información disponible del curso, tanto al inicio del mismo como mientras se ejecuta. Porque, como dijimos anteriormente el profesor puede modificarlo durante la ejecución del mismo. En caso de cambios, los comunica al agente de ingeniería de conocimiento.

### 4.2. Módulo de ingeniería de conocimiento

Los agentes de ingeniería de conocimiento son solo dos, pero en ellos se concentra el motor y vehículo de toda la información que recorre la arquitectura.

- El *agente validador* del curso permite revisar que los usuarios hayan definido las instancias mínimas de requisitos del curso y el estudiante, de manera que el planificador, en su momento, pueda evaluar la información proporcionada. Es un primer paso antes de la validación "inteligente" que lleva a cabo el planificador.
- El agente de *traducción y colocación* pretende organizar los predicados recibidos por los agentes de recuperación de información en archivos descritos en lenguaje de planificación para comunicarlos a los agentes de planificación, también traducir los recibidos por los agentes de planificación a sentencias que puedan ser comprendidas por los agentes de comunicación o, en base a lo comunicado por el agente validador y los planificadores, enviar los mensajes correspondientes a los comunicadores.

### 4.3. Módulo de planificación

Los agentes de planificación son los encargados de comunicarse con los diferentes servicios de planificación y validación inteligente de planes.

- El *agente validador* se comunica con la herramienta inteligente de validación de planes y le cuestiona sobre la validez actual de un itinerario formativo de acuerdo a los nuevos objetos de aprendizaje que se han introducido o a las nuevas características del curso. Para aclarar, las diferencias con la validación anterior, radican en que ahora si se analiza la información y relaciones de la misma, en la anterior solo se revisaba que dicha información existiera y fuera válida.

En caso de invalidez, lo comunica al agente planificador para que genere un nuevo plan o a los agentes de ingeniería de conocimiento para que proporcionen dicha información a los interesados.

- El *agente planificador* se encarga de elegir al planificador más conveniente en cada caso, en caso de que sea posible planificar. Por tanto necesita un motor de aprendizaje que le indique cuáles son las situaciones en que necesita determinado tipo de planificador, ya sea para reutilizar planes ya existentes, generar planes en el menor tiempo posible, generar planes nuevos con el mínimo de cambios posible pero que sean válidos, etc.

### 4.4. Módulo de comunicación

Estos agentes se encargan de decidir cuál información de la proporcionada por los agentes de ingeniería de conocimiento va dirigida a qué usuario y en qué formato. Deciden cómo mostrar, en lenguaje natural, los itinerarios formativos adaptados y sus detalles de ejecución a cada estudiante y al profesor. También deciden cómo mostrar errores en la generación de estos itinerarios al profesor o al estudiante, dependiendo del tipo de error encontrado, así como advertencias y necesidades respecto a la incorrecta ejecución del plan o la definición del perfil o el curso respectivamente.

Con esto terminamos de describir cada uno de los conjuntos de agentes que integran la arquitectura. A continuación se describirán las pruebas que se han llevado a cabo hasta ahora sobre las interface que muestran los resultados de interacción de los agentes con los usuarios (profesor y estudiante).

## 5. Pruebas iniciales

Dado que este es un trabajo en proceso, las pruebas iniciales del sistema se han llevado a cabo sólo con las interfaces gráficas desarrolladas sobre el servidor de nuestro ambiente virtual de aprendizaje. Las interfaces desarrolladas para este sistema, están enfocadas a la enseñanza/aprendizaje, respectivamente, de Inglés. No contamos con pruebas particulares de razonamiento de cada uno de los agentes por separado.

En las últimas pruebas de usabilidad de las interfaces participaron siete profesores de inglés y 10 estudiantes de diferentes niveles de licenciatura que cursan el nivel PETB (intermedio) de inglés. Se les asignaron una serie de tareas a desarrollar de manera que utilizaran todas las funciones del sistema y, finalmente, se les hicieron una serie de preguntas relacionadas con dichas tareas; no solo con la cuestión estética de las interfaces, sino también con las funciones interactivas relacionadas con la adaptación de los itinerarios formativos a los diferentes tipos de estudiantes clasificados. Cabe recordar que la adaptación de los itinerarios formativos que toma en cuenta el planificador inteligente se basa en factores como el estilo de aprendizaje, el desempeño en el curso, los recursos tecnológicos con los que cuenta el estudiante, entre otros factores.

A pesar que de la cantidad de usuarios empleados durante las pruebas, son pocos, se pudieron obtener resultados preliminares interesantes y de utilidad para el desarrollo del trabajo planteado. Los profesores comentaron que "el sistema" les puede ayudar a evaluar y dar seguimiento más fácilmente a su gran número de estudiantes, de manera automática. Y que es importante la personalización del material porque eso ayudará a sus estudiantes a aprender mejor.

En el caso de los alumnos, se mostraron interesados en conocer su estilo de aprendizaje, nos sugirieron aumentar la información proporcionada acerca de cada una de las actividades propuestas y de su progreso en el curso y en su nivel de inglés en general, y les agradó el material personalizado que se les mostró ya que cumplía con las necesidades y requerimientos particulares que se habían capturado durante la prueba. Algo que es interesante mencionar es que el 20% de los estudiantes llegó a confundir la herramienta con un sustituto del profesor para sus clases de inglés, a pesar de que sugirieron que se incluyera un poco más de interacción y animaciones. Recordemos que el propósito de la misma no es sustituir al profesor, si no servir de apoyo y completar su aprendizaje, pero estos comentarios reflejan resultados positivos respecto al seguimiento que dan los agentes a los alumnos.

## 6. Conclusiones y trabajo futuro

Como podemos observar hasta el momento, la arquitectura de agentes propuesta resulta ser bastante aceptada por los usuarios finales debido a que permite una comunicación más rápida y flexible acerca de las diferentes propuestas de itinerarios formativos dadas por el planificador inteligente.

Una de las ventajas más claras de esta arquitectura es que cada vez se hace menos necesaria la presencia del profesor para monitorizar las actividades del estudiante, ya que eso lo hacen los agentes. También que dichos agentes permiten razonar sobre la información incongruente capturada por los profesores, sirviendo así de auxiliares en el proceso de definición de las relaciones entre las diferentes actividades propuestas para su curso.

Los agentes también permiten responder al usuario con enunciados comprensibles en caso de que los planificadores no encuentren itinerarios formativos que se adapten a sus necesidades, pudiendo solicitar más información para poder crearlos.

Cabe mencionar que en todo este proceso nunca se deja de lado a los usuarios y, sobre todo, al profesor durante el proceso de adaptación de itinerarios formativos, ya que siempre se pide su consentimiento para poder mostrarlos al estudiante. Eso hace que la herramienta sea más fácilmente aceptada y que cubra las necesidades reales de los usuarios para los cuales se está desarrollando el sistema.

Sin embargo, aún quedan por implementar y conectar con las interfaces algunas de las capacidades de razonamiento inherentes al proceso de detección de inconsistencias, así como la detección en tiempo real de algunos cambios en el perfil del estudiante y su integración en el proceso de compilación de información.

Después de terminado el sistema se planea llevar a cabo una etapa de pruebas sobre un curso real, ya no en el laboratorio como las que se mencionan en este artículo.

## Referencias


1. Agarwal, R., Amrita, D., Swati, D.: Intelligent Agents in E-learning. ACM SIGSOFT Software Engineering Notes, vol. 29, no. 2, pp. 1 (2004)
2. Bouras, C., Guido, H., Vassilis, T., Thrasyvoulos, T.: Architectures Supporting e-Learning through Collaborative Virtual Environments: The Case of INVITE. In Proceedings of ICALT, pp. 13–16 (2001)
3. Brusilvosky, O., Vassileva, J.: Course Sequencing Techniques for Large-Scale Web Based Education. International Journal of Continuing Engineering Education and Lifelong Learning, vol.13, no. 1, pp. 75–94. Inder Science (2003)
4. Buraga, S.C.: Developing Agent-Oriented e-Learning Systems. In: Proceedings of the 14th International Conference on Control Systems and Computer Science, vol. 2, no. 1. Dumitrache and C. Buiu, Eds, Politehnica (2003)
5. Garrido, A., Onaindía, E., Morales, L., Castillo, L., Fernández, S., Borrajo, D.: Modeling E-Learning Activities in Automated Planning. In Proceedings of ICKEPS-ICAPS 2009, pp. 18–27 (2009)
6. Garro, A., Luigi, P.: An XML Multi-Agent System for E-Learning and Skill Management. In Agent Technologies, Infrastructures, Tools, and Applications for E-Services, pp. 283–294. Springer Berlin Heidelberg (2003)
7. Gregg, D. G.: E-learning Agents. The Learning Organization, vol. 14, no. 4, pp. 300–312 (2007)
8. Long, D., Fox, M.: PDDL 2.1: An Extension to PDDL for Expressing Temporal Planning Domains. Journal of Artificial Intelligence Research, vol. 20, no. 1, pp. 149-154. AAAI Press (2003)
9. Mencke, S., Reiner, R. D.: A Framework for Agent-Supported E-learning. In: Proceedings of the International Conference of 'Interactive computer aided learning'ICL2007. EPortofolio and Quality in e-Learning (2007)
10. Neji, M., Mohamed, B. A.: Agent-based Collaborative Affective e-Learning Framework. Electronic Journal of e-Learning, vol. 5, no. 2 (2007).
11. Nielsen, J.: Usability Engineering. Academic Press Professional, Boston (1993)
12. Pankratius, V., Olivier, S., Wolffried, S.: Retrieving Content with Agents in Web Service E-Learning Systems. In: The Symposium on Professional Practice in AI, IFIP WG12, vol. 6 (2004)
13. Sampson, D., Charalampos, K., Fabrizio, C.: An Architecture for Web-Based e-Learning Promoting re-Usable Adaptive Educational e-Content. Educational Technology & Society, vol. 5, no. 4, pp. 27v37 (2002)
14. Silveira, R. A., Rosa, M. V.: Improving Interactivity in e-Learning Systems with Multi-Agent Architecture. Adaptive Hypermedia and Adaptive Web-Based Systems, pp. 466–471, Springer Berlin Heidelberg (2002)

# Gestión del conocimiento en la micro y pequeña empresa mexicana de la industria del software

José Sergio Ruiz Castilla[1], Yulia Ledeneva[2], Héctor Cuesta Arvizu[1]

[1] Centro Universitario UAEM Texcoco
Texcoco, Estado de México, México

[2] Unidad Académica Profesional Tianguistenco,
Tianguistenco, Estado de México, México

jsergioruizc@gmail.com, yledeneva@yahoo.com, hm_cuesta@yahoo.com.mx

**Resumen.** Las empresas que desarrollan software en México son microempresas con diez o menos empleados que no cuentan con sistemas de gestión del conocimiento. Se parte de un modelo de transferencia de conocimiento de los desarrolladores expertos a los no expertos, a través de un sistema de gestión del conocimiento. Se enfocó la investigación en conocer ¿Cómo sucede la gestión de conocimiento en las micro y pequeñas empresas que desarrollan software en México? Se encontró que el problema no es solo tecnológico, sino también cultural, en dichas organizaciones. Por lo anterior se diseñó un instrumento para medir la cultura de la gestión del conocimiento, se aplicó y se muestran los resultados. Se detectó que los desarrolladores consideran muy importante compartir el conocimiento y de hecho lo hacen informalmente. Se concluyó que dichas organizaciones deben incluir la gestión del conocimiento en sus procesos de desarrollo de software.

**Palabras clave:** Conocimiento, gestión del conocimiento, base de conocimiento, activos de conocimiento, compartir conocimiento, industria del software.

## 1. Introducción

En México la industria del software se concentra en el 1.5% de empresas con 100 empleados o más, mientras que el 78.5% son MiyPEDS (Micros y Pequeñas Empresas Desarrolladoras de Software) con menos de 10 empleados [4], además de la concentración de las 2853 empresas en 5 estados de la República Mexicana; en el Distrito Federal con el 28%, Nuevo León con el 11%, Jalisco con el 8%, el Estado de México con el 6%, Puebla con el 4% y el 43% en los demás estados del país [4].
El objetivo de este trabajo de investigación es conocer la GC (Gestión del Conocimiento) en las MiyPEDS, ofreciendo resultados obtenidos a partir de la aplicación de un instrumento de medición de diversos aspectos de la GC buscando hacer una aportación para las propias MiyPEDS.

El desarrollo del conocimiento con sentido de equidad social identifica y pondera las necesidades, intereses y demandas de grupos sociales que son beneficiados por el desarrollo científico y tecnológico. Por lo anterior se proponen "redes socio-culturales de innovación" formada por expertos diversos para producir conocimiento [12].

El conocimiento en las PyMEs (Pequeñas y Medianas Empresas) desarrolladoras de software mantienen conocimiento tácito en la mente de sus trabajadores y conocimiento explícito en: sus proceso, sistemas, manuales, patentes, y otros medios; sin embargo deben valorar que la gestión del conocimiento puede tener una influencia positiva y fomentadora de los niveles de productividad y competitividad.

Los trabajadores son los poseedores del conocimiento, habilidades, destrezas y experiencias; Así lo demuestran los resultados de la influencia que tiene la gestión del conocimiento en el nivel de competitividad de las PyMEs manufactureras de Aguascalientes [5].

Los SBC (Sistemas Basados en el Conocimiento) también son un factor importante en la solución de problemas. En la industria del vestido por ejemplo es posible mejorar el diseño usando el conocimiento de expertos y un SBC para obtener resultados más óptimos en la utilización de los recursos materiales [11].

Las empresas deben aprender y capitalizar el conocimiento. Para aprender se requiere que se comparta el conocimiento que puede ser en comunidades de prácticas y de aprendizaje. Generar activos de conocimiento y compartirlos dentro de la organización. En medida que la organización aprenda promueve la cultura de la gestión del conocimiento [6].

Para lograr la adquisición del conocimiento y capital intelectual en la empresa de manufactura en México es necesario identificar cuáles son los activos intangibles de la propia empresa y de otras organizaciones que guardan similitudes para buscar y lograr la transferencia de conocimiento. Las organizaciones deben de contar con su base de conocimiento para almacenar sus mejores prácticas, procesos e innovaciones. La gestión del conocimiento concluirá con una creación de valor para la empresa [6].

Los activos tangibles son conocimientos explícitos en medios impresos o digitales; mientras que los intangibles se refieren al conocimiento tácito que está en la mente de los desarrolladores. En las MiyPEDs se carece de recursos, pero también de cultura para llevar a cabo la gestión del conocimiento. Existe la necesidad de contar con modelos y recursos para llevar a cabo gestión del conocimiento en dichas organizaciones. La gestión del conocimiento en las MiyPEDs deben tener la misión de la mejora en los procesos [1].

## 2. Gestión del conocimiento

En el trabajo de Ruiz se presentó un modelo para la gestión del conocimiento (Fig. 1). En dicho modelo se busca promover la GC en la industria del software en México [10].

El modelo de gestión de conocimiento es parte de un proyecto que busca establecer primero un modelo y una plataforma para un SGC (Sistema de Gestión del Conocimiento) así como proponer esquemas para el fomento de la GC en las MiyPEDS en México.

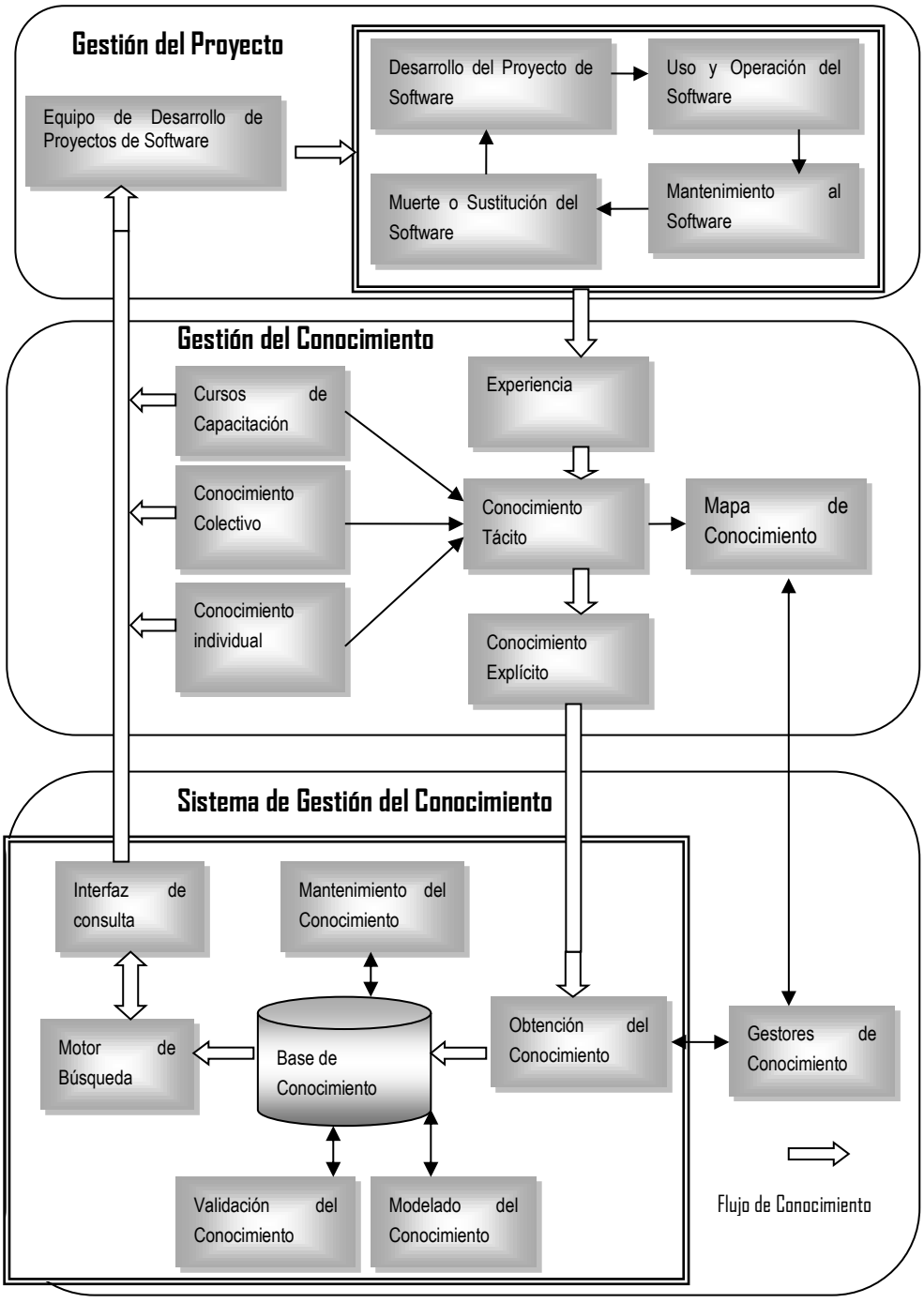



**Fig. 1.** Modelo de Gestión del conocimiento para el desarrollo de software (Ruiz et al., 2012).

El modelo cuenta con tres componentes: el primero la gestión de proyectos donde se genera experiencia al desarrollar los proyectos. El segundo la GC donde el conocimiento tácito se origina a partir de la experiencia al desarrollar proyectos de software, de los demás elementos del equipo de trabajo, de los cursos de capacitación y del conocimiento adquirido durante su formación. El conocimiento tácito se debe transformar en conocimiento explícito susceptible de almacenamiento en una base de conocimiento. El tercero es el SGC, donde se requiere que el conocimiento explícito debe obtenerse, almacenarse, validarse, mantenerse y consultarse por los usuarios no expertos. Lo anterior con el fin de reutilizar el conocimiento y mejorar los procesos de desarrollo de software. Es importante en los SGC se consulte y use el conocimiento, de lo contrario se corre el riesgo de crear un cementerio de conocimiento [10].

### 2.1. Actividad de compartir el conocimiento en las MiyPEDS en México

Las lecciones aprendidas se obtienen al culminar una tarea, actividad o un proyecto, las cuales se deben documentar para una consulta posterior por todos los integrantes de la organización [8]. En México en el modelo MoProSoft (Modelos de Procesos para la industria del software) recomienda almacenar la documentación de los procesos en una base de conocimiento de acceso libre para los integrantes de la organización [8].

Compartir el conocimiento en las organizaciones puede ser una ventaja competitiva que permite agregar valor a los productos y servicios y a la organización misma. Sin embargo para compartir el conocimiento es necesario un ambiente, herramientas y procesos definidos para dicha tarea [2].

El conocimiento se crea todos los días en la organización y crece como conocimiento tácito en la mente de cada uno de los miembros de la organización, pero es mejor si se comparte lo que se sabe con los demás.

El conocimiento se genera y se acumula a partir de lo que sé, de lo que aprendo de otros y lo que aprendo haciendo. Una vez que se posee se debe compartir con los demás logrando un conocimiento colectivo u organizacional [10].

La actividad de compartir conocimiento, debe de premiarse con estímulos económicos y no económicos que motiven a los trabajadores del conocimiento a seguir aportando conocimiento.

Compartir el conocimiento de persona a persona requiere de un ambiente adecuado, pero en organizaciones distribuidas o grandes es necesaria una estrategia para trasformar el conocimiento tácito a explícito y de explícito a tácito [7].

### 2.2. Transferencia del conocimiento en las MyPEDS en México

En México la transferencia sucede a través de cursos de capacitación, de cara a cara y a través de internet; sin embargo los desarrolladores lo hacen informalmente, porque no existen procesos y políticas que lo incluyan como actividad necesaria en el desarrollo de software.

Vale la pena conocer cómo se transfiere el conocimiento en otras organizaciones y culturas, para contrastar con la cultura de la transferencia del conocimiento en México. La transferencia del conocimiento suele darse primeramente cara a cara en

las expendedoras de agua, salas de descanso y comedores, además de espacios destinados para ello como ferias de conocimiento y foros abiertos, dónde las personas socializan. Se trata de una nueva forma de pensar, en IBM en algún momento se informó a los trabajadores *¡Dejen de conversar y pónganse a trabajar!* cuando se requiere de *¡Comiencen a conversar y pónganse a trabajar!* En el caso de los japoneses pasan muchas horas reunidos después del trabajo con el fin de socializar y transferir el conocimiento. La segunda forma es a través de documentos, repositorios de documentos, la intranet, groupware y SGC. Y la tercera forma consiste en aprender viendo, se dispone de un experto y un aprendiz que al observar cómo se desarrollan las actividades aprende [3].

Es posible trabajar con grupos de trabajo con un moderador donde se busque la socialización y la transferencia de conocimiento tácito [2]. Es importante que los empleados coincida en el mismo sitio, como comedores, áreas de café, salas de juntas y así tener contacto entre ellos y dialoguen, solo así se puede promover la transferencia del conocimiento. La socialización es el medio más eficaz para la transferencia del conocimiento tácito-tácito, por eso en Japón se acostumbra que las personas socialicen, de hecho prefieren la transferencia del conocimiento tácito [7].

### 2.3. Almacenamiento del conocimiento

Es importante obtener un inventario de activos de conocimiento, que corresponden al conjunto de conocimientos, que posee una persona o plasmado en un medio y que guarda de forma cohesiva la solución de un problema específico, dentro de un dominio del conocimiento.

Los activos de conocimiento tácito podrán existir en la mente de los desarrolladores, y se generan nuevos activos de conocimiento a medida que el desarrollador tiene nuevas experiencias u obtiene conocimientos de otros desarrolladores.

¿Dónde almacenar los activos de conocimiento? En la empresa se guardan los activos de conocimiento explícito en una base de conocimiento; dichos activos de conocimiento explícito aumentarán en medida que se introduzcan a la base de conocimiento.

Finalmente el conocimiento de la empresa debe considerar el conocimiento tácito del capital humano y los activos de conocimiento explícito de la base de conocimiento.

### 2.4. Inventario del conocimiento

El inventario de conocimiento consiste en identificar cada activo de conocimiento y registrarlo en la base de conocimiento. ¿Existe redundancia de conocimiento? Sí. Cuándo dos o más desarrolladores poseen el conocimiento. En el caso anterior solo aparecerá una vez en la base de conocimiento. En ese caso son dos poseedores, pero un solo activo de conocimiento. El sistema de almacenamiento de conocimiento no debe permitir que se dupliquen los activos de conocimiento.

Para el almacenamiento del conocimiento es necesario un repositorio de conocimiento denominado base de conocimiento donde se almacenan: archivos de texto, de audio o de video. ¿Quiénes debe administrar el conocimiento? Será

necesario destinar personas que administrarán el conocimiento que pueden ser gestores o gerentes de la GC. El conocimiento almacenado forma parte de un inventario de conocimiento. La unidad son componentes de conocimientos llamados activos de conocimiento [3, 6].

Otro nombre que se le ha dado al repositorio de conocimiento ha sido sedes del conocimiento, de igual manera tiene la misión de determinar dónde está el conocimiento, en este caso puede ser un sistema Web con acceso a los usuarios para la búsqueda y consulta del conocimiento [2].

## 3. Método utilizado

### 3.1. Modelo para la gestión del conocimiento

Tomando como base el modelo de conocimiento de la Figura 1. Permite establecer cómo se genera, transforma, almacena, administra y utiliza el conocimiento en la industria del software.

### 3.2. Cultura de la gestión del conocimiento

Se diseñó un instrumento que permitiera aplicarse a desarrolladores de MyPEDS y conocer la cultura de la gestión del conocimiento. Cuenta con 50 preguntas que el desarrollador puede contestar con una elección. Se dividió en 11 secciones que se muestran en la Tabla 1. Para medir su comportamiento, posición y opinión. Se aplicó la siguiente escala: totalmente de acuerdo, de acuerdo, ni de acuerdo ni en desacuerdo, en desacuerdo y totalmente en desacuerdo.

**Tabla 1.** Secciones del instrumento de medición de la cultura de la gestión.

| | **Secciones del instrumento** |
|---|---|
| **1** | Actividad de compartir el conocimiento |
| **2** | Con quienes se comparte el conocimiento |
| **3** | Valor del conocimiento |
| **4** | Razones por la que no se comparte el conocimiento |
| **5** | Ventajas de compartir el conocimiento |
| **6** | Pérdida de conocimiento al convertir el conocimiento tácito a explícito |
| **7** | Pago y recompensa por compartir el conocimiento |
| **8** | Disposición para consultar el conocimiento |
| **9** | Medios para la transferencia del conocimiento |
| **10** | Cultura de la gestión del conocimiento |
| **11** | Tiempo disponible para la consulta del conocimiento |

Las respuestas dependen de las políticas y prácticas de la empresa, así como de la forma de pensar y actuar de los desarrolladores. Con el instrumento se buscó conocer

si los desarrolladores llevan a cabo prácticas de gestión y compartir el conocimiento. Se agregó un apartado para saber que esperan como pago o recompensa por compartir el conocimiento. Se intentó además conocer su posición para llevar a cabo la gestión del conocimiento si la empresa lo requiere. En este caso se aplicó el instrumento en el estado de Michoacán en una empresa dedicada al desarrollo de software. La empresa tienen sus clientes principalmente del ramo financiero en México D. F. Desarrollan software a medida de las necesidades de las organizaciones, soporte técnico y hosting de bases de datos. Cuentan con un turno de 9:00 a 18:00 horas de lunes a sábado y trabajan tanto hombres como mujeres.

## 4. Resultados

Una vez aplicado el instrumento se obtuvieron datos que se procesaron y se muestran resultados en las tablas siguientes.

Lo primero que se preguntó fue la posición de los desarrolladores acerca de compartir el conocimiento con otros desarrolladores, como desarrollador de una organización o como independiente, Tabla 2.

**Tabla 2.** Actividad de compartir el conocimiento.

| Actividad de compartir el conocimiento | | | |
|---|---|---|---|
| | ¿Cuál es su posición en relación a compartir su conocimiento con otros desarrolladores? | ¿Es más factible compartir el conocimiento cuando se trabaja en una organización? | ¿Es más factible compartir el conocimiento cuando se trabaja en forma independiente? |
| Totalmente de acuerdo | **92.0%** | **46.0%** | 12.5% |
| De acuerdo | 8.0% | 42.0% | 4.2% |
| Ni de acuerdo, ni en desacuerdo | 0.0% | 12.0% | **45.8%** |
| En desacuerdo | 0.0% | 0.0% | 12.5% |
| Totalmente en desacuerdo | 0.0% | 0.0% | 25.0% |
| | 100.0% | 100.0% | 100.0% |

De acuerdo a los resultados el 100% está de acuerdo en compartir el conocimiento. El 88% Perciben más factible compartir el conocimiento dentro de una organización, mientras que las opiniones se dividen cuando trabajan de forma independiente.

Se les cuestionó con quien están de acuerdo en compartir el conocimiento, considerando sus colegas y compañeros de equipo de trabajo, Tabla 3.

**Tabla 3.** Con quien compartir el conocimiento.

| Con quien compartir el conocimiento | | | | |
|---|---|---|---|---|
| | El conocimiento no se debe compartir | Compartir su trabajo con su equipo de trabajo | Compartir conocimiento en sus empresa | Compartir conocimiento con cualquier desarrollador |
| Totalmente de acuerdo | 8.3% | 12.5% | **25.0%** | **45.5%** |
| De acuerdo | 0.0% | 0.0% | **25.0%** | 18.2% |
| Ni de acuerdo, ni en desacuerdo | 0.0% | 16.7% | 20.8% | 27.3% |
| En desacuerdo | 0.0% | 0.0% | 8.4% | 4.5% |
| Totalmente en desacuerdo | **91.7%** | **70.8%** | 20.8% | 4.5% |
| | 100.0% | 100.0% | 100.0 % | 100.0 % |

Los resultados indican que el 91.7% refieren que no están de acuerdo en no compartir el conocimiento. Mientras que el 70.8% no están de acuerdo en compartir el conocimiento solo con sus compañeros. A el 50% le interesa compartir a nivel de toda la organización y el 63.7% con cualquier otro desarrollador.

Otro cuestionamiento fue acerca del valor del conocimiento que le dan los desarrolladores para sí mismos, para su empresa, la industria y el país, Tabla 4.

**Tabla 4.** Valor del conocimiento.

| Valor del conocimiento | | | | |
|---|---|---|---|---|
| | El conocimiento tiene alto valor para las personas | El conocimiento tiene alto valor para las organizaciones | El conocimiento tiene alto valor para la industria del software | El conocimiento tiene alto valor para la economía del país |
| Totalmente de acuerdo | **91.6%** | **95.8%** | **91.3%** | **95.7%** |
| De acuerdo | 4.2% | 0.0% | 8.7% | 4.3% |
| Ni de acuerdo, ni en desacuerdo | 4.2% | 4.2% | 0.0% | 0.0% |
| En desacuerdo | 0.0% | 0.0% | 0.0% | 0.0% |
| Totalmente en desacuerdo | 0.0% | 0.0% | 0.0% | 0.0% |
| | 100.0% | 100.0% | 100.0% | 100.0% |

El 95.8% de los desarrolladores están de acuerdo en que el conocimiento tiene un alto valor para sí mismos y su empresa y el 100% está de acuerdo en que tiene un alto valor para la industria del software y para el país. Se les preguntó también por las

razones por las que no se comparte el conocimiento y de la propiedad del conocimiento Tabla 5.

**Tabla 5.** Razones por las que no se comparte el conocimiento.

| Razones por las que no se comparte el conocimiento | | | | |
|---|---|---|---|---|
| | Porque es personal | Porque es de la organización | Porque es estratégico para la organización | Porque es propiedad del desarrollador |
| Totalmente de acuerdo | 0.0% | 21.7% | **50.0%** | 4.5% |
| De acuerdo | 0.0% | 17.4% | **16.7%** | 9.1% |
| Ni de acuerdo, ni en desacuerdo | 12.5% | 17.4% | 8.3% | 9.1% |
| En desacuerdo | 12.5% | 8.7% | 12.5% | **45.5%** |
| Totalmente en desacuerdo | **75.0%** | **34.8%** | 12.5% | **31.8%** |
| | 100.0% | 100.0% | 100.0% | 100.0% |

El 87.5% de los desarrolladores cree que el conocimiento es personal y el 43% considera que es de la organización, con opiniones muy divididas. Por lo tanto coinciden que el conocimiento es de la organización aun cuando lo poseen los trabajadores.

Se incluyó un apartado para conocer qué ventajas avizoran al compartir el conocimiento al dar y recibir conocimiento Tabla 6.

**Tabla 6.** Ventajas de compartir el conocimiento.

| Ventajas de compartir el conocimiento | | | | |
|---|---|---|---|---|
| | En posible reutilizar el conocimiento | Que otros desarrolladores conozcan y usen su conocimiento | En aprender de otros a | Permite productividad en las organizaciones |
| Totalmente de acuerdo | **91.7%** | **74.0%** | **79.2 %** | **83.3%** |
| De acuerdo | 8.3% | 13.0% | 20.8% | 16.7% |
| Ni de acuerdo, ni en desacuerdo | 0.0% | 13.0% | 0.0% | 0.0% |
| En desacuerdo | 0.0% | 0.0% | 0.0% | 0.0% |
| Totalmente en desacuerdo | 0.0% | 0.0% | 0.0% | 0.0% |
| | 100% | 100% | 100% | 100% |

Todos están convencidos que el conocimiento se puede reutilizar. El 87% está dispuesto a compartir el conocimiento hacia otros. Todos están de acuerdo en que

pueden aprender de otros y que existe mayor productividad en la organización al compartir el conocimiento.

Se preguntó acerca de la pérdida de conocimiento que existe cuando exteriorizan el conocimiento tácito con el fin de convertirlo en explícito en algún medio, Tabla7.

**Tabla 7.** Pérdida de conocimiento al convertirlo de tácito a explícito.

| Pérdida de conocimiento al convertirlo de tácito a explícito | | | | |
|---|---|---|---|---|
| | No sabe cómo documentar su conocimiento | No desea revelar los detalles del conocimiento | Existen aspectos imposibles de explicar | No sabe cómo exponer sus experiencias |
| Totalmente de acuerdo | 4.1% | 0.0% | 12.5% | 0.0% |
| De acuerdo | 16.7% | 4.2% | 20.8% | **29.2%** |
| Ni de acuerdo, ni en desacuerdo | **41.7%** | 16.6% | 25.0% | 20.8% |
| En desacuerdo | 12.5% | 12.5% | 8.4% | **29.2%** |
| Totalmente en desacuerdo | 25.0% | **66.7%** | **33.3%** | 20.8% |
| | 100.0% | 100.0% | 100.0% | 100.0% |

No están de acuerdo y desconocen el proceso de convertir el conocimiento, se asume que lo hacen, pero no tienen claro cómo. Mientras que el 79.2% no desean hacerlo.

Se cuestionó el pago por el conocimiento para conocer si están de acuerdo en donar o vender su conocimiento tácito y explícito, Tabla 8.

**Tabla 8.** Pago y recompensas por el conocimiento.

| Pago y recompensas por el conocimiento | | | | |
|---|---|---|---|---|
| | ¿Compartiría su conocimiento sin pago y reconocimientos? | ¿Compartiría su conocimiento por algún reconocimiento no monetario? | ¿Compartiría su conocimiento por un bono? | ¿Compartiría su conocimiento por un pago en efectivo? |
| Totalmente de acuerdo | **50.0%** | **62.5%** | **50.0%** | **37.5%** |
| De acuerdo | 29.1% | 12.5% | 12.5% | 8.3% |
| Ni de acuerdo, ni en desacuerdo | 12.5% | 16.6% | 16.6% | 29.2% |
| En desacuerdo | 4.2% | 4.2% | 4.2% | 4.2% |
| Totalmente en desacuerdo | 4.2% | 4.2% | 16.7% | 20.8% |
| | 100.0% | 100.0% | 100.0% | 100.0% |

Los resultados resultaron interesantes porque existe la voluntad de donar el conocimiento por parte del 79.1% pero el 75% esperan reconocimientos y el 62.5% un bono y solo el 45.8% esperan un pago en efectivo.

También se incluyeron preguntas de la disposición que tienen para consultar el conocimiento, si existiere en la organización, Tabla 9.

**Tabla 9.** Disposición para consultar el conocimiento.

| Disposición para consultar el conocimiento | | | | |
|---|---|---|---|---|
| | Le interesa consultar el conocimiento de otros | Consultaría conocimiento, solo si es política de la organización | Mientras desarrolla, si consulta conocimiento | Siempre consulta conocimiento |
| Totalmente de acuerdo | **81.8%** | 8.7% | **76.2%** | **43.5%** |
| De acuerdo | 13.6% | 4.4% | 19.0% | 13.0% |
| Ni de acuerdo, ni en desacuerdo | 4.6% | 21.7% | 4.8% | 34.8% |
| En desacuerdo | 0.0% | 21.7% | 0.0% | 8.7% |
| Totalmente en desacuerdo | 0.0% | **43.5%** | 0.0% | 0.0% |
| | 100.0% | 100.0% | 100.0% | 100.0% |

Se encontró que el 95.4% les interesa el conocimiento de otros. El 65.2% opina que no es necesaria una política de la organización. Es importante notar que el 95.2% consulta conocimiento, pero otro 56.5% siempre lo hace.

Se incluyeron preguntas para conocer los medios conde almacenan y comparten el conocimiento, Tabla 10.

**Tabla 10.** Medios para la transferencia del conocimiento.

| Medios para la transferencia del conocimiento | | | | |
|---|---|---|---|---|
| | Prefiere cursos para obtener nuevos conocimientos | Prefiere manuales para obtener nuevos conocimientos | Prefiere videos para obtener nuevos conocimientos | Prefiere un sistema de gestión del conocimiento |
| Totalmente de acuerdo | **37.5%** | 25.0% | **33.3%** | 34.8% |
| De acuerdo | 33.3% | **33.3%** | 29.2% | **52.2%** |
| Ni de acuerdo, ni en desacuerdo | 20.8% | 29.2% | 37.5% | 8.7% |
| En desacuerdo | 4.2% | 8.3% | 0.0% | 4.3% |
| Totalmente en desacuerdo | 4.2% | 4.2% | 0.0% | 0.0% |
| | 100.0% | 100.0% | 100.0% | 100.0% |

De acuerdo las opiniones aceptan los diferentes formatos mencionados para la transferencia del conocimiento. El 70.8% prefieren cursos, el 58.3% manuales, 62.5% prefieren videos y el 87% prefieren un SGC.

Se incluyeron las preguntas más importantes para conocer las actividades de la gestión del conocimiento para conocer la cultura de la GC, Tabla 11.

**Tabla 11.** Cultura de la gestión del conocimiento.

| Cultura de la Gestión del conocimiento | | | | |
|---|---|---|---|---|
| | ¿Genera un repositorio de su conocimiento? | ¿Consulta su repositorio de conocimiento? | ¿Comparte el conocimiento de su repositorio? | ¿Consulta el conocimiento de otros desarrolladores? |
| Totalmente de acuerdo | 17.4% | **34.8%** | **56.5%** | **68.2%** |
| De acuerdo | **34.8%** | 26.1% | 21.8% | 22.8% |
| Ni de acuerdo, ni en desacuerdo | 30.4% | 13.0% | 13.0% | 4.5% |
| En desacuerdo | 13.0% | 17.4% | 8.7% | 4.5% |
| Totalmente en desacuerdo | 4.4% | 8.7% | 0.0% | 0.0% |
| | 100.0% | 100.0% | 100.0% | 100.0% |

Se encontró que el 52.2% tiene una repositorio de conocimiento, que el 60.9% lo consulta, el 78.3% comparte su repositorio y el 91% consulta el conocimiento de otros.

Se incluyeron preguntas para conocer el tiempo que disponen y estarían dispuestos a dedicar a la gestión del conocimiento, Tabla 12.

**Tabla 12.** Tiempo disponibles para la gestión del conocimiento.

| Tiempo disponible para la gestión del conocimiento | | | | |
|---|---|---|---|---|
| | ¿Considera que no tiene tiempo para gestionar el conocimiento? | ¿Dispondría de hasta 5 horas a la semana? | ¿Dispondría de hasta 10 horas a la semana? | ¿Dispondría de más de 10 horas a la semana? |
| Totalmente de acuerdo | 8.8% | **34.8%** | 8.7% | 8.7% |
| De acuerdo | 13.0% | 26.1% | **30.4%** | 21.7% |
| Ni de acuerdo, ni en desacuerdo | 21.7% | 21.7% | 17.5% | 21.7% |
| En desacuerdo | 21.7% | 8.7% | 21.7% | **26.2%** |
| Totalmente en desacuerdo | **34.8%** | 8.7% | 21.7% | 21.7% |
| | 100.0% | 100.0% | 100.0% | 100.0% |

Se rescató que el 56.5% está en de acuerdo con que no tiene tiempo, el 60.9% dedicaría hasta 5 horas a la semana, el 43.4% está en desacuerdo en invertir hasta 10 horas y el 47.9% está en de acuerdo en invertir más de 10 horas en la gestión del conocimiento.

## 5. Conclusiones y trabajos futuros

Los desarrolladores están de acuerdo en compartir el conocimiento dentro de su organización y ámbito. Prefieren compartir el conocimiento con sus compañeros y luego hacia el exterior de la organización. Consideran que el conocimiento tiene un alto valor para sí mismos, su organización la industria y el país. Consideran que el conocimiento es personal, pero que le pertenece a la organización. Desconocen y les resulta compleja la conversión del conocimiento tácito a explícito. Están convencidos de que el conocimiento se puede reutilizar y que aprender de otros permite mayor productividad a la organización. Que están dispuestos a compartir el conocimiento algunos sin pago, pero en otros esperan reconocimientos, bonos y pagos. Que les interesa y consultan el conocimiento de otros aun cuando no haya una política de la organización que así lo determine. Para la transferencia del conocimiento prefieren un SGC y en segundo lugar cursos, manuales y videos. Que cuentan con sus propios repositorios de conocimiento y que los comparten. Que están dispuestas a dedicar hasta cinco horas para la GC.

Se emite una propuesta de recomendaciones para las MiyPEDS para considerar práctica de la GC y la necesidad de un SGC.

- Incluir procesos de GC en su empresa.
- Sensibilizar a los desarrolladores para fomentar la GC.
- Capacitar a los desarrolladores para manejar el conocimiento con seguridad.
- Definir un método y técnicas para la conversión de conocimiento tácito a explícito.
- Contar con un SGC para captar, almacenar y compartir el conocimiento.
- Crear incentivos monetarios y no monetarios para fomentar la GC.
- Medir el impacto de la GC una vez implementada.

Esta investigación se desprende de otra, de la cual se ha creado un modelo que ya se ha publicado; también se está trabajando en una plataforma de un SGC para implementarse en MiyPEDS con el fin de probar el modelo, hacer los ajustes necesarios para proponer se valore su uso en las organizaciones que desarrollan software en México.

De acuerdo a los resultados los trabajadores del conocimiento de las MiyPEDS practican y muestran disposición para llevar a cabo la Gestión del Conocimiento. Por lo anterior es más importante sensibilizar de la necesidad de considerar esta actividad como importante e incorporarla en sus procesos de dichas organizaciones. Ante tal escenario existen interrogantes como: ¿Cuál es la prospectiva de los directivos de las MiyPEDS acerca de la Gestión del Conocimiento? ¿Modificar políticas fomentará la Gestión del conocimiento? ¿Incluir la gestión del conocimiento como un proceso más hará más productivas a las MiyPEDS? ¿Se logrará cambiar la cultura de la Gestión del Conocimiento en las MiyPEDS? Para encontrar estas incógnitas se continuará con esta investigación.

Finalmente se pretenden integrar técnicas de inteligencia artificial en el SGC con el fin de hacerlo más efectivo.

## Referencias


1. Capote, J., Llanten, J., Pardo, C., González, A., Collazos, C.: Gestión del conocimiento como apoyo para la mejora de procesos software en las micro, pequeñas y medianas empresas. Revista ingeniería e investigación, 28(1), 137–145 (2008)
2. Collison, C., Parcell, G.: La gestión del conocimiento, Lecciones prácticas de una empresa líder. Buenos Aires: Paidós (2003)
3. Davenport, T., Prusak, L.: Conocimiento en acción, Como las organizaciones manejan lo que saben. Buenos Aires: Prentice Hall (2001)
4. INEGI. Instituto Nacional de Geografia y Estadística. Recuperado el 10 de 06 de 2013, de http://www3.inegi.org.mx/sistemas/mapa/denue/default.aspx
5. Maldonado, G., Martínez, M. D., García, R.: La influencia de la gestión del conocimiento en el nivel de competitividad de las PyMEs manufactureras de Aguascalientes. Investigación y Ciencia (55), 24–31 (2012)
6. Martínez, Martínez, A., Corrales, M.: Administración de conocimiento y desarrollo basado en conocimiento, redes e innovación. México D. F.: CENGAGE Learning (2010)
7. Nonaka, S., Takeuchi, N.: La organización creadora del conocimiento. México: Oxford (1999)
8. Normalización y Certificación Electrónica A. C. Guía práctica de implantación de los requisitos de la NMX-I-059-NYCE-2005 (MoPRoSoft). México D. F.: Normalización y Certificación Electrónica, A. C. (2007)
9. NYCE A.C. Guía práctica de implantación de los requisitos de la NMX-I-059-NYCE-2005 (MoProSoft) (Primera ed.). México D. F.: Normalización y Certificación A. C. (2007)
10. Ruggles, R., Holtshouse, D.: La ventaja del conocimiento. México: CECSA (1999)
11. Ruiz, S., Ledeneva, Y., Morales, R.: Base de conocimiento de los procesos de desarrollo de software a través de un modelo de un sistema de gestión del conocimiento. Research in Computing Science, 55, 113–123 (2012)
12. Santiago, K., Laureano, A. L., Sánchez, J., Sarmiento, E., Domínguez, O.: Sistema basado en conocimiento para la industria del vestido en México. Research in computing science, 55, 151–162 (2012)
13. Valhondo, D.: Gestión del conocimiento: del mito a la realidad. Madrid: Ediciones Díaz de Santos (2003)
14. Velasco, A.: ¿Cómo pueden las ciencias, las técnicas, las artes y las humanidades contribuir a la democracia, la libertad y la equidad? Ciencia, Revista de la Academia Mexicana de Ciencias, 64(1), 10–17 (2013)

# Control predictivo usando un modelo difuso para la tasa de crecimiento bacteriano

Marco A. Márquez-Vera[1], Abel García Barrientos[1], Julio C. Ramos-Fernández[1], Carlos A. Márquez Vera[2], Ubaldo Baños-Rodríguez[1]

[1] Depto. de mecatrónica, Universidad Politécnica de Pachuca, Zempoala, Hgo., México

[2] Depto. de Química, Universidad Veracruzana, Poza Rica, Ver., México

{marquez, abel, jramos}@springer.com, ubaroz.79@gmail.com, carmarquez@uv.mx

**Resumen.** Existen procesos difíciles de controlar como lo son los procesos químicos, en el presente trabajo se va a tratar con el crecimiento bacteriano en un proceso biotecnológico, para llevarlo a cabo el control de la tasa de crecimiento se ha propuesto usar un modelo difuso. Fue utilizada la técnica de *clustering* para las funciones de pertenencia del antecedente y emplear mínimos cuadrados para obtener los consecuentes de las reglas. En la práctica no siempre se puede garantizar que el actuador empleado en el control podrá aplicar la señal de control propuesta dada una saturación o por su respuesta a la frecuencia, de manera que se propuso emplear un control predictivo para anticipar la señal de control. Una tabla comparativa muestra los resultados obtenidos con diferentes horizontes de predicción. Finalmente, el uso de un modelo de referencia puede reducir el error a la referencia mediante algún criterio de error.

**Palabras clave:** Lógica difusa, control predictivo, inversión de modelo, tasa de crecimiento.

## 1. Introducción

En los procesos químicos, usualmente es difícil diseñar una ley de control [1], además, en este caso, trabajar con microorganismos presenta una diversidad de problemas a afrontar, como lo son el tratar con un sistema no lineal, tener dinámicas pobremente conocidas, incertidumbres en el modelo, variación en el tiempo y parámetros desconocidos [2]. Por otro lado, la reacción química puede ser modelada como una perturbación no estacionaria.

La idea principal en este trabajo es usar la lógica difusa para mejorar el control de la tasa de crecimiento bacteriana. La lógica difusa fue usada para obtener un modelo difuso [3] y mediante la inversión de este modelo obtener la señal de control a aplicar en el proceso. Este método es una alternativa cuando se tiene vaguedad en la dinámica del proceso [4]; si el modelo difuso es aproximado al sistema real, podemos decir que

la inversión del modelo puede cancelar, en cierta manera, las no linealidades del proceso.

Un sistema difuso es básicamente un sistema basado en reglas, donde el conocimiento de un experto o de un ingeniero puede ser usado para construir la base de reglas [5]. Dado que el proceso es variante en el tiempo y se carece de un conocimiento profundo en procesos químicos, no resulta sencillo determinar las funciones de pertenencia ni los consecuentes de las reglas, de manera que un sistema adaptable debería ser implementado para aproximar al sistema. Debido a esto, un sistema tipo Sugeno [6] puede ser sintonizado gracias a su sencillez y a que sus consecuentes pueden ser submodelos lineales [7,8]

Sin embargo, existen algunas desventajas en el diseño de los parámetros, en [9] se menciona que:

- Existen muchos parámetros como el número de funciones de pertenencia, el método de agregación de las reglas o la T-norma que se va a utilizar, que pueden ser asignados de forma arbitraria.
- La base de reglas puede no tener los parámetros óptimos, o bien, ellos deben ser variantes en el tiempo.

El artículo está organizado de la siguiente manera, primero se presenta una descripción del proceso a controlar en la Sección 2, donde también se da una breve introducción a la lógica difusa y a los sistemas de inferencia en la subsección 2.1; la subsección 2.2 muestra al control predictivo y el control de la tasa de crecimiento en el proceso. Los resultados están en la Sección 3 donde es posible ver una comparación entre diferentes criterios de error a manera de discusión empleando diferentes horizontes de control. Finalmente, las conclusiones se presentan en la Sección 4.

## 2. Definición del problema

El proceso a controlar en un bioreactor, donde se emplean microorganismos para procesar fenol. El fenol es un contaminante común en el agua, algunos procesos textiles y tratamientos para la madera emplean esta sustancia [10]. En los procesos biotecnológicos, el contaminante es usado como alimento para las bacterias, y de esta forma se purifica el agua, los lodos activados son comúnmente usados para tratar las aguas residuales [8]. El modelo matemático del proceso fue propuesto y validado en [10].

Existen dos modelos generales para el crecimiento bacteriano, son los modelos de Haldane y Monod [1], dichos modelos se muestran en (1) y (2) respectivamente, y en forma gráfica en la Fig. 1. donde $x(t)$ es la concentración de sustrato, y su tasa de crecimiento es escalada en el modelo del proceso para encontrar las tasas de consumo o producción de determinadas sustancias [11].

$$\mu_1(x(t)) = \frac{\mu_{\max} x(t)}{K + x(t)}, \quad (1)$$

$$\mu_2(x(t)) = \frac{\mu_{\max} x(t)}{K_1 + x(t) + K_2 x^2(t)}, \quad (2)$$

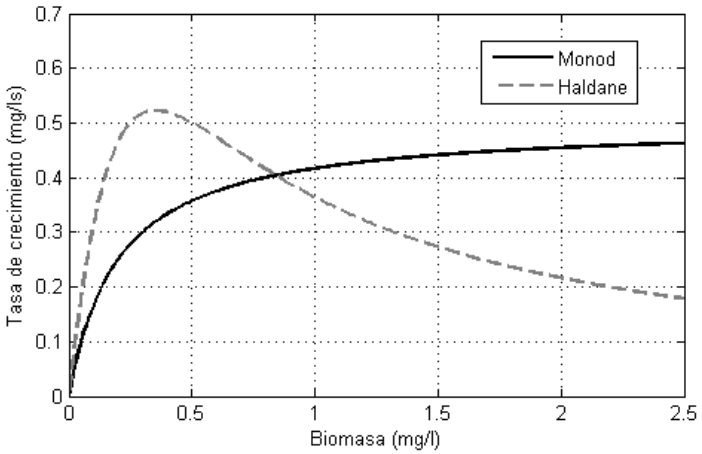



**Fig. 1.** Modelos de las tasas de crecimiento microbiano.

En el proceso, el fenol es tóxico para las bacterias [10], de manera que se utilizó el modelo de Haldane. Para validar el control, las constantes del modelo utilizado se muestran en la Tabla 1, estos valores pueden ser usados para simulación, dado que el ciclo de vida de las bacterias cambia, el modelo es variante en el tiempo, por lo que se debería utilizar un modelo adaptable.

**Tabla 1.** Parámetros del modelo de crecimiento.

| Símbolo | Valor | Unidades |
|---|---|---|
| $\mu_{\max}$ | 0.4 | 1/min |
| $K_1$ | 2 | mgl-1 |
| $K_2$ | 16 | mgl-1 |

### 2.1. Sistema de inferencia difuso

La lógica difusa emergió como una alternativa para trabajar con información imprecisa o incorrecta, así como para tomar decisiones en proceso con incertidumbres no aleatorias, tal como sucede con en el lenguaje natural con el uso de términos ambiguos [3]. La noción sobre un conjunto difuso es una buena referencia para construir sistemas de inferencia, pero de una manera más general que usando conjuntos clásicos [6]. La idea es trabajar con variables lingüísticas más sencillas de interpretar.

Un conjunto difuso es aquel en donde los elementos que contiene tienen diferentes grados de pertenencia a dicho conjunto, los valores de pertenencia de los elementos son obtenidos mediante una función de pertenencia, existen diversas formas para dibujar una función de pertenencia, la condición es que se genere un conjunto convexo [6], de forma análoga a los conjuntos clásicos, se pueden usar operadores basados en la norma triangular para realizar la intersección entre conjuntos difusos [3]. Con los conjuntos difusos es posible generar proposiciones lógicas para conceptos que no son claros.

En este trabajo, se uso el sistema de inferencia tipo Sugeno gracias a que sus consecuentes pueden ser constantes, y a su vez, obtenidos por algún método para aproximar un modelo matemático complejo o no bien conocido [12]. Las reglas difusas pueden representarse como (3)

$$R^i : \text{Si } x^1 \text{ es } A^1 \text{ y} \cdots \text{y } x^n \text{ es } A^n, \text{entonces } y^i = a_0 + a_1 + \cdots + a_n, \quad (3)$$

donde $R^i$ representa la i-ésima regla, $x$ es el vector de entrada para encontrar sus valores de pertenencia, $A$ es un conjunto difuso y $y^i$ es la salida de la regla, la cual puede ser una combinación lineal de las entradas. La base de reglas es una síntesis de todas las reglas del sistema de inferencia y también es llamada memoria asociativa difusa [3].

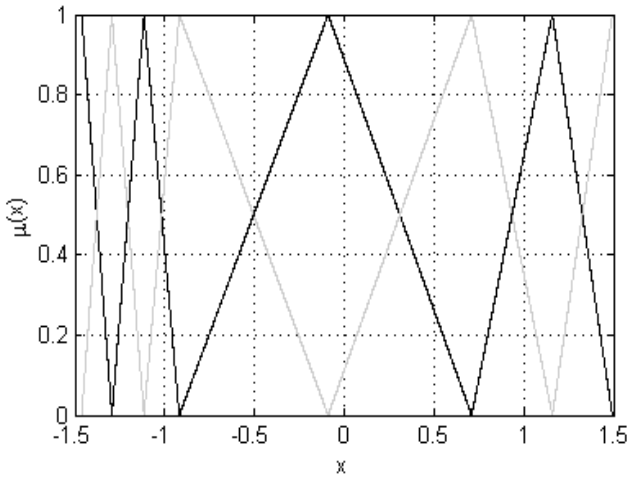



**Fig. 2.** Funciones de pertenencia.

Se utilizaron mínimos cuadrados [6] para obtener los consecuentes de las reglas, en especial, para los antecedentes se utilizó el algoritmo de Gustafson-Kessel (G-K) el cual no emplea la norma euclidiana, sino la distancia de Mahalanobis [13], este algoritmo de cluster es el que encuentra la partición difusa del universo de discurso $x$, la cual se muestra en la Fig. 2, donde ocho centros fueron usados para la partición, encontrándose las ocho funciones de pertenencia, las cuales son ordenadas y normalizadas para encontrar la forma presentada.

Los mínimos cuadrados de definen como en (4) donde las variables medidas son usadas para sintonizar los consecuentes en el sistema de inferencia, para lograrlo, se emplean los valores de pertenencia para encontrar la matriz $\Phi$.

$$\Phi = \left(M_1 X_e, M_2 X_e, \cdots, M_n X_e\right),$$

donde $X_e$, es la matriz del regresor extendida de las mediciones $X$, es decir, $X_e = \left(X \mid 1\right)$[14], y $M_i$ es una matriz diagonal que contiene los valores de pertenencia obtenidos mediante las funciones de pertenecía. Esta matriz $\Phi$ es usada para obtener los consecuentes de las reglas difusas como se muestra a continuación

$$\theta = \left(\Phi^T \Phi\right)^{-1} \Phi^T y, \tag{4}$$

donde $y$ es la salida del proceso, los valores de pertenencia usados para llenar las matrices $M_i$ están normalizados de la forma mostrada en (5)

$$\mu_i'(x_j(t)) = \frac{\mu_i(x_j(t))}{\sum_{i=1}^{n} \mu_i(x_j(t))}, \tag{5}$$

aquí, $i$ denota la i-ésima regla difusa, $j$ determina la j-ésima medición y $\mu'$ es la pertenencia normalizada de $\mu$. De esta forma la salida del sistema de inferencia difuso es $y_{dif} = \Phi\theta$.

En la Fig. 3 se muestra una comparación entre usar solo mínimos cuadrados para aproximar un sistema sin tener la mejor partición difusa, y usando el algoritmo G-K que arrojó las funciones mostradas en la Fig. 2. En [13] un modelo difuso fue usado para controlar la neutralización del pH en un proceso.

Una vez sintonizado un modelo difuso, es necesario invertir el modelo, el control de la tasa de crecimiento se realiza modificando el flujo de entrada en el bioreactor, el control de la tasa de crecimiento se muestra en la Fig. 4, para tener una idea del desempeño este tipo de control se uso la integral del valor absoluto del error (IAE) (6), este criterio fue usado dado que si el error es menor a uno, usar otro criterio como la integral del error cuadrático sería muy pequeño (7). La IAE encontrada fue de 4.9562. Para reducir este criterio de error, es posible utilizar un modelo de referencia, el cual da una referencia suave (clase $C_\infty$) suavizando también la señal de control aplicada al actuador [14], dicho filtro en tiempo continuo usado como modelo de referencia está determinado por (8) usando la transformada de Laplace [14].

$$\mathrm{IAE} = \int_0^T \left|e(t)\right| dt, \tag{6}$$

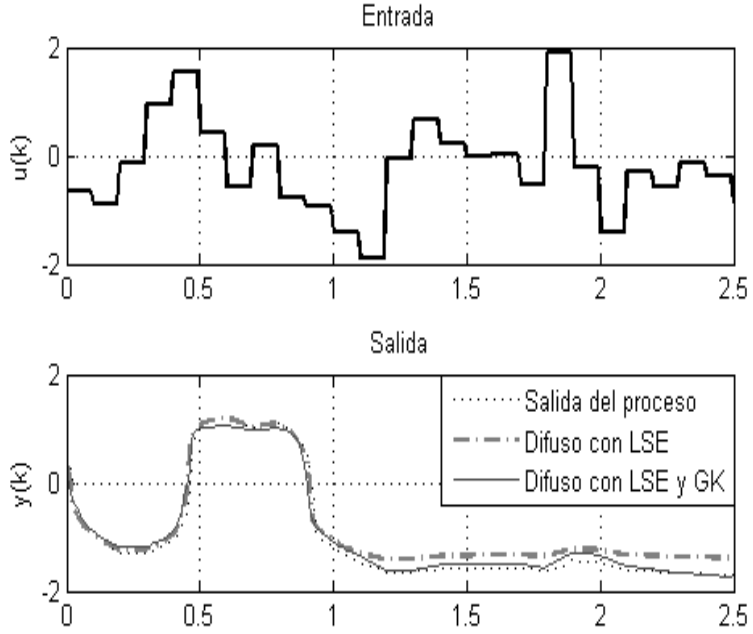



**Fig. 3.** Validación de los modelos difusos.

$$\mathrm{ISE} = \int_0^T e^2(t)dt\,, \tag{7}$$

$$\mathrm{Mod}_{\mathrm{ref}}(s) = \frac{3.5}{s+3.5}\,. \tag{8}$$

Si la salida del sistema de inferencia es calculada usando (9), realizando el cálculo de la salida en tiempo discreto

$$y(k+1) = \sum_{i=1}^{n} \mu_i{}'\big(y(k),\ldots,u(k\text{-}n_u+1)\big) \left(\sum_{j=1}^{n_y} a_{ij}\, y(k-j+1) \sum_{j=1}^{n_u} b_{ij}\, u(k-j+1)\right), \tag{9}$$

donde $a_{ij}$ y $b_{ij}$ se encuentran dentro del vector de consecuentes $\theta$ en la forma $\theta = \left(\left(a_1^T, b_1\right), \left(a_2^T, b_2\right), \cdots \left(a_n^T, b_n\right)\right)$. Con esto, la señal de control se calcula usando (10):

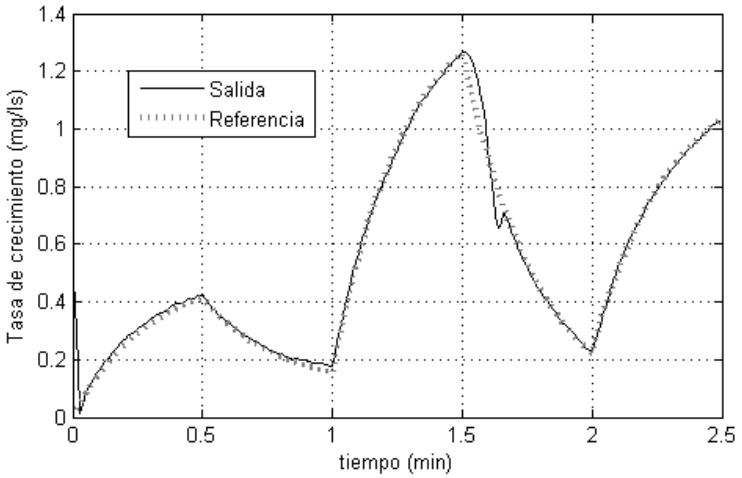



**Fig. 4.** Control por inversión del modelo difuso.

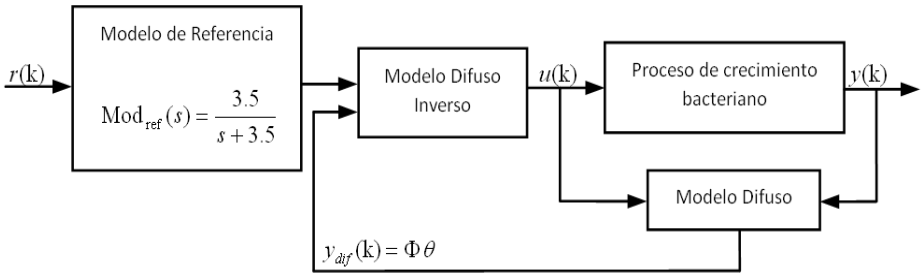



**Fig. 5.** Diagrama de control usando el modelo inverso.

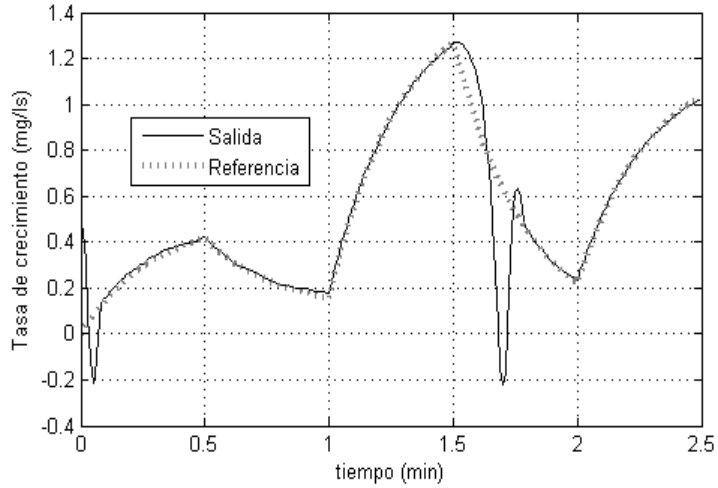



**Fig. 6.** Simulación del control con una restricción en la señal de control.

$$\mathrm{u(k)} = \frac{r(k+1) - y(k)}{\sum_{i=1}^{n} \mu_i'(y(k),\ldots,u(k-n_u+1)b_{ij})}, \tag{10}$$

El diagrama de control se presenta en la Fig. 5, aquí el modelo difuso aproximado se usa para cancelar las dinámicas del proceso y es usado para generar la señal de control.

En la práctica, es probable que se tenga una restricción física en los actuadores, de manera que le limite el poder responder a la señal de control aplicada, en este caso se limitó el flujo de entrada a l proceso a ±20 mg/ls, los resultados de tener esta restricción en el sistema de control se presentan en la Fig. 6, la IAE obtenida para este caso fue de 17.1986 incluso usando el modelo de referencia.

### 2.2. Control predictivo

Podemos comparar la teoría de control clásica a ir manejando un vehículo viendo hacia abajo, la señal de control cambia hasta que se presenta alguna situación; usando un control predictivo es similar a conducir viendo al frente, y cuando se vislumbra alguna situación al frente, se cambia la señal de control con anticipación. En control predictivo se usan dos horizontes conocidos como el horizonte de control y el de predicción [15], si no se emplean corrimientos en los horizontes, la señal de control sería como la usada por inversión del modelo y se ha demostrado que usar horizontes muy grandes empeora el resultado incrementando el error de salida [13]. En el presente trabajo los horizontes son de dos muestras en el futuro, es decir, se usa la señal de referencia que se va a aplicar dentro de dos periodos de muestreo para calcular la señal de control presente.

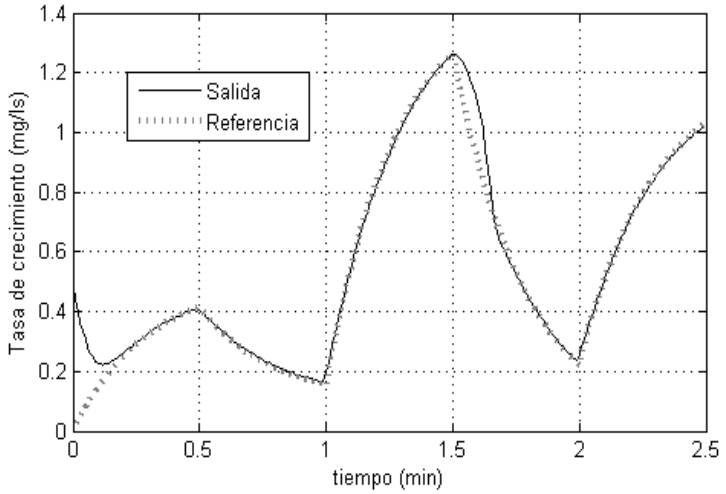



**Fig. 7.** Simulación del control predictivo con restricción en la señal de control.

El horizonte de predicción $H_p$ es construido con la salida aproximada del modelo difuso del proceso $y_{dif}(k+i)$ para $i=1,2,\ldots H_p$ y las señales de control futuras se determinan hasta el horizonte de control $H_c$, tal que se obtiene $u(k+i)$ usando el modelo y la referencia para calcular la señal de control para $i=1,2,\ldots H_c-1$ y siempre el horizonte de control debe ser menor o igual al de predicción, i.e. $H_c \leq H_p$ [13]. En la Fig. 7 se muestra el control de la tasa de crecimiento usando el modelo difuso para calcual un control predictive, teniendo a su vez la restricción en la señal de control de ±20 mg/ls, para este caso, la IAE encontrada fue 6.5240.

## 3. Resultados y discusión

El uso de la lógica difusa hizo posible controlar la tasa de crecimiento microbiano al interior de un bioreator, existen alguna otras técnicas lineales para hacer esto, pero en este caso, la idea fue utilizar un modelo difuso asumiendo que se desconoce el modelo exacto del proceso, y que este a su vez, puede ser variante en el tiempo, el error encontrado fue aceptable para este tipo de sistemas, y gracias a la velocidad de reacción de estos procesos, es posible calcular los valores aproximados que tendrán algunas señales en el futuro usando el modelo difuso encontrado. Dado que en la práctica, no siempre es posible aplicar la señal de control calculada debido a limitaciones en los actuadores el control predictivo mostró ser una buena estrategia de control reduciendo el error encontrado al tener una restricción de saturación en la señal de control.

En la Tabla 2 se presentan los criterios de error encontrados para el control por inversión de modelo, para ese mismo control con restricción en la señal de control y para cuando se aplica control predictivo al tener dicha restricción.

**Tabla 2.** Errores de seguimiento.

| Acción de control | IAE | ISE |
|---|---|---|
| Control por inversión de modelo | 0.6317 | 4.9562 |
| Control con restricción en $u(k)$ | 10.7533 | 17.1986 |
| Control predictivo con restricción | 1.2580 | 6.5240 |

Ahora, la situación podría ser, cómo determinar los horizontes usados en el control predictivo, un método es la optimización branch and bound [13] en donde se calculan cotas para los valores candidatos a tener la mejor opción, los límites fueron de cero muestras de predicción en los horizontes, a cinco muestras dadas por el modelo difuso; en este caso se encontró que $H_c = H_p = 2$, en la Fig. 8 es posible ver las simulaciones del control usando diferentes horizontes y la Tabla 3 muestra los errores obtenidos para algunos casos usando la IAE y la ISE.

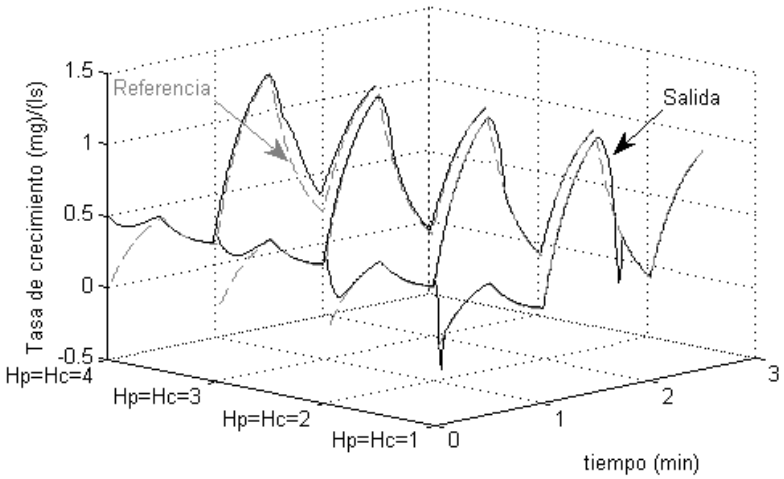



**Fig. 8.** Control predictivo con diferentes horizontes.

**Tabla 3.** Comparación con diferentes horizontes de control.

| Horizonte de control $H_c$ | IAE | ISE |
|---|---|---|
| $H_c = 1$ | 7.4740 | 1.6153 |
| $H_c = 2$ | 6.5240 | 1.2580 |
| $H_c = 3$ | 11.2908 | 2.0400 |
| $H_c = 4$ | 20.7562 | 3.8680 |

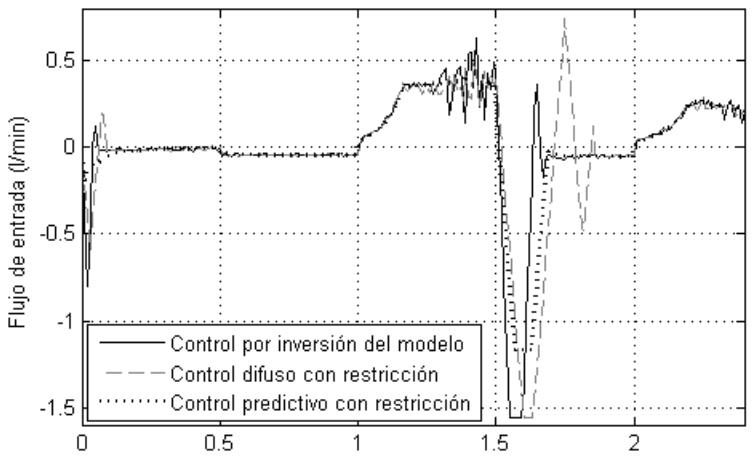



**Fig. 9.** Señales de control.

Dada la restricción en la señal de control, la salida del sistema puede no seguir exactamente la referencia establecida, en la Fig. 9 se muestran las señales de control cuando el sistema tiene una restricción en su amplitud de ±40 mg/ls, la segunda gráfica en con la restricción de ±20 mg/ls, la restricción en el control puede verse como un retardo en la señal, el control predictivo, por su parte, se anticipa a los cambios, reduciendo este aparente retardo y reduciendo el error [16]. Finalmente, la señal de control se obtiene a partir del modelo difuso y no sólo de las mediciones del sistema, las cuales pueden presentar ruido.

Los resultados obtenidos pueden ser comparados con los obtenidos en [15] donde también se trabajo con un reactor piloto y en donde el proceso consiste en controlar una reacción exotérmica con un calentador como actuador, un análisis similar sobre la región de atracción para el control predictivo puede ser aplicado a este trabajo, la diferencia principal sería que el modelo en este caso es basado en lógica difusa y que se presentó una comparación entre diferentes horizontes de control. En la Tabla 3 no se presentó el caso con $H_c = 0$, el cual sería el mismo mencionado al inicio en la Fig. 6.

## 4. Conclusiones

La lógica difusa fue capaz de filtrar el ruido de las mediciones y de absorber la incertidumbre en los parámetros del modelo. En el presente trabajo se afrontó un problema común tanto en el área química como en la biotecnológica, la saturación de los actuadores, problema que en simulación suele ignorarse.

En condiciones ideales un modelo difuso puede aproximar lo suficiente a un proceso real, tal que al invertir dicho modelo, se cancelen las dinámicas del proceso, en la práctica, en los procesos químicos, se tiene con frecuencia un retardo debido, tanto al tiempo de reacción, como a fenómenos de transporte, añadiendo además la saturación del actuador, es difícil garantizar un seguimiento perfecto de la referencia o un error cero en estado estable. Se obtuvo un modelo difuso para controlar la tasa de crecimiento bacteriano en un bioreactor, ante las limitaciones del actuador, se propuso emplear un control predictivo, reduciendo el error encontrado en simulación. Para proponer los horizontes de control y predicción se usó la optimización branch and bound.

El controlar la tasa de crecimiento bacteriano puede mejorar la biodegradación de contaminantes presentes en aguas residuales, en este caso, del fenol, además de tener una concentración de biomasa estable, ya que el exceso generado por los lodos activados debe purgarse e incinerarse. Finalmente, Como trabajo futuro se plantea implementar una técnica adaptable para el modelo difuso, a fin de tener un controlador robusto ante cambios en el comportamiento de la biomasa.

## Referencias


1. Guadayol, J.M.: Control, Instrumentation and Automation of Chemical Process: Problems. Ediciones de la Universidad Politécnica de Cataluña, España (1999)
2. Dunn, I., Heinzle, E., Ingham, J., Prenosil, J.: Biological Reaction Engineering. Wiley-VCH Verlag GmbH, Germany (2003)
3. Ross, T.: Fuzzy Logic with Engineering Applications. John Wiley & Sons, Ltd., West Sussex (2008)
4. Teixeira, M., Assunçao, E., Avellar, R.: On relaxed lmi-based designs for fuzzy regulators and fuzzy observers. IEEE Trans. on Fuzzy Systems, 11(5) 613–623 (2003)
5. Tiwari, S.P., Srivastava, A.K.: Fuzzy rough sets, fuzzy preorders and fuzzy topologies. Fuzzy sets and systems, 210, 63–68 (2013)
6. Passino, K., Yurkovich, S.: Fuzzy control. AddisonWesley Longman Inc. California (1998)
7. Boyd, S., Ghaoui, E., Feron, E., Balakrishnan, V.: Linear Matrix Inequalities in System and Control Theory. Society for Industrial and Applied Mathematics, Filadelfia (1994)
8. Theilliol, D., Ponsart, J.C., Harmand, J., Join, C., Gras, P.: On-line estimator of unmeasured inputs for anaerobic wastewater treatment process. Control Engineering Practice, 11, 1007–1019 (2003)
9. Aceves, A.: The use and abuse of fuzzy logic for process control: one alternative to model the incomplete of information and the shadowy in a sample. In: Proc. Conf. Of Con Mantenimiento Productivo, 1, pp. 12–17 (2001)
10. Vázquez, G., Ben-Youssef, C., Waissman, J.: Two step modelling of the biodegradation of phenol by an acclimated activated sludge. Chemical Engineering J., 117(3) 245–252 (2006)
11. Martínez, S.A., Rodríguez, M.G.: Tratamiento de aguas residuales con Matlab. Reverté, México (2009)
12. Márquez, M.A., Waissman, J., Gutú, O.: Fuzzy model based iterative learning control in phenol biodegradation. Foundations of Fuzzy Logic and Soft Computing. Lecture Notes in Artificial Intelligence, Vol. 4529. Springer-Verlag Berlin Heidelberg, Germany, pp. 328–337 (2007)
13. Babuska, R.: Fuzzy Modeling for Control. International Series in Intelligent Technologies, Aachen, Alemania (1998)
14. Åström, K., Wittenmark, B.: Adaptive control. Pearson Education, India (2006)
15. Ferramosca, A., Gruber, J.K., Limón, D., Camacho, E.F.: Control predictivo para seguimiento de sistemas no lineales. Aplicación a una planta piloto. Revista Iberoamericana de Automática e Informática industrial, 10, 18–29 (2013)
16. Lendek, Z., Guerra, T., Babuska, R., De-Shutter, B.: Stability Analysis and Non-linear Observer Design Using Takagi-Sugeno Fuzzy Models. Studies in Fuzziness and Soft Computing, Vol. 262, Springer-Verlag Berlin Heidelberg, India (2010)

# Método rápido de preprocesamiento para clasificación en conjuntos de datos no balanceados

Liliana Puente-Maury[1], Asdrúbal López-Chau[2], William Cruz-Santos[2], Lourdes López-García[2]

[1] Universidad Autónoma Metropolitana, Unidad Cuajimalpa, Distrito Federal, México

[2] Universidad Autónoma del Estado de México, CU UAEM Zumpango, Estado de México, México

alchau@uaemex.mx

**Resumen.** El problema de desbalance en clasificación se presenta en conjuntos de datos que tienen una cantidad grande de datos de cierto tipo (clase mayoritaria), mientras que el número de datos del tipo contrario es considerablemente menor (clase minoritaria). En este escenario, prácticamente todos los métodos de clasificación presentan un bajo desempeño. En este artículo se propone un nuevo método de preprocesamiento, que utiliza un enfoque similar a las técnicas de basadas en enlaces Tomek, pero cuyo tiempo de ejecución es dramáticamente reducido con respecto al cálculo por fuerza bruta, comúnmente utilizado en dichas técnicas. Los resultados obtenidos en los experimentos demuestran la efectividad del método propuesto para mejorar las áreas de las curvas ROC y PRC de métodos de clasificación aplicados a conjuntos de datos reales no balanceados.

**Palabras clave:** Tomek, desbalance, clasificación.

## 1. Introducción

En clasificación, el problema de desbalance se presenta de manera natural en diversos dominios del mundo real [9]. Por ejemplo, en condiciones normales, la cantidad de operaciones fraudulentas en transacciones bancarias es considerablemente menor a las operaciones no fraudulentas. En este caso, la clase minoritaria correspondería a las primeras transacciones mencionadas, mientras que la clase mayoritaria correspondería a las segundas. En medicina, se ha observado que ciertas enfermedades afectan a un número reducido de personas; por lo que el número de expedientes de personas que las padecen es reducido, comparado con el total de expedientes médicos.

En este tipo de escenarios, es de vital importancia que los sistemas de reconocimiento automático puedan predecir correctamente instancias de clase minoritaria y, al mismo tiempo, que no dañen la precisión de las predicciones para la clase mayoritaria.

El problema de desbalance es complejo, y no solamente depende de la proporción que existe entre el número de instancias de cada clase, dicho problema es conocido como "desbalance entre clases" [1,7]. La complejidad de los datos juega un papel importante en este tipo de problemas. Entiéndase ésta como el traslape entre clases, la falta de datos representativos en algunas regiones del espacio de entrada o la existencia de subconceptos [6]. Cuando dentro de un problema de clasificación existen subconceptos que contienen pocas instancias, se presenta lo que se conoce como el "desbalance al interior de las clases".

Para mejorar el desempeño de sistemas de reconocimiento de patrones en conjuntos de datos desbalanceados, se han propuesto soluciones que intentan balancear o limpiar los datos antes de aplicarlos a métodos de clasificación existentes. Estas soluciones son llamadas métodos externos y trabajan con los datos en una etapa de preprocesamiento. En otras propuestas, se modifican los algoritmos de clasificación con la finalidad de incluir en ellos un mecanismo para hacer que las instancias de la clase minoritaria sean consideradas de mayor importancia que el resto, o en otras palabras, se fuerza a que el método de clasificación realice una generalización que favorezca a la clase minoritaria.

Uno de los primeros métodos externos para reducir el desbalance al interior de las clases fue la utilización de *enlaces Tomek*. De manera informal, un enlace Tomek está conformado por dos instancias de clase contraria (una de clase minoritaria y la otra de la mayoritaria) cuya distancia es la menor con respecto a cualquier otra instancia del conjunto de datos. La idea subyacente de los métodos que usan enlaces Tomek, consiste en identificar y eliminar del conjunto de datos, aquellos objetos de clase mayoritaria que se encuentran ya sea cerca de la frontera de decisión o al interior de un subconcepto perteneciente a la clase minoritaria. Aunque este método funciona en la mayoría de los casos, una desventaja es que su complejidad es cuadrática en espacio de almacenamiento, y casi cúbica en tiempo de cómputo. Otro método ampliamente utilizado es SMOTE [2], que agrega instancias creadas artificialmente al conjunto de datos. Esto hace poco práctico la utilización directa de enlaces Tomek y de SMOTE para datos grandes.

En este artículo, se propone un nuevo método externo para preprocesamiento de conjuntos de datos no balanceados. El método toma una idea similar al que emplea enlaces Tomek, sin embargo, en nuestro caso, la complejidad computacional es considerablemente menor, tal y como se demuestra en los resultados de los experimentos realizados. Nuestro método realiza particiones el espacio de entrada en regiones de baja entropía, y luego detecta regiones adyacentes de clase contraria. Mediante esta técnica, la cantidad de instancias candidatas a formar enlaces Tomek se reduce considerablemente. Para demostrar la efectividad de la propuesta presentada en este artículo, se utilizan seis conjuntos de datos reales disponibles públicamente en Internet, y se emplea al clasificador C4.5 para medir su desempeño antes y después de aplicar nuestros algoritmos. El resto del artículo está organizado como sigue. En la sección 2 se muestran algunas definiciones de las medidas utilizadas para comparar el desempeño de los métodos de clasificación, se describen a los enlaces Tomek y a los árboles de decisión. En la sección 3 se presentan los algoritmos desarrollados que implementan el método

propuesto y se muestra el análisis de complejidad del mismo. Los resultados de los experimentos se presentan en la sección 4. El artículo termina con conclusiones y líneas de investigación futuras.

## 2. Preliminares

En esta sección se presentan algunas medidas estadísticas relacionadas con la medición del desempeño de clasificadores en conjuntos de datos desbalanceados. También se incluye la definición de los enlaces Tomek y el algoritmo básico para su detección. Además, en la última parte de esta sección, se muestran los árboles de decisión, utilizados en una parte del método propuesto.

### 2.1. Medidas utilizadas en clasificación de conjuntos de datos no balanceados

Las métricas utilizadas para medir el desempeño de métodos de clasificación sobre conjuntos de datos balanceados, tales como precisión de clasificación y error medio, resultan inadecuadas para el problema de clasificación en datos no balanceados [5]. Por ello, se han desarrollado nuevas métricas para este tipo de escenarios. A continuación, se presentan las más importantes.

Sea $P$ el conjunto de datos de clase positiva ($p$) (conjunto de instancias de la clase minoritaria), y $N$ el conjunto de datos de clase negativa ($n$) (conjunto de instancias de clase mayoritaria). Definamos la función $f_C : X \to \{n, p\}$ como:

$$f_C(x) = \begin{cases} p \text{ Si } C \text{ clasifica a } x \in X \text{ como positivo,} \\ n \text{ si } C \text{ clasifica a } x \in X \text{ como negativo} \end{cases}$$

donde $C$, es el método de clasificación utilizado, y $X$ es el conjunto de instancias a clasificar. Entonces, los conjuntos de los falsos positivos y el de los falsos negativos se pueden formalizar como sigue:

$$FP = \{x \in X \ | x \in N, f_C(x) = p\} \text{ y } FN = \{x \in X \ | x \in P, f_C(x) = n\}$$

De manera análoga, el conjunto de los verdaderos positivos y el de los verdaderos negativos se define como:

$$TP = \{x \in X \ | x \in P, f_C(x) = p\} \text{ y } TN = \{x \in X \ | x \in N, f_C(x) = n\}$$

Dos medidas importantes son la tasa de verdaderos positivos ($TP_{rate}$) y la de falsos positivos ($FP_{rate}$), calculadas como:

$$TP_{rate} = {}^{TP}/_{(TP+FN)}$$

$$FP_{rate} = {}^{FP}/_{(FP+TN)}$$

La precisión se obtiene usando la ecuación (1), y se puede interpretar como la cantidad de objetos que fueron etiquetados como positivos(negativos), son realmente de ese tipo.

$$Precision = {}^{TP}/_{(TP+FP)}. \tag{1}$$

El recuerdo (*Recall*) es una medida que ofrece un panorama acerca de la relación entre la cantidad de muestras con etiqueta positiva/negativa que fueron predichos como positivos/negativos, y se obtiene con la ecuación (2)

$$Recall = {}^{TP}/_{(TP+FN)} \tag{2}$$

F-Measure (ecuación (3)) combina tanto la precisión como el recuerdo.

$$F - Measure = {}^{(1+\beta)^2 \cdot Recall \cdot Precision}/_{(\beta^2 \cdot Recall + Precision)}, \tag{3}$$

donde $\beta$ es un coeficiente para ajustar la importancia relativa de la precisión contra el *Recall* (usualmente $\beta = 1$).

La curva *ROC* (Receiver Operating Curve) se forma graficando $TP_{rate}$ contra $FP_{rate}$, por lo que un punto en este espacio corresponde al desempeño de un algoritmo de clasificación en una distribución dada. Esta curva es muy útil, pues ofrece una representación visual del compromiso (*trade-off*) entre los beneficios (reflejados por los $TP$) y los costos (reflejados por los $FP$) de la clasificación [6].

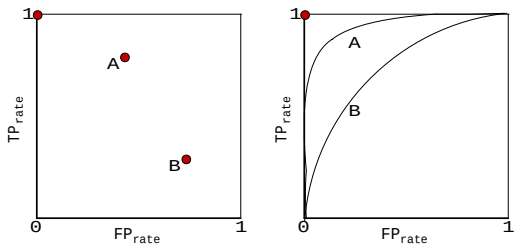



Fig. 1: Espacio ROC para clasificadores *hard-type* (izquierda) y *soft-type* (derecha).

Si el clasificador es *hard-type* (aquellos en la que la predicción indica solamente la clase a la que pertenece una instancia, no su grado de pertenencia a dicha clase), entonces producirá un par $(TP_{rate}, FP_{rate})$ que corresponde a un punto en el espacio *ROC*. El par $(0, 1)$ corresponde a una clasificación perfecta (costo cero, beneficio máximo), así que un clasificador será mejor que otro cuanto más cerca se encuentre del punto $(0, 1)$. En la Figura 1 (izquierda), el clasificador asociado al punto $A$ es mejor que aquél asociado a $B$. Si el clasificador es continuo o *soft-type* (o sea, que produce un valor numérico continuo para representar la confianza de una instancia que pertenece a la clase predicha), se puede utilizar un

umbral para producir una serie de puntos en el espacio $ROC$. Esta técnica puede generar una curva en lugar de un solo punto $ROC$. En este caso, el clasificador que tenga una mayor área bajo su curva, será mejor. En la Figura 1 (derecha), el clasificador asociado a la curva $A$ es mejor que aquél asociado a $B$.

El coeficiente de correlación de Matthews, ecuación (4), se usa como una medida de la calidad de las clasificaciones de dos clases. Devuelve un valor entre -1 (ninguna relación entre predicción y observación) y 1 (predicción perfecta).

$$MCC = \frac{TP.TN - FP.FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}. \tag{4}$$

Las curvas $PRC$ (Precision-Recall Curve) pueden ofrecer mayor información sobre la valoración del desempeño en el caso de conjuntos de datos altamente sesgados; por esto muchos trabajos actuales usan este tipo de curvas para evaluaciones de desempeño y comparaciones. Estas curvas se definen graficando la tasa de precisión contra la tasa de *Recall*. Las curvas $PR$ tienen una estrecha relación con las curvas $ROC$: una curva domina en el espacio $ROC$ si, y sólo si, domina en el espacio $PR$[6][3].

### 2.2. Enlaces Tomek

Los enlaces Tomek se definen usando el concepto de distancia. Típicamente se utiliza la Euclidiana; sin embargo, es posible utilizar cualquier otra función $d$ que satisfaga las condiciones de métrica sobre un conjunto P, $d : P \times P \mapsto R$,

1. $d(a, b) \geq 0 \ \forall a, b \in P$.
2. $d(a, b) = d(b, a) \ \forall a, b \in P$.
3. $d(a, b) \leq d(a, c) + d(c, b) \ \forall a, b, c \in P$.
4. $d(a, b) = 0$ implica que $a = b$.

Dado un conjunto de datos $X$ con atributos numéricos, un par de objetos $\alpha$, $\beta$, forman un enlace Tomek si satisfacen la siguiente condición:

$$d(\alpha, \beta) < d(\alpha, \gamma) \text{ y } d(\alpha, \beta) < d(\beta, \gamma), \ \forall \gamma \in X, \tag{5}$$

donde $d(a, b)$ es la distancia entre los objetos $a$ y $b$, $\gamma \in X$.

Si los objetos $\alpha$ y $\beta$ forman un enlace Tomek, entonces uno de ellos puede ser considerado como ruido, o ambos se encuentran cerca de la frontera de decisión. La Figura 2 ejemplifica esta idea.

Para conjuntos de datos desbalanceados, a cada objeto $x$ de clase minoritaria se le calcula el objeto $y$ de clase mayoritaria que forme un enlace Tomek con él, y se elimina $y$ del conjunto de datos. El Algoritmo 1 muestra el pseudocódigo que implementa esta idea. En los algoritmos mostrados, $X$ es el conjunto de datos, con $X = X^+ \cup X^-$ y $X^+ \cap X^- = \emptyset$, donde $X^+$ y $X^-$ son el conjunto de instancias de clase minoritaria y mayoritaria, respectivamente.

[3] Una curva $PRC$ dominante, se encuentra en la parte superior derecha del espacio $PR$.

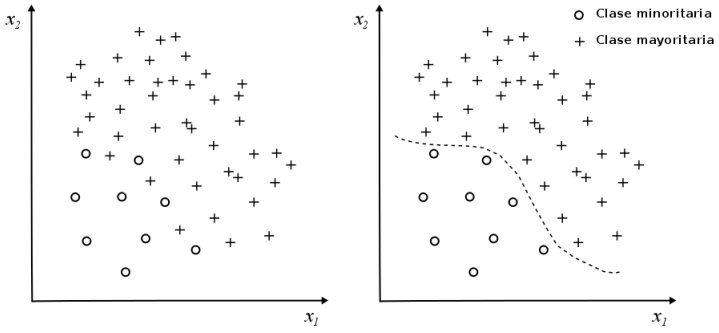



Fig. 2: Aplicación de enlaces de Tomek para eliminar instancias de clase mayoritaria.

**Algorithm 1:** Preprocesamiento usando enlaces Tomek.

**Input** : $X$: Conjunto de datos
**Output**: $X_{proc}$: Conjunto de datos procesado

**begin**
  $X_{proc} \leftarrow X$`//Copia de` $X$;
  $T_{Links} \leftarrow$ Pares de objetos que forman enlaces Tomek (Algoritmo 2);
  **foreach** $x_i \in T_{links}$ && $x_i \in X^-$ **do**
    $X_{proc} \leftarrow X_{proc} - \{x_i\}$`//Elimina` $x_i$
  **end**
  **return** $X_{proc}$
**end**

El Algoritmo 2 muestra el pseudocódigo para determinar todos los enlaces Tomek de un conjunto de datos. La implementación simple usando fuerza bruta tiene un orden de complejidad aparente de $O(n^2)$ (doble ciclo: el exterior de 1 a $n$ y el interior que recorre los restantes $n-1$ puntos). Sin embargo, para determinar que se cumpla la condición (5), se requiere un ciclo adicional. Por lo que el peor caso es $O(n^3)$. Esta es una de las principales limitaciones prácticas al aplicar enlaces de Tomek en conjuntos de datos grandes.

### 2.3. Árboles de decisión

El método propuesto utiliza un árbol de decisión para realizar particiones en el espacio de entrada. En esta subsección, se realiza una breve revisión de este clasificador. Los árboles de decisión son un método de aprendizaje supervisado cuya estructura puede ser representada por un grafo acíclico que forma un árbol [4]. Las partes principales de dicha estructura son nodos (raíz, internos y hojas) y vértices que unen a los nodos.

**Algorithm 2:** Cálculo de objetos que forman enlaces Tomek.

**Input** :
$X$: Conjunto de datos
**Output**:
$T_{links}$: Colección de pares de objetos que forman enlaces Tomek

```
begin
    X+ = {x_i ∈ X tal que y_i = +1}//Clase minoritaria;
    X− = {x_i ∈ X tal que y_i = −1}//Clase mayoritaria;
    foreach x_i ∈ X− do
        foreach x_j ∈ X+ do
            dT ← d(x_i, xi);
            if ¬∃δ ∈ X tal que dT > d(δ, x_j) o d(δ, xj) then
                T_links ← T_links ∪ (x_i, x_j);
            end
        end
    end
    return T_links
end
```

Se ha demostrado que crear un árbol de decisión óptimo en términos de tamaño y desempeño es un problema NP completo [3,8], es por ello que se utilizan heurísticas que permiten construir árboles de una manera más eficiente. El método básico para *inducir* un árbol de decisión a partir de datos, consiste en seleccionar un atributo en el cual se divida en dos un conjunto de datos, de tal forma que después de separado, cada parte sea "más pura" que el conjunto antes de la partición. Elegir el mejor atributo en el cual realizar la división, implica medir la pureza de un conjunto de instancias. Las medidas más populares utilizadas en árboles de decisión son la entropía, ganancia de información, la tasa de ganancia y el índice Gini. Las ecuaciones (6), (7), (8) y (9) muestran cómo calcular estas medidas.

$$Entropy(X) = -\sum_{i=1}^{n} p_i log_2(p_i) \tag{6}$$

$$Gain(X, A) = Entropy(S) - \sum_{v \in valores(A)} p_i log_2(p_i) \tag{7}$$

$$GainRatio(A) = \frac{Gain(A)}{-\sum_{j=1}^{v} \frac{\|D_j\|}{\|D\|} log_2\left(\|D_j\| \|D\|\right)} \tag{8}$$

$$Gini(X) = 1 - \sum_{i=1}^{m} p_i^2 \tag{9}$$

## 3. Método propuesto

La implementación directa del cálculo de enlaces Tomek (Algoritmo 2) tiene una complejidad computacional superior a la cuadrática. Para evitar este elevado costo, se desarrolló un nuevo método que realiza un preprocesado rápido a los datos similar al que emplea enlaces Tomek, y que permite mejorar el desempeño de algoritmos de clasificación.

La idea del método propuesto en este artículo es simple de implementar, pero efectiva y eficiente. El procedimiento general puede resumirse en los siguientes pasos:

1. Particionar el espacio de entrada en regiones de baja entropía,
2. Numerar las particiones,
3. Determinar las particiones adyacentes de cada partición encontrada,
4. Detectar las instancias más cercanas que pertenecen a dos particiones adyacentes de clase contraria,
5. Repetir el paso 4 para cada partición,
6. Eliminar del conjunto de datos aquellas instancias detectadas en el paso 4 que sean de clase mayoritaria.

Para el particionado del espacio de entrada en regiones de baja entropía, el método presentado utiliza un árbol de decisión C4.5. La idea es realizar esta tarea de una forma rápida, de tal manera que cada región dividida del espacio de entrada contenga la mayor cantidad de instancias de una clase, y al mismo tiempo, la menor cantidad de instancias de clase contraria.

Dos observaciones importantes, que permitieron diseñar un nuevo método para detectar rápidamente instancias de clase mayoritaria que pueden ser eliminadas del conjunto de datos, son las siguientes:

1. En una región del espacio de entrada con entropía mínima, los enlaces Tomek se presentan cerca de regiones adyacentes con instancias de clase contraria.
2. Dos regiones adyacentes con entropía mínima y de la misma clase, no pueden formar enlaces Tomek entre ellas.

Una parte esencial consiste en determinar cuáles particiones del espacio de entrada son adyacentes. Para ello, se diseñaron dos algoritmos. El primero de ellos, Algoritmo 3, realiza un recorrido por un árbol de decisión recopilando información sobre los límites que definen cada hoja del árbol. Las hojas corresponden a las regiones de baja entropía. El segundo, Algoritmo 4, detecta todas las particiones adyacentes de cada partición, haciendo una búsqueda en los límites determinados durante el recorrido del árbol. La Figura 3 muestra un ejemplo hipotético de un espacio de entrada particionado, y la estructura del árbol que representa las particiones. La Tabla 1 muestra los límites de cada partición encontrados por los Algoritmos 3 y 4. Las columnas $B_{i,L}$ y $B_{i,H}$ corresponden a los límites inferior (L) y superior (H) del atributo i-ésimo, respectivamente.

El análisis de complejidad de nuestro método es el siguiente. Sin perder generalidad, se puede suponer que los enlaces Tomek se determinarán desde

**Algorithm 3: Cálculo de los límites de las hiper cajas.**

**Input** : $\mathcal{T}$: Un árbol de decisión inducido a partir de X
**Output**: $M$: Matrix con los límites de cada hoja de $\mathcal{T}$

**begin**
  Crear vector $B \in R^{2d}$;
  Inicializar con $-\infty$ los elementos de $B$ con índice 1 a d;
  Inicializar con $+\infty$ los elementos de $B$ con índice d+1 a 2d;
  Invocar a DescubreLímites($B, M$) desde nodo raíz;
  regresar $M$;
**end**

**DescubreLímites(**$B$,$M$**)**;
**if** *nodo visitado es hoja* **then**
  Agregar $B$ como la última fila de $M$;
**end**
**else**
  Crear $B_L$ y copiar límites desde B;
  $B_L$: Cambiar $h_{ij}$ usando el índice de atributo y el valor de particionado del nodo actual;
  Invocar a DescubreLímites($B_L$, M) con el hijo izq. del nodo actual;
  Crear $B_R$ y copiar límites desde B;
  $B_R$: Cambiar $l_{ij}$ usando el índice de atributo y el valor de particionado del nodo actual;
  Invocar a DescubreLímites($B_R$, M) con el hijo derecho del nodo actual;
**end**

**return**

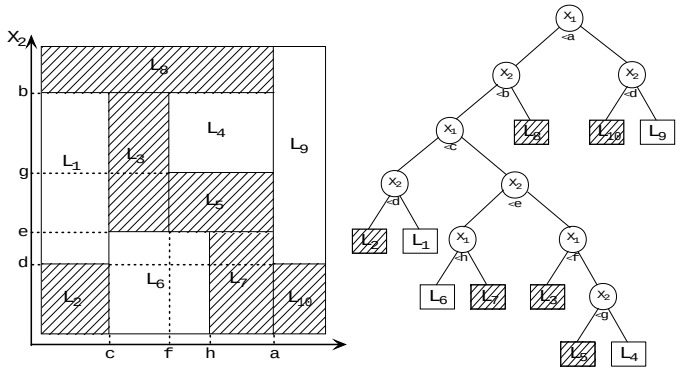



Fig. 3: Particiones en el espacio de entrada.

el centro de una partición hacia el centro de otra. Entonces, habría que calcular las distancias desde dicho centro, hacia cada centro de las particiones adyacentes que contienen instancias de clase contraria. Para conjuntos de datos balanceados, se puede suponer que las instancias están distribuidas de manera uniforme

dentro de las particiones, y que estas tienen una entropía mínima. Bajo estas suposiciones, el número de instancias en cada partición es aproximadamente $n/m$, donde $m$ es la cantidad total de particiones. El peor caso ocurre cuando todas las particiones son adyacentes entre sí, y la mitad de ellas contienen instancias de clase mayoritaria y la otra mitad de clase minoritaria. El número de distancias que se tendrían que calcular sería el especificado por la ecuación (10).

**Algorithm 4: Determinación de los vecinos de una hiper caja.**

**Input** : $M$: Matriz de límites (Algoritmo 3)
$L$: Hiper caja referencia
**Output**: $L_{vecino}$: Lista de vecinos de L

```
begin
    for i=1 to d do
        Crear lista L_vecino con todas las cajas que compartan un límite con la
        caja L (usar M para detectar esto);
        foreach elemento e_j de L_1 do
            if e_j forma espacio conectado con caja L then
                Agregar e_j a L_vecino
            end
        end
        Eliminar elementos repetidos en L;
        regresar L_vecino;
    end
end
```

Tabla 1: Limites de las particiones.

| Partición | $B_{1,L}$ | $B_{1,H}$ | $B_{2,L}$ | $B_{2,H}$ |
|---|---|---|---|---|
| $L_1$ | $-\infty$ | c | d | b |
| $L_2$ | $-\infty$ | c | $-\infty$ | d |
| $L_3$ | c | f | e | b |
| $L_4$ | f | a | g | b |
| $L_5$ | f | a | g | b |
| $L_6$ | c | h | $-\infty$ | e |
| $L_7$ | h | a | $-\infty$ | e |
| $L_8$ | $-\infty$ | a | b | $\infty$ |
| $L_9$ | a | $\infty$ | d | $\infty$ |
| $L_{10}$ | a | $\infty$ | $-\infty$ | d |

$$\left(\frac{m}{2}\right)\left(\frac{m}{2}\right)\left(\frac{n}{m}\right)\left(\frac{n}{m}\right) = \left(\frac{n^2}{4}\right) \tag{10}$$

Lo que da una complejidad de $O(n^2)$. En estas condiciones, el método propuesto tiene una complejidad computacional similar al caso de cálculo por fuerza bruta. Sin embargo, en la práctica hemos observado que las cosas ocurren rara vez como en el peor caso. En general, lo que sucede es lo siguiente:

1. El número de particiones cambia de un conjunto de datos a otro. Atribuimos esto a la diferencia entre complejidad de los conceptos en ellos [6].
2. En conjuntos de datos no balanceados, la cantidad de particiones que contienen instancias de clase minoritaria es significativamente menor a las particiones con instancias de clase contraria.
3. Algunas particiones contienen muchas instancias, mientras que otras contienen pocas de ellas.
4. No todas las particiones son vecinas entre sí. Más aún, la mayoría de las particiones de clase mayoritaria tienen particiones adyacentes con clase similar.

Como el número de particiones de clase minoritaria es pequeño, la cantidad de distancias que se calcula es mucho menor al calculado por el método de fuerza bruta. Esto hace que nuestra propuesta sea superior en velocidad.

## 4. Resultados

Para probar el desempeño del método propuesto, y realizar comparaciones con el método de fuerza bruta, se utilizaron los conjuntos de datos mostrados en la Tabla 3. Estos conjuntos de datos se encuentran disponibles públicamente en el repositorio Keel $http://sci2s.ugr.es/keel/datasets.php$.

Tanto el método propuesto como el de fuerza bruta fueron implementados en lenguaje de programación Java. Se utilizó Weka como herramienta para cargar archivos, realizar evaluaciones y el algoritmo C4.5 (llamado J48 en Weka) para particionado del espacio de entrada. Todos los experimentos fueron ejecutados en una computadora con las siguientes características: Procesador Intel i7 3720QM 2.60 GHz, 8.0 GB en RAM, Sistema Operativo Windows 7 de 64 bits.

Para validar los resultados, se repitieron 100 veces los experimentos en cada conjunto de datos utilizado. En cada ejecución se eligieron aleatoriamente el 70 % de los datos para entrenamiento, y el 30 % restante se usó para prueba. Los resultados presentados en la Tabla 2 corresponden al promedio de ese número de ejecuciones. Una vez procesados los datos, el método C4.5 fue aplicado al conjunto de datos con las instancias de clase mayoritaria eliminadas. Por razones de espacio, sólo se presentan los resultados con este clasificador. El significado de las columnas de la Tabla 2 es el siguiente: La columna T representa el tiempo de preprocesamiento en mili segundos; TP es la tasa de verdaderos positivos; FP es la tasa de falsos positivos; Prec representa la precisión; Recall es el recuerdo con respecto a la clase indicada en la última columna; F-M es F-Measure; MCC es el coeficiente phi; ROC es el área bajo la curva ROC y PRC es el área bajo la curva PR. Las filas *Ninguno*, son los resultados sin preprocesamiento; las filas *Propuesta* corresponden a los resultados obtenidos con la aplicación del método

Tabla 2: Resultados obtenidos en los experimentos.

| Preprocesamiento Método | T(ms) | TP | FP | Prec | Recall | F-M | MCC | ROC | PRC | Class |
|---|---|---|---|---|---|---|---|---|---|---|
| Conjunto de datos **car-good** | | | | | | | | | | |
| Ninguno | NA | 0.954 | 0.033 | 0.557 | 0.954 | 0.698 | 0.713 | 0.962 | 0.589 | Positiva |
| Ninguno | | 0.967 | 0.046 | 0.998 | 0.967 | 0.982 | 0.713 | 0.962 | 0.997 | Negativa |
| Propuesta | **62.134** | 0.969 | 0.038 | 0.518 | 0.969 | 0.671 | 0.691 | 0.965 | 0.526 | Negativa |
| Propuesta | | 0.962 | 0.031 | 0.999 | 0.962 | 0.980 | 0.691 | 0.965 | 0.998 | Positiva |
| Tomek | 18,303.949 | 1.000 | 1.000 | 0.040 | 1.000 | 0.076 | 0.000 | 0.500 | 0.040 | Negativa |
| Tomek | | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.500 | 0.960 | Positiva |
| Conjunto de datos **dermatology-6** | | | | | | | | | | |
| Ninguno | NA | 0.965 | 0.400 | 0.398 | 0.965 | 0.468 | 0.446 | 0.782 | 0.377 | Positiva |
| Ninguno | | 0.600 | 0.035 | 0.648 | 0.600 | 0.621 | 0.446 | 0.782 | 0.977 | Negativa |
| Propuesta | **1.957** | 0.958 | 0.283 | 0.457 | 0.958 | 0.536 | 0.529 | 0.837 | 0.432 | Positiva |
| Propuesta | - | 0.717 | 0.042 | 0.777 | 0.717 | 0.744 | 0.529 | 0.837 | 0.982 | Negativa |
| Tomek | 135.002 | 0.966 | 0.410 | 0.392 | 0.966 | 0.462 | 0.438 | 0.778 | 0.372 | Positiva |
| Tomek | | 0.590 | 0.034 | 0.638 | 0.590 | 0.612 | 0.438 | 0.778 | 0.976 | Negativa |
| Conjunto de datos **ecoli-0-1-4-7_vs_5-6** | | | | | | | | | | |
| Ninguno | NA | 0.913 | 0.746 | 0.120 | 0.913 | 0.198 | 0.101 | 0.584 | 0.112 | Positiva |
| Ninguno | | 0.254 | 0.087 | 0.351 | 0.254 | 0.291 | 0.101 | 0.584 | 0.938 | Negativa |
| Propuesta | **1.468** | 0.897 | 0.661 | 0.138 | 0.897 | 0.222 | 0.144 | 0.619 | 0.129 | Positiva |
| Propuesta | | 0.339 | 0.103 | 0.489 | 0.339 | 0.393 | 0.144 | 0.619 | 0.943 | Negativa |
| Tomek | 14.531 | 0.895 | 0.649 | 0.144 | 0.895 | 0.230 | 0.151 | 0.624 | 0.132 | Positiva |
| Tomek | | 0.351 | 0.105 | 0.490 | 0.351 | 0.400 | 0.151 | 0.624 | 0.943 | Negativa |
| Conjunto de datos **glass-1** | | | | | | | | | | |
| Ninguno | NA | 0.981 | 0.914 | 0.372 | 0.981 | 0.537 | 0.087 | 0.535 | 0.375 | Positiva |
| Ninguno | | 0.086 | 0.019 | 0.335 | 0.086 | 0.135 | 0.087 | 0.535 | 0.670 | Negativa |
| Propuesta | **1.951** | 0.978 | 0.902 | 0.375 | 0.978 | 0.539 | 0.098 | 0.540 | 0.379 | Positiva |
| Propuesta | | 0.098 | 0.022 | 0.367 | 0.098 | 0.153 | 0.098 | 0.540 | 0.673 | Negativa |
| Tomek | 33.132 | 0.980 | 0.913 | 0.372 | 0.980 | 0.536 | 0.086 | 0.537 | 0.378 | Positiva |
| Tomek | | 0.087 | 0.020 | 0.340 | 0.087 | 0.136 | 0.086 | 0.537 | 0.671 | Negativa |
| Conjunto de datos **vowel0** | | | | | | | | | | |
| Ninguno | NA | 0.958 | 0.382 | 0.337 | 0.958 | 0.450 | 0.419 | 0.791 | 0.351 | Positiva |
| Ninguno | | 0.618 | 0.042 | 0.825 | 0.618 | 0.694 | 0.419 | 0.791 | 0.962 | Negativa |
| Propuesta | **4.804** | 0.957 | 0.406 | 0.343 | 0.957 | 0.450 | 0.413 | 0.781 | 0.355 | Positiva |
| Propuesta | | 0.594 | 0.043 | 0.795 | 0.594 | 0.665 | 0.413 | 0.781 | 0.961 | Negativa |
| Tomek | 274.320 | 0.958 | 0.376 | 0.338 | 0.958 | 0.452 | 0.422 | 0.794 | 0.352 | Positiva |
| Tomek | | 0.624 | 0.042 | 0.835 | 0.624 | 0.702 | 0.422 | 0.794 | 0.963 | Negativa |

presentado en este artículo; las filas *Tomek* corresponden a las aplicadas con el Algoritmo 1 con el enfoque de fuerza bruta.

Como puede observarse, el método de preprocesamiento propuesto es eficiente, y permite mejorar el desempeño del algoritmo de clasificación C4.5 para conjuntos de datos no balanceados. Esto coincide con los resultados encontrados en [10], donde se indica que el preprocesamiento contribuye a que C4.5 obten-

Tabla 3: Conjuntos de datos utilizados en los experimentos.

| Nombre | Tamaño | Atributos | Clase mayoritaria | Clase minoritaria | RI |
|---|---|---|---|---|---|
| car-good_vs_3-6-8 | 1,728 | 6 | 905 | 99 | 9.14 |
| cleveland-0_vs_4 | 173 | 13 | 160 | 13 | 12.31 |
| dermatology-6 | 358 | 34 | 338 | 20 | 16.90 |
| ecoli-0-1-4-7_vs_5-6 | 332 | 6 | 307 | 25 | 12.28 |
| glass1 | 214 | 10 | 138 | 76 | 1.82 |
| vowel0 | 988 | 13 | 898 | 90 | 9.98 |

ga mejores resultados. Aunque no se presentan resultados, otros clasificadores también son beneficiados con este tipo de procesamiento a los datos.

## 5. Conclusiones

El problema de clasificación en conjuntos de datos no balanceados representa actualmente un reto importante para las comunidades científicas de inteligencia artificial, minería de datos y aprendizaje automático. Son varios los factores que hacen que un problema de este tipo sea complicado, por ejemplo, desbalance entre las clases, desbalance al interior de las clases e instancias anómalas. Los métodos externos realizan un preprocesamiento a los conjuntos de datos no balanceados para cumplir con uno o más de los siguientes objetivos: balancear el conjunto de datos, quitar instancias consideradas como ruido, eliminar traslape entre clases o buscar prototipos que representen el conjunto de datos de una manera que sea fácil de procesar por métodos de clasificación o agrupamiento.

En este artículo, se presenta un nuevo método rápido de preprocesamiento para conjuntos de datos no balanceados. El método sigue una idea similar a los que usan enlaces Tomek, sin embargo, el tiempo de ejecución es dramáticamente reducido. Los resultados obtenidos con el método de clasificación empleado (C4.5) muestran que, una vez preprocesado el conjunto de datos, el desempeño de C4.5 es mejorado. Como trabajo futuro, se plantea la aplicación el método en conjuntos de datos gigantes (Big Data) y la modificación para flujos de datos de alta velocidad.

## Referencias

1. Batista, G.E.A.P.A., Prati, R.C., Monard, M.C.: A study of the behavior of several methods for balancing machine learning training data. SIGKDD Explor. Newsl. 6(1), 20–29 (Jun 2004)
2. Chawla, N., Bowyer, K., Hall, L., Kegelmeyer, W.: Smote: Synthetic minority oversampling technique. Journal of Artificial Intelligence Research 16, 321–357 (2002)
3. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman & Co., New York, NY, USA (1979)

4. Han, J., Kamber, M., Pei, J.: Data Mining: Concepts and Techniques. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 3rd edn. (2011)
5. He, H.: Self-Adaptive Systems for Machine Intelligence. Wiley (2011)
6. He, H., Garcia, E.A.: Learning from imbalanced data. IEEE Trans. on Knowl. and Data Eng. 21(9), 1263–1284 (Sep 2009)
7. Hulse, J.V., Khoshgoftaar, T.: Knowledge discovery from imbalanced and noisy data. Data & Knowledge Engineering 68(12), 1513–1542 (2009)
8. Hyafil, L., Rivest, R.L.: Constructing optimal binary decision trees is np-complete. Inf. Process. Lett. 5(1), 15–17 (1976)
9. Japkowicz, N., Stephen, S.: The class imbalance problem: A systematic study. Intelligent Data Analysis 6(5), 429 (2002)
10. Luengo, J., Fernandez, A., Herrera, F., Herrera, F.: Addressing data-complexity for imbalanced data-sets: A preliminary study on the use of preprocessing for c4.5. In: Intelligent Systems Design and Applications, 2009. ISDA '09. Ninth International Conference on. pp. 523–528 (2009)

# Redes bayesianas aplicadas a las condiciones climáticas al interior de un invernadero con ventilación natural

Alejandra Álvarez-López[1], Oscar Delfín-Santiesteban[1], Enrique Rico-García[2], Guillermo De la Torre-Gea[1]

[1] CA Biosistemas, Universidad Tecnológica de Corregidora, Querétaro, México

[2] CA Ingeniería de Biosistemas, División Estudios de Posgrado, Facultad de Ingeniería, Universidad Autónoma de Querétaro, Santiago de Querétaro, Querétaro, México

gtorre@abanet.mx

**Resumen.** La ventilación natural en invernaderos produce altas tasas de intercambio de aire; sin embargo, este comportamiento se producen cerca de las ventanas, lo que ocasiona un bajo intercambio de aire en la zona central del invernadero, debido a un efecto de estancamiento que reduce la distribución del viento en todo el invernadero. La predicción de los gradientes en un invernadero con ventilación natural es difícil de lograr, debido a la naturaleza inherentemente estocástica del flujo de aire. Las Redes bayesianas son técnicas numéricas de incertidumbre que se pueden utilizar para estudiar este problema. Se obtuvo un conjunto de datos experimentales: temperatura del aire, humedad del aire, velocidad del viento, y concentración de CO2 a uno y tres metros sobre el suelo, en el espacio de cultivo. El conjunto de datos fue discretizado y utilizado para desarrollar un modelo de Red Bayesiana que describe las relaciones entre las variables estudiadas. El modelo muestra las diferencias que nos permitan identificar el grado de dependencia de las variables, así como cuantificar su inferencia.

**Palabras clave:** Invernaderos, redes bayesianas, ventilación natural, clima.

## 1. Introducción

Es común que en los invernaderos se presenten temperaturas muy altas durante el verano. Este hecho provoca problemas para los agricultores que no tienen equipo de refrigeración para evitar el sobrecalentamiento. Al interior de los invernaderos, la distribución adecuada de los parámetros climáticos, tales como el flujo de aire, temperatura del aire, humedad del aire y la concentración de $CO_2$, son los principales factores que influyen en la uniformidad de crecimiento de los cultivos. En la actualidad, se han realizado numerosos estudios sobre la ventilación natural, centrándose en la comprensión de las relaciones entre las variables que definen el clima dentro del invernadero, como se muestra a continuación.

La velocidad del viento fuera del invernadero es un factor importante para la definición de flujo de aire y el clima dentro del invernadero. Las condiciones de frontera de la distribución de la velocidad del viento se deducen de los datos experimentales y de la dirección del viento con respecto al eje longitudinal del invernadero (De la torre- Gea *et al*., 2011b). Rico-García *et al*. (2006) mostraron que un invernadero con ventanas cenitales orientadas a la dirección del viento o barlovento funciona mejor que orientadas de forma opuesta o sotavento, mientras que en invernaderos con ventanas laterales sucede lo contrario, esto indica que la orientación del viento afecta el nivel de ventilación.

En un estudio realizado por Khaoua *et al*. (2006), pudieron determinar las tasas de ventilación entre 9 y 26,5 intercambios de aire por hora para el barlovento y 03.07 a 12.05 en la condición de sotavento, respectivamente. Un buen diseño puede mantener las condiciones climáticas aceptables y uniformes para los casos particulares en donde el viento es perpendicular al eje principal del invernadero. Por otra parte, la velocidad del viento presenta una influencia lineal con relación a las tasas de intercambio de aire, mientras que la dirección del viento no afecta en la misma magnitud. Majdoubi *et al*. (2009) determinaron que el flujo de aire por encima del cultivo es mayor que al interior y por debajo de éste. El viento al interior del invernadero propicia que el aire cálido y húmedo salga a través de las ventanas cenitales, sin embargo aún no se conoce cómo el aumento de la ventilación influye en una mejor uniformidad en las condiciones climáticas, debido a la escaza información sobre el movimiento del aire y su relación con la eficacia de la refrigeración y la uniformidad del medio ambiente (Sase, 2006).

El efecto de la radiación solar y temperatura a menudo se relacionan mediante el establecimiento de modelos que toman en cuenta el calentamiento de la pared y el calor específico del material con que está constituido el invernadero. La transferencia de radiación dentro del propio cultivo sigue siendo la principal preocupación, ya que determina las dos principales funciones fisiológicas de los cultivos: de transpiración y la fotosíntesis. Este problema actualmente no se ha descifrado y probablemente recibirá mucha atención en los próximos años (Bournet y Boulard, 2010). Pontikakos *et al*. (2006) analizaron los datos obtenidos a partir de un modelo de Dinámica de Fluidos Computacional (CFD), mostrando que la mayor temperatura externa es un parámetro fundamental que define el comportamiento en general de las temperaturas al interior del invernadero, mientras que la dirección del viento define las temperaturas en regiones específicas del invernadero.

Según Molina-Aiz *et al*. (2006), determinaron que la temperatura media del aire al interior del invernadero puede variar entre 28,2-32,9º C, cuando una temperatura exterior es de 26º C, presentándose variaciones de 13º C con respecto al exterior. Nebbali *et al*. (2006) utilizaron un método semianalítico para determinar el perfil de temperaturas del suelo a partir de los parámetros meteorológicos, para ayudar en la evaluación de cambio de flujo de calor entre la superficie del suelo y el aire. De acuerdo con los resultados de Majdoubi *et al*. (2009), la convección y la radiación son las formas dominantes de la transferencia de calor. Sus mediciones mostraron que la diferencia entre la temperatura del aire dentro y fuera del invernadero está fuertemente ligada a la radiación solar y en segundo lugar a la velocidad del viento.

Rico-García *et al*. (2008), mostraron que la ventilación en invernaderos debido al efecto de la temperatura produce altas tasas de intercambio de aire; Sin embargo, el

movimiento del aire se producen cerca de las ventanas solamente, sin que haya intercambio de aire en la zona central de invernadero, debido a un efecto de estancamiento que reduce la distribución del viento en todo el invernadero. Por otra parte, Chow y Hold (2010) obtuvieron que la radiación térmica sin la participación del aire, altera la distribución de la temperatura del aire en la zona superior, y ésta a su vez afecta a la temperatura del aire por conducción y convección. Las condiciones térmicas de las paredes de un invernadero definen las altas temperaturas del aire, pero no afectan su distribución. La radiación juegan un papel importante en la distribución de calor y la humedad relativa influye en la transferencia de calor.

Algunos estudios como los de Roy y Boulard (2005), Roy *et al*. (2008), Campen (2008), Kim (2008), y Majdoubi *et al*. (2009) simularon las distribuciones de humedad dentro del invernadero, obteniendo buenas aproximaciones con diferentes métodos, que incluyen modelos de CFD. El estudio de humedad es importante para la interacción el cultivo y su entorno. Sólo unos pocos estudios han logrado obtener modelos de gradientes en invernaderos. Actualmente no existen modelos para predecir gradientes de $CO_2$ siendo que influye directamente en la asimilación de los cultivos (Teittel *et al*., 2010).

La predicción de gradientes en un invernadero es difícil, debido a la naturaleza inherentemente estocástica del flujo de aire y por la cantidad de factores que influyen en la definición de las condiciones climáticas, por lo que es necesario incorporar nuevas técnicas que tomen en cuenta muchas variables a la vez. El objetivo de este estudio es abordar este problema mediante el enfoque de las Redes Bayesianas, para describir las relaciones entre las variables en un invernadero con ventilación deficiente. Las Redes bayesianas (BN por sus siglas en inglés) son técnicas numéricas de incertidumbre que hacen uso de la inferencia bayesiana como método heurístico (De la torre-Gea *et al*., 2011a).

## 2. Redes bayesianas

Las Redes Bayesianas (BN) son tipos de representación del conocimiento desarrollado en el campo de la inteligencia artificial para realizar aproximaciones en el campo del razonamiento (Pearl, 1988; Mediero, 2007; Gámez *et al*, 2011; Zaidan *et al*, 2011). Una BN es un gráfico acíclico cuyos nodos corresponden directamente a los conceptos o variables aleatorias y cuyos enlaces se corresponden con las relaciones o funciones (Correa *et al*., 2009). Las variables se definen en un dominio discreto o cualitativo, y las relaciones funcionales describen las inferencias causales expresadas en términos de probabilidades condicionales que se muestra en la ecuación (1):

$$P(x_1,\ldots,x_n) = \prod_{i=1}^{n} P(x_i \mid \text{parents}\ (x_i)) \qquad (1)$$

Una BN se puede utilizar para identificar las relaciones entre las variables anteriormente indeterminadas o para describir y cuantificar estas relaciones, incluso con un conjunto de datos incompletos (Hruschka *et al*, 2007; Reyes, 2010). El algoritmo de solución de BN permite el cálculo de la distribución de probabilidad esperada de las variables de salida. El resultado de este cálculo depende de las

distribuciones de las probabilidades de las variables de entrada. A nivel global, las BN pueden ser percibidas como una distribución de probabilidad conjunta de una colección de variables aleatorias discretas (Garrote *et al*., 2007).

$$P(c_j \mid x_i) = P(x_i \mid c_j)\, P(c_j) \,/ \sum_k P(x_i \mid c_k)\, P(c_k) \qquad (2)$$

Una probabilidad a priori $P(c_j)$ es una probabilidad de que una muestra xi pertenezca a la clase $c_j$, sin ninguna información sobre sus valores característicos, como se muestra en la ecuación (2). Las máquinas de aprendizaje en inteligencia artificial, están relacionadas con los métodos de minería de datos, la clasificación o agrupación y el reconocimiento de patrones. Los métodos estadísticos de aprendizaje automático se pueden aplicar al marco de la estadística bayesiana, sin embargo el aprendizaje automático se puede emplear en una variedad de técnicas de clasificación para producir otros modelos de BN. El objetivo de una BN de aprendizaje es encontrar un arreglo de red que mejor describa los datos observados.

En los modelos de aprendizaje, el método más representativo es el de *"búsqueda y resultado"* basado en el algoritmo K2. Dicho algoritmo comienza asignado un nombre a cada variable sin *"padres"*. A continuación agrega a cada variable un padre de forma incremental, la cual en su mayoría aumenta la puntuación de la estructura resultante. Cuando cualquier adición no puede aumentar el marcador, deja de aumentar padres a la variable. Tomando en cuenta un conocimiento previo del ordenamiento de las variables en base a su grado de dependencia, el espacio de búsqueda en esta restricción es mucho menor que el espacio de toda una estructura y no hay necesidad de comprobar los ciclos en el proceso de aprendizaje. Si no se tiene conocimiento anterior sobre el ordenamiento de las variables, se procede a nuevas búsquedas (Guoliang, 2009).

## 3. Materiales y métodos

Se realizaron muestreos y mediciones del flujo de aire mediante anemometría omnidireccional. Se obtuvo un conjunto de datos experimentales en un período de 36 horas comprendidos entre el 21 al 22 de agosto del 2011, mediante el empleo de sensores colocados en la parte central al interior del invernadero. El conjunto de datos se compone de las variables: Temperatura del aire, Humedad del aire, Velocidad del viento y Concentración de $CO_2$. Las mediciones fueron obtenidas a dos alturas; a un metro en el interior del cultivo y a tres metros del suelo sobre el cultivo. Las mediciones de temperatura y humedad fueron realizadas a intervalos de cuatro minutos por medio de un sensor de tipo LM335. La concentración de $CO_2$ fue determinada mediante un sensor de dióxido de carbono de tipo FYA600CO2H. La velocidad y dirección del aire fue determinada mediante anemómetros omnidireccionales, cuyo rango de operación es de 0 ms-1 a 20 ms-1 con una precisión de 0,03 m s-1. El conjunto de datos al interior del invernadero fueron tomados entre el 21 y 25 de agosto del 2011. Los datos fueron discretizados mediante el sistema ELVIRA, como se muestra posteriormente para ser empleados en el desarrollo del modelo de Redes Bayesianas que describe las relaciones entre todas las variables.

El invernadero está localizado en la Universidad Autónoma de Querétaro, Campus Amazcala, la cual se ubica en las coordenadas siguientes: longitud 100° 24' W; latitud 20° 36' N; y altitud 1820 m. La superficie del invernadero es de 964.8 $m^2$ (26.8 m de ancho y 36 m de largo). El invernadero es de 5.49 m de altura y 4.2 m de altura a la canaleta, con orientación norte-sur. Cuenta con cuatro ventanas cenitales, uno en cada nave, (0.9 m de ancho y 28 m de longitud) y cuatro ventanas de pared. Las ventanas en las paredes norte y sur son de 2.5 m de ancho y 20 m de longitud, y las ventanas de las paredes este y oeste son de 2.5 m de ancho y 28 m de longitud. Todas las ventanas son enrollables. Las ventanas cenitales y laterales constituyen el 10% y 24% de la cobertura total, respectivamente como se muestra en la Figura 1. Se empleó como cultivo la especie *Lycopersicum pimpinellifolim*, con una densidad de 2.7 plantas/$m^2$.

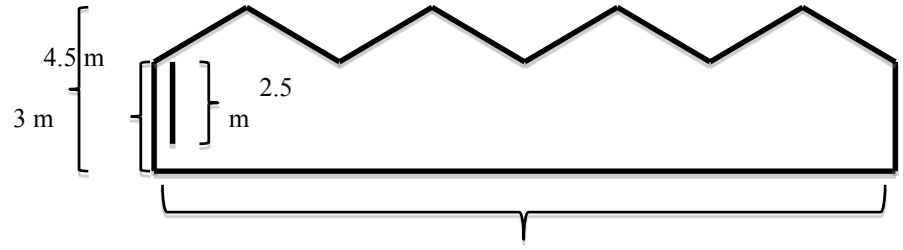



**Fig. 1.** Geometría del invernadero.

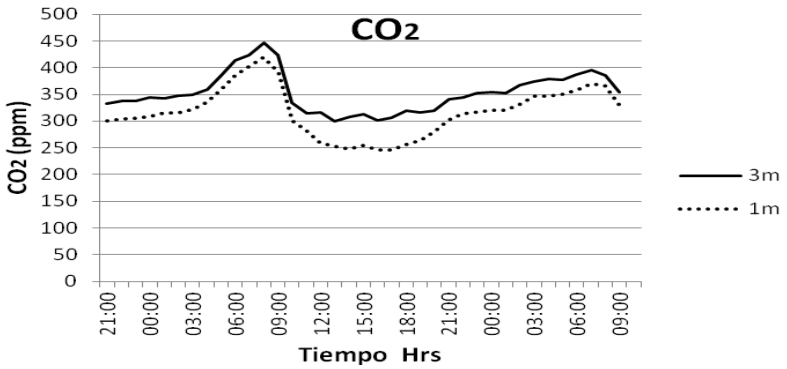
El comportamiento de las variables estudiadas fue similar tanto en los pasillos como al interior del cultivo, sin embargo, a un metro de altura, la temperatura fue más alta en el día y más baja en las noches, mientras que la humedad fue menor, como se muestra en la Fig. 2.

El análisis de BN fue realizado mediante el software ELVIRA versión 0.162 en tres etapas sugeridas por Mediero (2007).

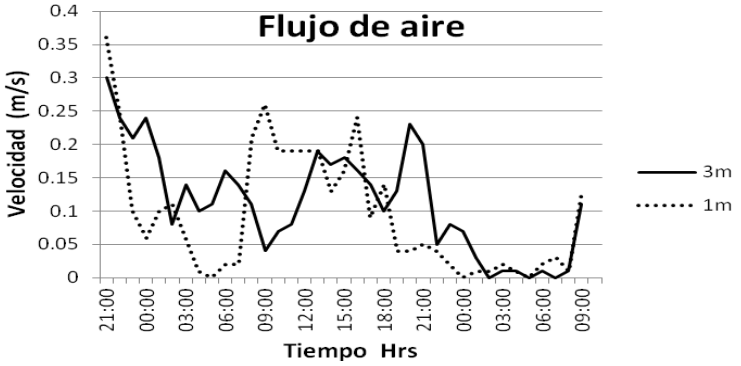
1. Preprocesamiento: Se llevó a cabo mediante el algoritmo de imputación "por promedios" para completar las series de datos parciales. Este algoritmo reemplaza los valores faltantes o desconocidos, por el promedio de los valores para cada variable. Este método no necesita parámetros y consiste en la discretización de los datos masivos mediante el algoritmo, empleando dos intervalos con la misma frecuencia.
2. Procesamiento: De acuerdo con Wang *et al*. (2006), la mejor estructura de red bayesiana se obtiene empleando el algoritmo K2 con un número máximo de padres igual a 3 y sin restricciones.
3. Postprocesamiento: Se realizó un análisis de dependencias para obtener la estructura topológica de la red, la cual representa a las variables y sus dependencias causales. Después de obtener la red de aprendizaje paramétrico, se realizó el cálculo de las probabilidades condicionales en las variables que muestren relación o dependencia.

Posteriormente, el conjunto de datos fue analizado en intervalos de 3 horas, para desarrollar una BN de tiempo discreto. Para poder validar el modelo, se empleó un

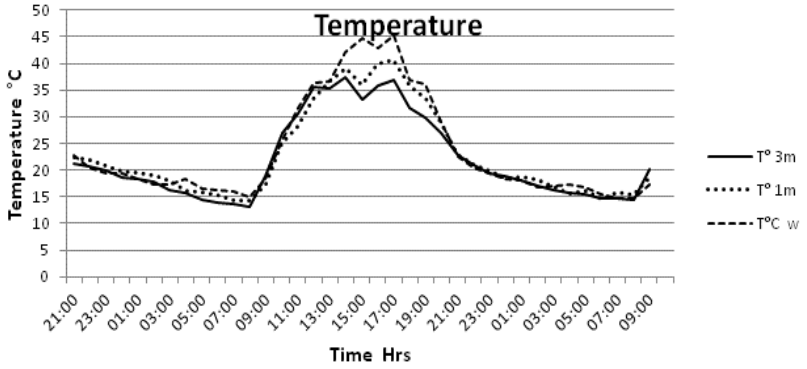
conjunto de datos diferentes al utilizado en la etapa de aprendizaje; las primeras mediciones fueron realizadas del 22 al 25 de agosto y correspondieron a las "Datos observados", el Segundo conjunto de datos fue medido entre el 18 y 21 de agosto y correspondieron a los "Datos esperados". El objetivo de esta prueba fue comprobar la BN para obtener una mejor solución, comparando las distribuciones de probabilidad.



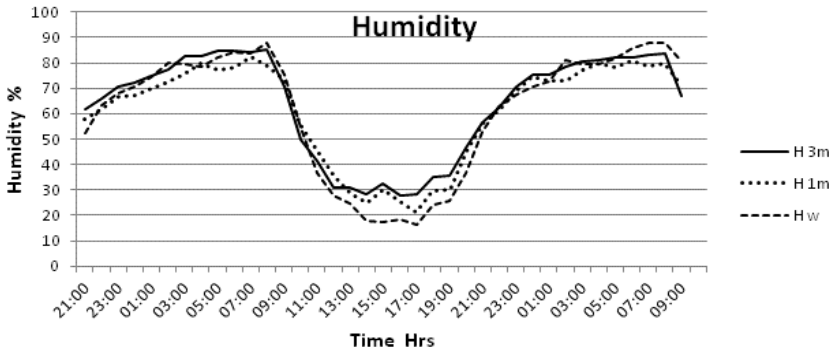

A) Concentración de $CO_2$ (ppm).




B) Velocidad del flujo de aire (m/s).




C) Temperatura media (°C).




D) Humedad relativa (%) a tres metros de altura (3m), un metro de altura (1m) y en los pasillos sin cultivo (w).

**Fig. 2.** Condiciones climáticas en la parte central del invernadero.

## 4. Resultados y discusión

Se obtuvo un modelo de BN con un 87.5% de precisión, calculado en la etapa de post-aprendizaje mediante el software ELVIRA, el cual muestra las relaciones entre las variables estudiadas. La Tabla 1. Describe los "Datos esperados" como los estados más probables de las variables. La validación del modelo de BN se muestra en la Tabla 2.

**Tabla 1.** Distribución de la probabilidad condicional entre $CO_2$ sobre la temperatura del aire a 3 m, temperatura del aire a 3 m sobre la humedad relativa a 1 m, y $CO_2$ a 3 m sobre $CO_2$ a 1 m.

| ***$CO_2$ 3m \ Tº3m*** | ***13º C*** | ***25º C*** | ***37º C*** | ***Tº 3m \ H 1m*** | ***26%*** | ***53%*** | ***80%*** | ***$CO_2$ 3m\$CO_2$ 1m*** | ***245 ppm*** | ***323 ppm*** | ***400 ppm*** |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 300 ppm | 0.067 | 0.133 | 0.75 | 13º C | 0.07 | 0.07 | 0.813 | 300 ppm | 0.866 | 0.067 | 0.0625 |
| 375 ppm | 0.067 | 0.734 | 0.188 | 25º C | 0.07 | 0.8 | 0.125 | 375 ppm | 0.067 | 0.866 | 0.0625 |
| 450 ppm | 0.867 | 0.133 | 0.063 | 37º C | 0.87 | 0.13 | 0.063 | 450 ppm | 0.067 | 0.067 | 0.875 |

**Tabla 2.** Validación del modelo de BN

| Relationship | Observed data | Expected data |
|---|---|---|
| $CO_2$ 3m\Tº 3m | r = -0.999 | r = -0.999 |
| Tº 3m\H 1m | r = -0.907 | r = -0.906 |
| $CO_2$ 3m\$CO_2$ 1m | r = 0.999 | r = 0.986 |

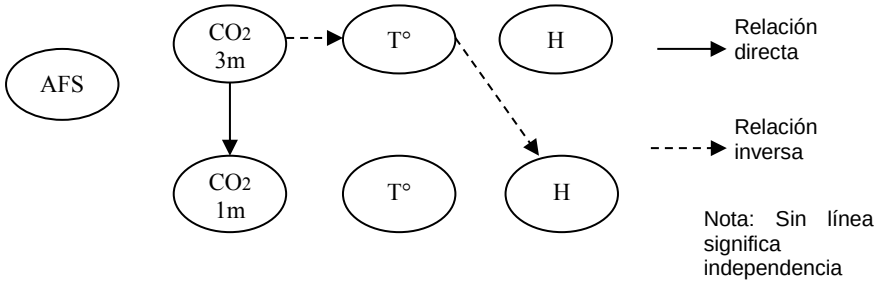



**Fig. 3.** Modelo de BN para un conjunto de datos de 36 horas en la parte central del invernadero: Velocidad del flujo de aire a 3m (AFS 3m), concentración de $CO_2$ a 1m ($CO_2$ 1m), concentración de $CO_2$ a 3m ($CO_2$ 3m), Temperatura a 3m (TºC 3m), Temperatura a 1m (TºC 1m), Humedad relativa a 3m (H 3m), y Humedad relativa a 1m (H 1m).

En la Figura 3 se muestran las relaciones entre variables considerando tanto el día cómo la noche, por lo que es importante establecer éste modelo de forma parcial, ya que las relaciones entre variables no son las mismas en el día y en la noche, haciéndose necesario un análisis más detallado en intervalos de tiempo menores. Este modelo muestra que cuando la velocidad del aire es menor a 0.4 m/s no afecta a otras variables. La concentración de $CO_2$ a 1 m y 3 m están directamente relacionadas. La concentración de $CO_2$ a 3 m y la Humedad Relativa a 1 m son inversamente proporcionales a la temperatura a 3 m.

Este análisis mostró que la velocidad de flujo de aire no afecta las otras variables cuando la velocidad del aire es baja, ya que no promueve un intercambio de calor. Esto coincide con estudios previos de Rico-García *et al*. (2008), Majdoubi *et al*. (2009) y Chow y Hold (2010). Mientras que las concentraciones de $CO_2$ situados a 1 m y 3 m son directamente proporcionales, el $CO_2$ a 3 m es inversamente proporcional a la temperatura del aire a 3 m, y del mismo modo la Humedad a 1 m.

Las concentraciones de $CO_2$ a 1 y 3 m están estrechamente relacionadas, tanto en la noche como en el día, el $CO_2$ a 1 m funciona como una fuente de $CO_2$ que incrementa su valor a 3 m elevándose rápidamente por encima del cultivo por diferencia de densidad. Durante el día, el cultivo consume $CO_2$ inicialmente a 1 metro, y el incremento en la temperatura hace que el $CO_2$ se vuelva menos denso y pasa a las capas superiores. No sucede lo mismo con la temperatura y la humedad, ya que tienen un comportamiento variable en el día y por la noche sobre 1 y 3 m, con una temperatura más alta en el día y una menor humedad en 1 m que a 3 m. Por la noche, estas relaciones se invierten, como se muestra en la Figura 4 (19:00 - 07:00). La relación inversamente proporcional del $CO_2$ y la humedad con la temperatura a 3 m muestra que esta variable es la de mayor importancia, ya que su variación provoca cambios en otras variables tanto de día como de noche. Siendo la humedad a 1 m la variable más sensible.

07:00-10:00. La temperatura del aire a 3 m aumenta influyendo sobre la zona de cultivo, sobre la humedad del aire a 3 metros y la concentración de $CO_2$ a 1 m, las cuales disminuyen cuando la temperatura del aire aumenta y calienta las capas inferiores a 1 m (T º C_BD ).

10:00-13:00. La concentración de $CO_2$ a 1 m (CO2_B) disminuye debido al aumento en la temperatura del aire y la actividad fotosintética. La humedad del aire a

3 m muestra una relación directa con la humedad a 1 m, que disminuye por el incremento de la temperatura. La concentración de $CO_2$ a 3 m disminuye su valor, pero no bajo la influencia de la temperatura dentro del invernadero, posiblemente debido a la ventilación.

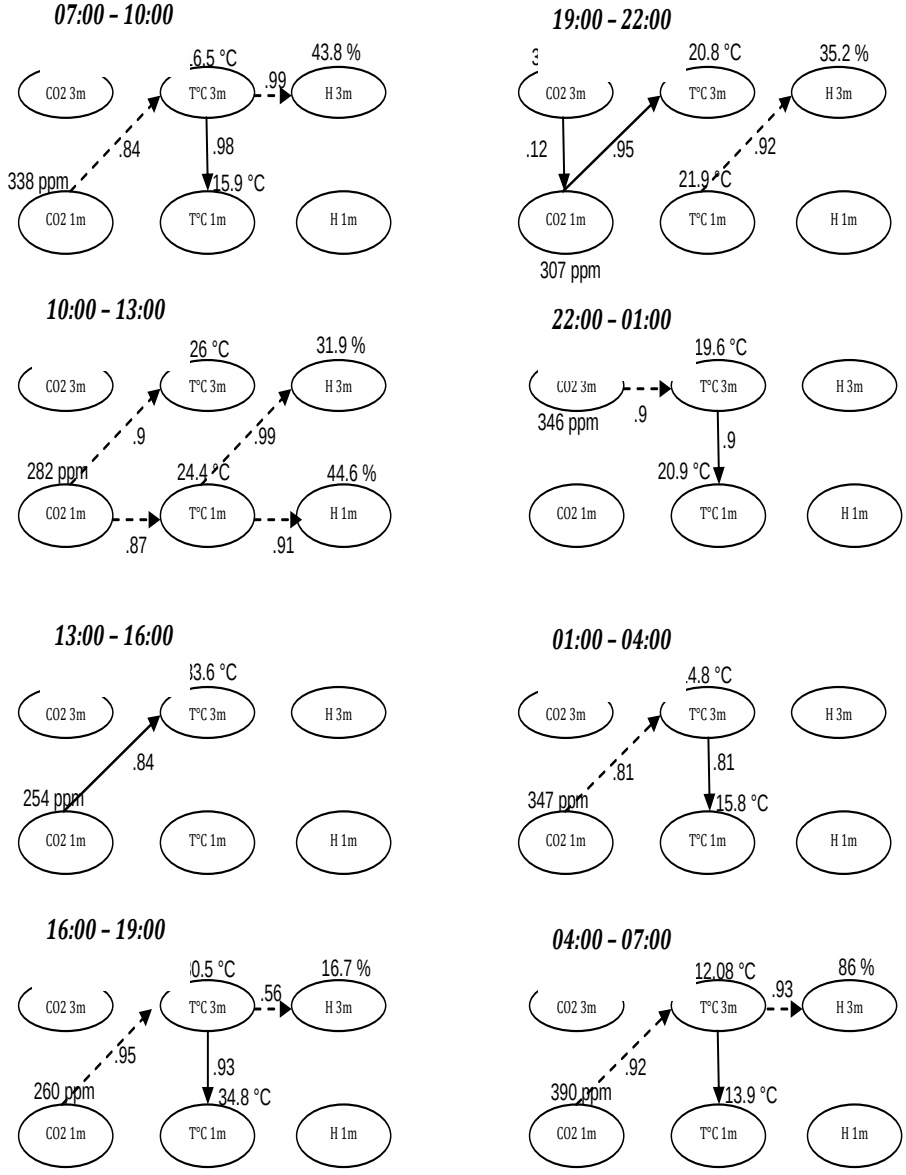



**Fig. 4.** Red Bayesiana para periodos de tres horas en un día. Concentración de CO2 a 3 m (CO2_A), concentración de CO2 a 1 m (CO2_B), temperatura del aire a 3 m (TºC_AD), temperatura del aire a 1 m (TºC_BD), humedad relativa a 3 m (H_AD), humedad a 1 m (H_BD). La línea continua indica una relación directa y la línea punteada indica una relación inversa.

13:00-16:00. La temperatura máxima y valores mínimos de humedad del aire y el $CO_2$ se alcanzó en el interior del invernadero en este período. La concentración de $CO_2$ a 1 m (CO2_B) muestra influencia directa con la temperatura del aire de 3 m (TºC_AD) lo cual es indicativo de la suspensión en la fotosíntesis por sobre calentamiento.

16:00-19:00. Baja la temperatura del aire y la fotosíntesis de la plantas se expresa una vez más en la relación inversa entre el $CO_2$ a 1 metro (CO2_B) y la temperatura del aire de 3 m (TºC_AD). De forma inversa al período entre las 7:00 a 10:00, cuando la temperatura del aire a 3 m disminuye, la temperatura del aire a 1 m (T C_BD º) también reduce sus valores desde este punto hasta el amanecer del día siguiente. La humedad del aire comienza a incrementarse.

19:00-22:00. El sol se oculta y las plantas paran la fotosíntesis, lo cual se expresa mediante la relación inversa entre la temperatura del aire a 3 m (TºC_AD), y el $CO_2$ a 1m (CO2_B) el cual se incrementa por la respiración de las plantas, aumenta la concentración de CO2 a 3 m (CO2-A). La temperatura del aire a 1 m (TºC_B) es mayor que a 3 m (TºC_A), y por lo tanto disminuye, presentando relación inversa con la humedad del aire a 3 m (H_AD), la cual aumenta.

22:00-01:00. La tendencia es similar al período anterior, sin embargo la concentración de CO2 en 1 m muestra la relación inversa con la temperatura del aire en 3 m debido a la respiración.

01:00-04:00. El $CO_2$ a 1 m muestra la relación inversa con la temperatura del aire en 3 m debido a la respiración.

04:00-07:00. En este punto, se han logrado mayores niveles de concentración de $CO_2$, la humedad del aire y disminuye la temperatura del aire. Al final de este período se tiene la mayor diferencia en la humedad del aire a los 3 m y a 1 m. Se registra la temperatura más baja a 3 m, mostrando relación inversa con el mayor nivel de humedad a 1 m, lo que sugiere que la humedad sube por efecto de la respiración del cultivo.

## 5. Conclusión

Utilizando un modelo de BN es posible observar y cuantificar las relaciones entre las variables Temperatura, Humedad relativa, velocidad del flujo de aire y la concentración de $CO_2$. Modelo de BN muestra que la Velocidad de flujo de aire no afecta a las otras variables cuando la velocidad del aire es baja. Los modelos de BN en tiempo discreto muestran las relaciones entre las variables que sugieren los procesos fisiológicos del cultivo y la interacción con su entorno. El estado de un proceso fisiológico en el cultivo está representado por un valor de probabilidad condicional en un intervalo de tiempo determinado y éste cambia a lo largo del día. Mediante un modelo de BN es posible conceptualizar el espacio de cultivo como un subsistema diferente a los pasillos y en el área por encima del cultivo, que interactúan con su ambiente dentro del invernadero. Las distribuciones de probabilidad condicionales son una medida cuantitativa de las relaciones entre las variables y muestran el estado más probable de estas variables. Los modelos de BN tienen la

capacidad de mostrar las relaciones entre las variables que involucran procesos fisiológicos de los cultivos en el interior de un invernadero.

**Agradecimientos.** El último autor es el autor para correspondencia. Este trabajo fue parcialmente financiado por el Consejo Nacional de Ciencia y Tecnología (CONACyT), la Universidad Autónoma de Querétaro y la Universidad Tecnológica de Corregidora.

## Referencias

1. Bournet, P.E., Boulard, T.: Effect of ventilator configuration on the distributed climate of greenhouses: A review of experimental and CFD studies. Comput Electron Agric, 74, 195 (2010)
2. Campen, J. B.: Vapor removal from the greenhouse using forced ventilation when applying a thermal screen. Acta Hortic, 801, 863 (2008)
3. Correa, M., Bielza, C., Paimes-Teixeira, J., Alique, J. R.: Comparison of Bayesian networks and artificial neural networks for quality detection in a machining process. Expert Syst Appl, 36(3), 7270 (2009)
4. Chow, K., Hold, A.E.: On the influence of boundary conditions and thermal radiation on predictive accuracy in numerical simulations of indoor ventilation. Building and Environment, 45, 437 (2010)
5. De la Torre-Gea, G., Soto-Zarazúa, G.M., Guevara-González, R., Rico-García, E.: Bayesian Networks for defining relations-hips among climate factors. IJPS, 6(18) 4412 (2011)
6. De la Torre-Gea, G., Soto-Zarazúa, G.M., López-Crúz, I., Torres-Pacheco, I., Rico-García, E.: Computational fluid dynamics in greenhouses: A review. AJB, 10(77), 17651 (2011)
7. Gámez, J.A., Mateo, J.L., Puerta, J.M.: Learning Bayesian networks by hill climbing: efficient methods based on progressive restriction of the neighborhood. Data Min. Knowl. Discov, 22, 106 (2011)
8. Garrote, L., Molina, M., Mediero, L.: Probabilistic Forecasts Using Bayesian Networks Calibrated with Deterministic Rainfall-Runoff Models. In: Vasiliev *et al*., Extreme Hydrological Events: New Concepts for Security. Springer, 173 (2007)
9. Guoliang, L.: Knowledge Discovery with Bayesian Networks, Ph. D. thesis, National University of Singapore, Singapore (2009)
10. Hruschka, E., Ebecken, N.F.F.: Bayesian networks for imputation in classification Problems. J Intell Inform Syst, 29, 231 (2007)
11. Khaoua, S.A.O., Bournet, P.E., Migeon, C., Boulard, T., Chassériaux, G.: Analysis of greenhouse ventilation efficiency based on computational fluid dynamics. Biosystems Eng, 95, 83 (2006)
12. Kim, K., Yoona, J.Y., Kwonb, H.J., Hanna, J.H., Sonc, J.E., Namd, S.W., Giacomelli, G.A., Lee, I.B.: 3-D CFD analysis of relative humidity distribution in greenhouse with a fog cooling system and refrigerative dehumidifiers. Biosystems Eng, 100, 245 (2008)
13. Majdoubi, H., Boulard, T., Fatnassi, H., Bouirden, L.: Airflow and microclimate patterns in a one-hectare canary type greenhouse: an experimental and CFD assisted study, Agr. Forest Meteorol, 149, 1050 (2009)

# Propuesta de construcción dinámica de espacios de búsqueda

Víctor Tomás Tomás-Mariano, Felipe de Jesús Núñez-Cárdenas, Efraín Andrade-Hernández

Escuela Superior de Huejutla, Universidad Autónoma del Estado de Hidalgo, Huejutla de Reyes, Hidalgo, México

{victor_tomasm, felipe.huejutla, andrade_a_h}@hotmail.com

**Resumen.** Se realiza el análisis de algoritmos para construir espacios de búsqueda tipo rejilla —laberintos— y su relación con grafos, los algoritmos analizados son: *Kruscal, Aldous-Broder, Prim* y *Recursivo Bactracker*. También se describe una propuesta para generar los gráficos por computadora. En el proceso de construcción se genera un grafo que permite aplicar algoritmos de búsqueda; así poder encontrar el camino entre dos nodos del grafo.

**Palabras clave:** Grafico por computadora, algoritmos de búsqueda, grafos.

## 1. Introducción

La construcción de un laberinto puede ser tan complicada como el resolverlo, los laberintos a realizar son de tipo cuadriculado o rejilla, éste debe ser perfecto y completamente conectado, es decir, se debe poder elegir cualquier punto del laberinto como entrada y cualquier otro punto como salida [1].

Un laberinto es conectado si hay un camino de cualquier celda a cualquier otra celda. Se desea que todos los laberintos sean conectados con el propósito de que cualquier celda pueda ser designada como inicio o salida, para poder encontrar una solución. Por consiguiente, se necesita poder garantizar que un laberinto sea conectado y que los algoritmos de construcción cumplan esta condición. En este proyecto se centra en explicar el comportamiento de los algoritmos de construcción de espacios de búsqueda –laberintos- y se recomienda una serie de algoritmos que se pueden aplicar en el medio de búsqueda para hallar una ruta entre dos vértices [2].

## 2. Construcción de laberintos de conexión simple

La construcción al azar permite generar gran cantidad de laberintos en poco tiempo y de forma sencilla, conteniendo corredores como las ramas de un árbol, persiguiendo la desorientación del explorador, siendo a su vez ésta la forma más fácil de

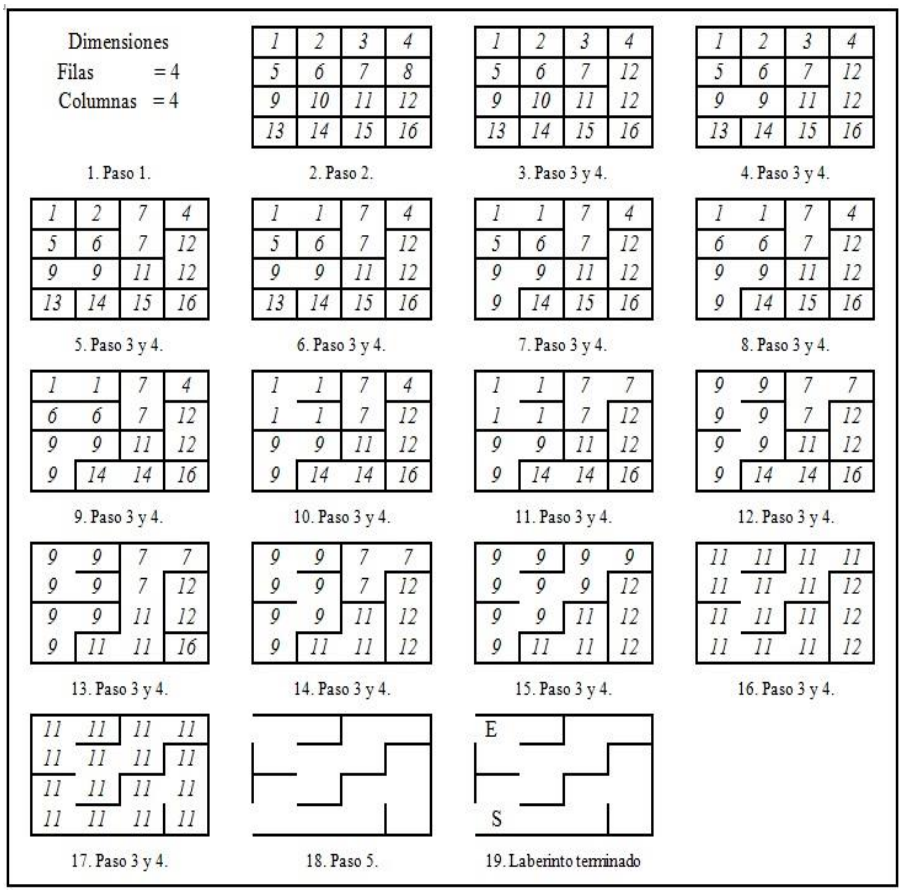



**Fig. 1.** Construcción de un Laberinto con el Algoritmo de Kruskal.

implementarse en un programa de computadora [1, 2, 3]. Los algoritmos explicados a continuación, generan laberintos completamente conectados.

La construcción al azar se basa en el enfoque que se llama "**Cavar túneles".** Como su nombre lo indica, se refiere a cavar túneles a lo largo y ancho del espacio de construcción hasta cubrirlo en su totalidad, como la explotación de una mina. A este tipo de construcción, pertenecen:

- Algoritmo de construcción Kruscal.
- Algoritmo de construcción Aldous-Broder.
- Algoritmo de construcción Prim's.
- Algoritmo de construcción Recursivo Bactracker.

**Algoritmo de Construcción Kruskal.** Este algoritmo genera laberintos de conexión simple (LCS). La construcción Kruskal genera laberintos simplemente conectados, este algoritmo cava túneles en varias secciones del laberinto, los pasos se enumeran a continuación, ver Figura 1:

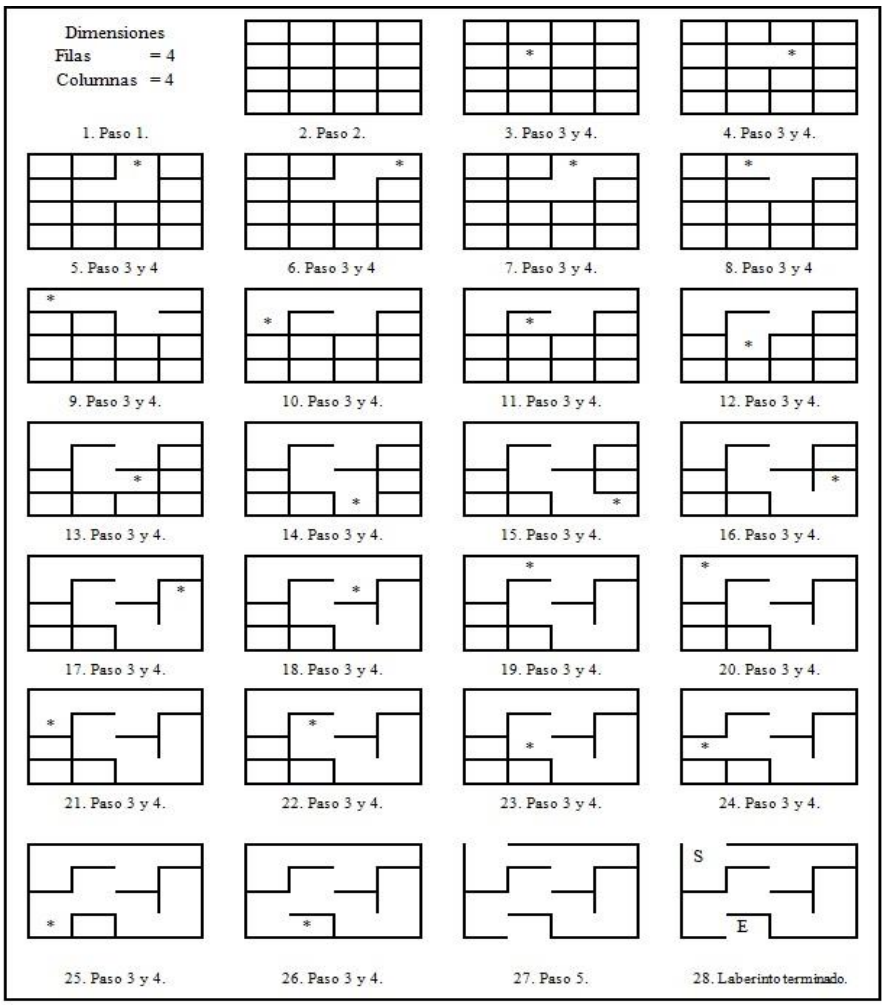



**Fig. 2.** Construcción de un Laberinto con el Algoritmo de Aldous-Broder.

1. Especificar el tamaño o las dimensiones del laberinto de acuerdo con las habitaciones que se desea que contenga.
2. Dividir el laberinto en habitaciones en base a sus dimensiones y etiquetarlas con una identificación única.
3. Seleccionar una habitación al azar y cavar un túnel hacia una habitación adyacente (en caso de existir más de una, elegir al azar) sólo si tiene diferente etiqueta, y etiquetar la habitación o habitaciones que se unieron con la identificación de la habitación inicial; repitiendo este proceso hasta que todas las habitaciones tengan la misma etiqueta.

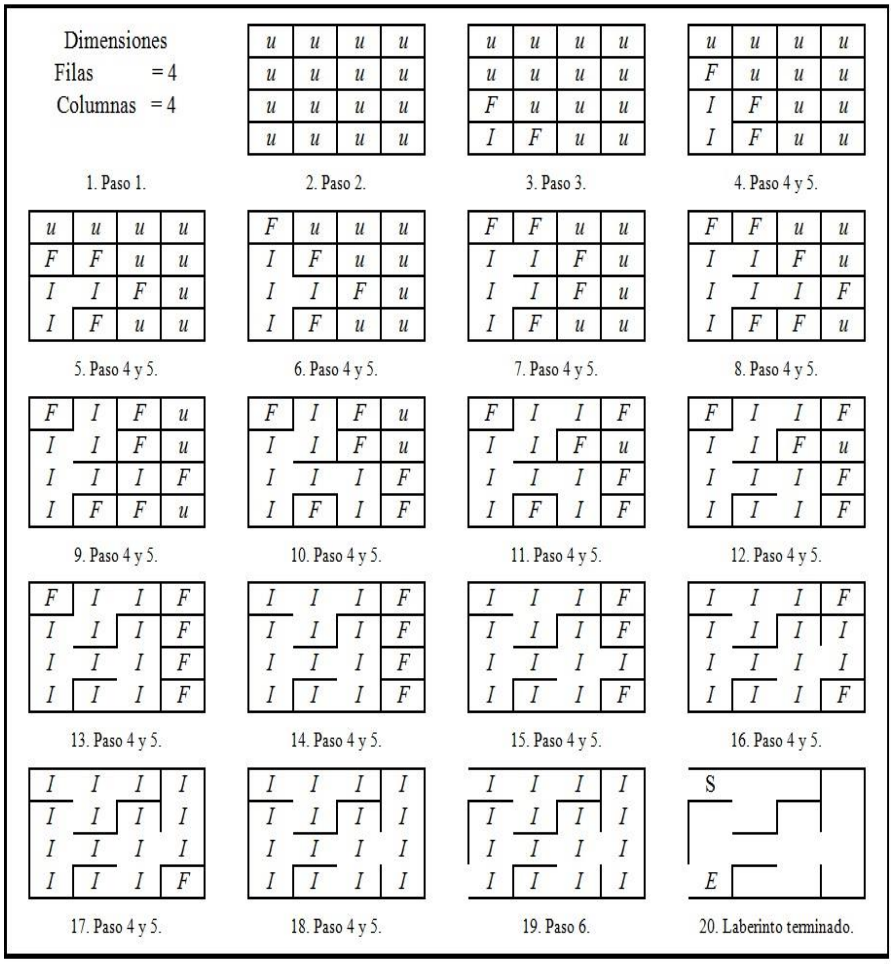



**Fig. 3.** Construcción de un Laberinto con el Algoritmo de Prim's.

4. Marcar la ubicación de la entrada o del punto inicial de la exploración y la ubicación de la salida.

**Algoritmo de Construcción Aldous-Broder.** Este algoritmo genera laberintos de conexión simple.

Es una técnica simple, pero puede llevar gran tiempo la construcción de un laberinto, es porque se mueve al azar dentro del laberinto sin discriminar las habitaciones en las que ya se cavaron túneles. Los pasos se enuncian a continuación, ver la Figura 2:

1. Especificar el tamaño o las dimensiones del laberinto de acuerdo con las habitaciones que se desea que contenga.
2. Dividir el laberinto en habitaciones en base a sus dimensiones.

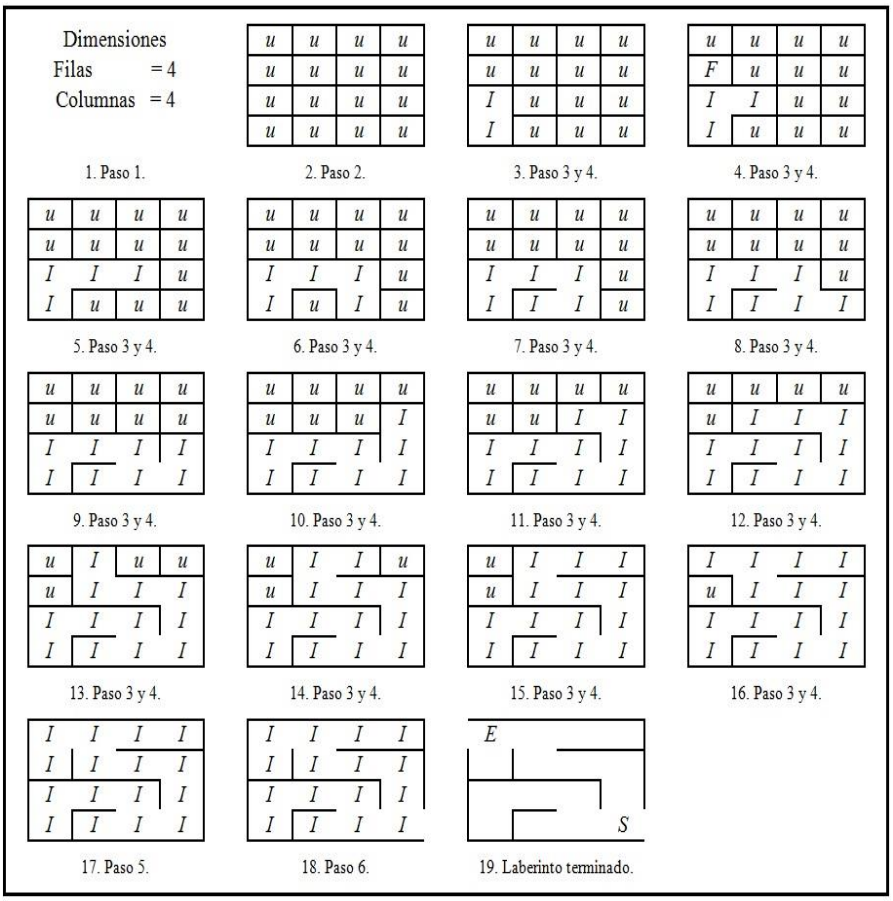



**Fig. 4.** Construcción de un Laberinto con el Algoritmo Recursivo Backtracker.

3. Elegir al azar una habitación dentro del laberinto.
4. Moverse a una habitación vecina adyacente al azar y cavar un túnel hacia la habitación anterior en el caso de que la nueva habitación no sea parte de otro túnel, continuando con este paso hasta que todas las habitaciones sean parte del laberinto.
5. Marcar la ubicación de la entrada o del punto inicial de la exploración y la ubicación de la salida.

**Algoritmo de Construcción Prim's.** Este algoritmo genera laberintos simplemente conectados, el procedimiento es el siguiente (ver la Figura 3):

1. Considera dos tipos de habitaciones:
2. Habitaciones etiquetadas con I son aquellas que forman parte del laberinto y han sido cavadas.

3. Habitaciones etiquetadas con F que no han sido cavadas, pero que son adyacentes a una habitación I.
4. Especificar el tamaño o las dimensiones del laberinto de acuerdo con las habitaciones que se desea que contenga.
5. Dividir el laberinto en habitaciones en base a sus dimensiones y etiquetarlas con una identificación única.
6. Seleccionar una habitación inicial, etiquetarla como I y etiquete sus habitaciones adyacentes con F.
7. Seleccione una habitación F aleatoriamente. Cavar un muro de la habitación I a la habitación F; cámbielo a I y cambie sus habitaciones adyacentes que no son I a F.
8. Repita este proceso mientras haya habitación F.
9. Marcar la ubicación de la entrada o del punto inicial de la exploración y la ubicación de la salida.

**Algoritmo de Construcción Recursivo Backtracker.** Este algoritmo genera laberintos de conexión simple. El procedimiento es el siguiente (ver la Figura 4):

1. Especificar el tamaño o las dimensiones del laberinto de acuerdo con las habitaciones que se desea que contenga.
2. Dividir el laberinto en habitaciones en base a sus dimensiones y etiquetarlas con una identificación única
3. Elegir al azar una habitación dentro del laberinto y empiece a cavar. Siempre cave hacia habitaciones adyacentes que no han sido cavadas, si las hay, si no, regrese a buscar uno, por el túnel ya cavado, hasta encontrar una.
4. Repita este procedimiento, mientras no se regrese al punto inicial.
5. Marcar la ubicación de la entrada o del punto inicial de la exploración y la ubicación de la salida.

Los algoritmos anteriores generan laberintos de conexión simple (LCS), se pueden construir laberintos de conexión múltiple (LCM) –laberintos con circuitos internos, "cavando" paredes en el interior del laberinto en forma aleatoria o en donde se formen callejones sin salida.

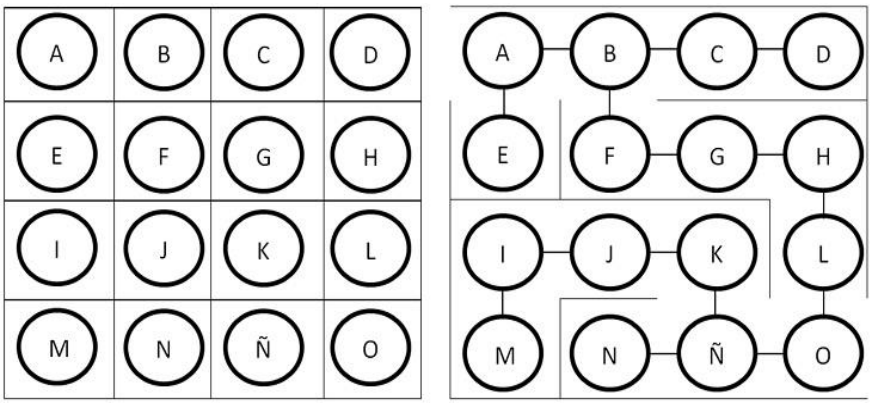
Los LCM, a diferencia de los LCS, que solo tienen una ruta como solución, estos laberintos, tienen varias "rutas solución", además tienen islas o circuitos internos, no tienen puntos muertos. Esto, dificulta encontrar la ruta entre dos puntos dentro del laberinto, pues se puede dar giros en un mismo segmento en forma indefinida. Por lo que de alguna manera hay que llevar el control de los lugares ya visitados. La propuesta planteada permite manipular este tipo de laberintos, al crear un grafo con circuitos internos.

## 3. Propuesta de construcción y graficación

En la construcción de un laberinto se manejan 2 matrices:

### 3.1. Matriz de vértices y aristas

Inicialmente esta matriz, se conforma por el número de vértices con base en las dimensiones que se desee tenga el laberinto. El grafo es completamente desconectado, tal como se ve en la figura 5a. Conforme se aplican los pasos de construcción, por ejemplo, el mostrado en la figura 4, se crea la arista entre los vértices respectivos y al final de la construcción, se obtiene un grafo completamente conectado, ver figura 5b.




a) b)

**Fig. 5.** Construcción laberinto a) grafo desconectado, b) grafo conectado asociado a un laberinto.

En la figura 5, se utilizan símbolos alfabéticos para los vértices, sin embargo, se puede utilizar símbolos numéricos para tener un mayor margen de amplitud en el número de vértices manejados.

### 3.2. Matriz de coordenadas

En esta matriz se representan las coordenadas del espacio de graficación. Se hace un mapeo en pixeles de las dimensiones de la matriz del laberinto, en la figura 6 se muestran las coordenadas para un laberinto de 4 filas, 4 columnas, y de las posiciones [i, j] respectivamente. Los movimientos permitidos en la construcción son: arriba, abajo, izquierda, derecha. Se propone el cálculo de coordenadas de graficación con base en la posición [i, j] de partida en el movimiento respectivo. Inicialmente, se traza el cuerpo del laberinto, extremadamente desconectado y se van cavando túneles a lo largo del proceso de construcción**.**

Movimientos permitidos, cavar túneles:

**Arriba: [i, j] a [i-1, j]**

Posición [i, j] origen: [2, 2]; posición [i, j] destino [1, 2]

Formula: [x0+ j*dx, y0+i*dy] , [x0+(j+1)*dx, y0+i*dy]

Posición origen: [i=2, j=2]:

Se obtiene: [0+2*10, 0+2*10], [0+3*10, 0+2*10] = [20, 20], [30, 20]

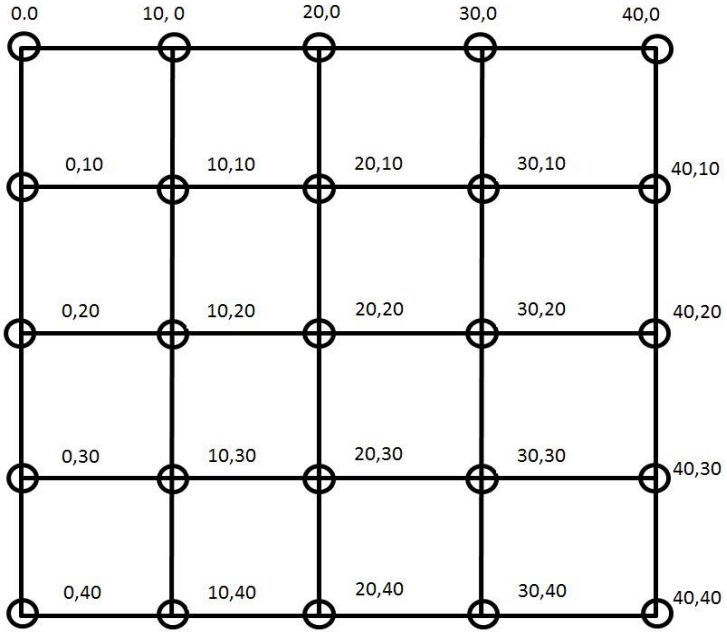



**Fig. 6.** Coordenadas del espacio de graficación.

Cavar_túnel (20,20, 30, 20)
Crear_vertice(nodoOri, nodoDes)
Comentario: incrementa en dx, constante en dy.

**Abajo: [i, j] a [i+1, j]**

Posición [i, j] origen:[2, 2]; posición[i, j] destino [3, 2]
Formula: [x0+j*dx, y0+(i+1)*dy], [x0+(j+1)*dx, y0+(i+1)*dy]
Posición origen: [i=2, j=2]:
Se obtiene: [0+2*10, 0+3*10] - [0+3*10, 0+3*10] = [20,30]-[30,30]
Cavar_túnel (20,30, 30, 30)
Crear_vertice(nodoOri, nodoDes)
Comentario: incrementa en dx, constante en dy.

**Izquierda: [i, j] a [i, j-1]**

Posición [i, j] origen: [1, 1]; posición [i, j] destino [1,0]
Formula: [x0+j*dx, y0+i*dy] , [x0+j*dx, y0+(i+1)*dy]
Posición origen: [i=2, j=2]:
Se obtiene; [0+2*10, 0+2*10] – [0+2*10, 0+3*10] = [20,20]-[20,30]
Cavar_túnel (20,20, 20, 30)
Crear_vertice(nodoOri, nodoDes)
Comentario: constante en dx, incrementa en dy.

**Derecha: [i, j] a [i, j+1]**

Posición [i, j] origen: [1, 1]; posición [i, j] destino [1,2]
Formula: [x0+(j+1)*dx, y0+i*dy]; [x0+(j+1)*dx, y0+(i+1)*dy]
Posición origen: [i=2, j=2]:

Se obtiene; [0+3*10, 0+2*10] – [0+3*10, 0+3*10] = [30,20]-[30,30]
Cavar_túnel (30,20, 30, 30)
Crear_vertice(nodoOri, nodoDes)
Comentario: constante en dx, incrementa en dy.

Los movimientos y valores de coordenadas obtenidas se pueden verificar en la figura 6, para el este ejemplo, se tiene un área limitada para el caso de 4 filas y 4 columnas.

Se inicia la graficación en las coordenadas iniciales x0=0, y0=0, con incrementos en dx y dy que representan la distancia entre celdas horizontal y vertical respectivamente, en este caso se maneja un valor igual a 10. Los valores relativos a las posiciones [i, j] dentro del arreglo dependen de la celda que se esté procesando durante el proceso de construcción.

Para un espacio de graficación de n * m, se obtiene un conjunto de coordenadas tal como se muestra en la figura 7:

Sean ***n*** filas en dx: 1, 2, 3…n.
Sean ***m*** columnas en dy: 1, 2, 3…m

| | | | | | |
|---|---|---|---|---|---|
| [x0],[y0] | [x0+1dx], [y0] | [x0+2dx], [y0] | [x0+3dx], [y0] | … | [x0+n*dx], [y0] |
| [x0],[y0+1dy] | [x0+1dx],[y0+1dy] | [x0+2dx],[y0+1dy] | [x0+3dx],[y0+1dy] | | |
| [x0],[y0+2dy] | [x0+1dx],[y0+2dy] | [x0+2dx],[y0+2dy] | [x0+3dx],[y0+2dy] | | |
| [x0],[y0+3dy] | [x0+1dx],[y0+3dy] | [x0+2dx],[y0+3dy] | [x0+3dx],[y0+3dy] | … | |
| ….. | | | | | |
| [x0],[y0+mdy] | | | | | [x0+n*dx],[y0+m*dy] |

**Fig. 7.** Matiz de coordenadas para un laberinto de *m filas* por *n columnas*.

## 4. Resultados

En la representación de la figura 6, el grosor de las líneas del laberinto es de un píxel, sin embargo, la propuesta realizada permite manipular también el "grosor" de las líneas, algunos resultados obtenidos se visualizan en la figura 8, en esta figura se muestra un laberinto con dimensiones de 50 filas por 50 columnas y grosor de 1 pixel, figura 8 a); laberinto con dimensiones de 50 filas por 50 columnas y grosor de 4 pixeles, figura 8 b); laberinto con dimensiones de 50 filas por 50 columnas y grosor de 10 pixeles, figura 8 c).

De los resultados obtenidos en las graficas con grosor de un píxel, se observa que no son perceptibles a simple vista, por lo que resultan difíciles de explorar en la interfaz de una computadora, sin embargo, se obtienen mejores visualizaciones graficas al incrementar el grosor de las líneas a partir de 4 píxeles o mayor, otro factor que influye es el número de filas y columnas del espacio de búsqueda, es decir, en laberintos pequeños resulta fácil explorarlos, sin embargo, el grado de dificultad incrementa conforme se aumenta el numero de celdas que se desee tenga el espacio de búsqueda.

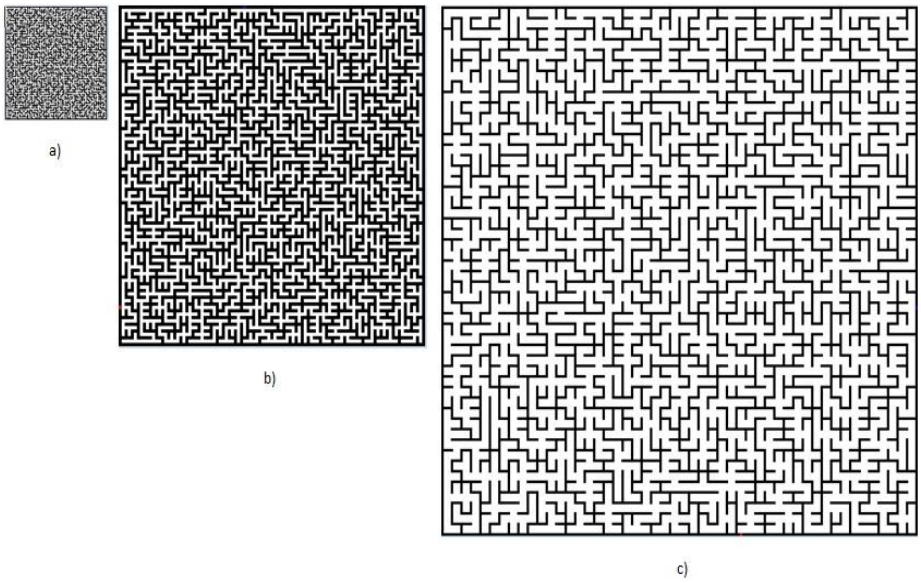



**Fig. 8.** Grosor líneas laberintos. a) Dimensiones: 50f, 50c, 1p; b) Dimensiones: 50f, 50c, 4p; c) Dimensiones: 50f, 50c, 10p.

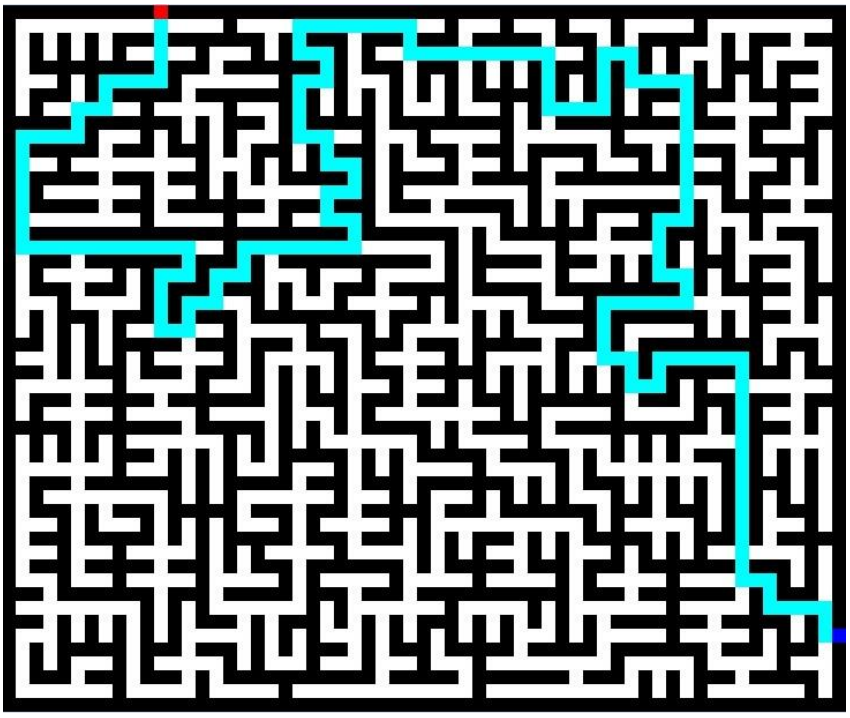


**Fig. 9.** Laberinto con solución. Dimensiones: 25f, 30c, 15p.

La entrada del laberinto se simboliza con un cuadro en color rojo y la salida en color más claro.

## 5. Búsqueda de solución en espacio de estados

La propuesta de generación de un grafo en el proceso de construcción de un laberinto, permite modelar el problema para aplicar algoritmos de búsqueda en grafos sin pesos, con la finalidad de encontrar una ruta entre dos vértices; entre las técnicas de la inteligencia artificial (IA) [4, 5] que se pueden aplicar en este tipo de problemas están: *Primera Búsqueda en Profundidad, Primera Búsqueda en Amplitud, Dijkstra y el A estrella*, los dos primeros hacen una búsqueda a ciegas y se aplica en casos en donde el punto de salida es desconocido y en las aristas no hay pesos[6]. El algoritmo A estrella es otro de los más aplicados en técnicas de búsqueda, y se aplica cuando el punto de salida es conocido, su funcionamiento tiende a realizar movimientos en diagonal, horizontal y vertical, permite asignar un costo en los movimientos. Este algoritmo utiliza una búsqueda heurística para encontrar la ruta óptima entre dos puntos [5, 7]. Maneja tres funciones: F, G y H.

- La función G es el costo del mejor camino desde la celda inicial a la celda n obtenido hasta el momento durante la búsqueda.
- La función H es el costo del camino más corto desde la celda n a la celda objetivo más cercano n.
- F = G + H, Es decir, F es el costo del camino más corto desde la celda inicial a la celda objetivo.

El algoritmo de Dijkstra [4] permite encontrar la ruta más corta entre dos vértices de un grafo, en las aristas se pueden colocar pesos positivos que significan costos que implica ir de una vértice a otro vértice.

Una posible ruta entre la entrada y salida que se obtiene con el grafo generado se muestra en la figura 9.

## 6. Conclusiones

Se han analizado los diferentes algoritmos para la construcción de laberintos de conexión simple. El proceso de construir un laberinto involucra que el resultado final de la construcción debe ser un laberinto completamente conectado, los algoritmos analizados construyen laberintos completamente conectados.

La técnica propuesta genera un grafo completamente conectado en un espacio de estados, y a partir del grafo, hacer una búsqueda de una solución que conecte a dos vértices, se pueden aplicar algoritmos para recorrido de grafos. Con respecto a la graficación del espacio de búsqueda se generan gráficos dinámicos, se contempla las posiciones [i, j] para el cálculo de coordenadas de graficación y los cambios en función de los movimientos permitidos en el algoritmo de construcción.

## Referencias

1. Suits, David B.: Playing With Mazes. Department of Philosophy, Rochester Institute of Technology (1995)
2. Bernabe, S. E.: Construcción y Recorrido de Laberintos. Tesis de Licenciatura, Pachuca de Soto, Hidalgo, México (Julio 2004)
3. Depue, K.: Maze Complexity. Hastings College, Hastings, Nebraska, USA, B.A., Mathematics and Computer Science (May 2005)
4. Johnsonbaugh, R.: Matemáticas discretas. Pearson Educación (2005)
5. Marín, R., Palma, J.T.: Inteligencia Artificial: métodos, técnicas y aplicaciones (2008)
6. Molina, V. J., Torres, P. C., Restrepo, P. C.: Técnicas de inteligencia artificial para la solución de laberintos de estructura desconocida, Scientia et Technica UTP (2008)
7. Lester, P.: A* pathfinding for beginners. Almanac of Policy Issues: see http://www.policyalmanac.org/games/aStarTutorial.htm. (2005)

# Detección y diagnóstico de fallas para la dinámica lateral de un automóvil utilizando máquinas de soporte vectorial multiclase

Jesús Alejandro Navarro Acosta, Juan Pablo Nieto González

Corporación Mexicana de Investigación en Materiales, S.A. de C.V. (COMIMSA), México

{jesus.navarro, juan.nieto}@comimsa.com

**Resumen.** En la actualidad el correcto y eficiente monitoreo de sistemas altamente complejos como el automóvil, representa una tarea desafiante para la ingeniería en general. Los objetivos primordiales en un sistema de detección y diagnóstico de fallas para un vehículo son: evitar daño en el mismo y evitar situaciones de peligro para los tripulantes. Los principales retos para lograr un óptimo monitoreo de un automóvil se derivan de la fuerte correlación entre diversas variables, lo cual aumenta la probabilidad de generar falsas alarmas. Además de la existencia de incertidumbre inherente al sistema causada por mediciones con presencia de ruido. En este artículo se presenta un sistema de monitoreo y diagnóstico de la dinámica lateral de un vehículo. El mejor desempeño de la nueva propuesta se valida comparando los resultados obtenidos contra los de un sistema de monitoreo encontrado en la revisión del estado del arte y que combina tres técnicas para dar el diagnóstico final. Tales técnicas son escalamiento multidimensional (EMD), rangos de percentiles (RP) y análisis de componentes principales (ACP). Se presenta un caso de estudio en donde el sistema de diagnóstico se diseñó para el monitoreo de seis variables. Los datos se obtuvieron mediante el empleo del simulador VEHDYNA. Se muestra como los resultados obtenidos con la nueva propuesta son prometedores.

**Palabras clave:** Automóvil, SVM multiclase, dinámica lateral.

## 1. Introducción

Debido al constante avance de la tecnología, el sector industrial es capaz de elaborar productos cada vez más complejos. Por tal motivo, llevar a cabo un correcto y eficiente monitoreo para la detección y diagnóstico de fallas es un reto para la ingeniería actual. La detección y diagnóstico de fallas en sistemas de ingeniería está relacionada con la detección de fallas en máquinas complejas mediante aprendizaje de patrones específicos de comportamiento en los datos observados. Un vehículo moderno es un ejemplo de tal sistema de ingeniería complejo en el que hay un gran número de sensores, controladores y módulos informáticos

integrados en el vehículo, los cuales se encargan de recolectar un gran número de señales [1]. Dichas señales cuentan con características como rangos variables de magnitud, oscilación, frecuencia, etc. Por tal motivo, el procesamiento de estos datos es una tarea extremadamente difícil para los sistemas basados en técnicas de control clásico. Debido a esto, es importante el estudio e implementación de técnicas de inteligencia artificial dotadas de la flexibilidad necesaria para procesar distintos tipos de variables. Las cuales tengan como común denominador la presencia de ruido, correlación, características no lineales, etc. Las máquinas de soporte vectorial (MSV) con base en la teoría del aprendizaje estadístico y principio de minimización del riesgo estructural, se han aplicado en el ámbito del reconocimiento de fallas por su excelente capacidad de generalización [2]. En este artículo se presenta un sistema de detección y diagnóstico de fallas basado en máquinas de soporte vectorial multiclase (MSVMC) el cual requiere datos en modo de operación normal y datos en modo de falla para su entrenamiento. Su desempeño es comparado frente a un sistema de detección y diagnóstico de fallas propuesto por [1], el cual combina escalamiento multidimensional (EMD), rangos de percentiles (RP) y análisis de componentes principales (ACP). Los resultados se obtienen a partir de una simulación para un modelo de la dinámica lateral de un automóvil el cual consta de seis variables. La organización del artículo es la siguiente: la sección 2 presenta brevemente los principios de MSV, MSVMC, EMD, RP y ACP. La sección 3 presenta las descripciones generales de los sistemas de monitoreo a comparar. La sección 4 muestra el caso de estudio. La sección 5 muestra como ambos sistemas trabajan frente a una simulación para fallas individuales (fallas sobre una variable a la vez). La sección 6 presenta las conclusiones.

## 2. Técnicas

### 2.1. Maquinas de soporte vectorial

Las máquinas de soporte vectorial son un algoritmo de aprendizaje automático que se utiliza para la clasificación y la regresión de conjuntos de datos de alta dimensionalidad. Este ofrece grandes resultados debido a sus capacidades para lidiar con problemas como mínimo local, pocas muestras de datos y problemas no lineales. Las MSV estándar son utilizadas para problemas de clasificación binaria [3]. La idea central de esta técnica es determinar una separación lineal (hiperplano separador) el cual es orientado en dirección tal que su distancia a los puntos de datos más cercanos en cada una de las dos clases (margen) sea la máxima. Los puntos de datos más cercanos son conocidos como vectores de soporte [4].

**Caso linealmente separable.** Se tienen vectores de entrada $\mathbf{x}_i \in \mathbb{R}^d$ $(i = 1, 2, ..., n)$ correspondientes con las etiquetas $\mathbf{y}_i \in \{-1, +1\}$. Existe un plano separador cuya función es [5]

$$\mathbf{w} \cdot \mathbf{x} + b = 0 \tag{1}$$

Donde $\mathbf{w} \in \mathbb{R}^n$ es un vector normal que define la frontera, $b$ es un umbral escalar y $\frac{|b|}{\|\mathbf{w}\|}$ representa la distancia perpendicular desde el hiperplano separador hacia el origen. Mientras que la distancia del hiperplano al margen esta dada por $d = \frac{1}{\|\mathbf{w}\|}$. La Figura 1 muestra el hiperplano separador y el margen máximo entre clases (lineas punteadas).

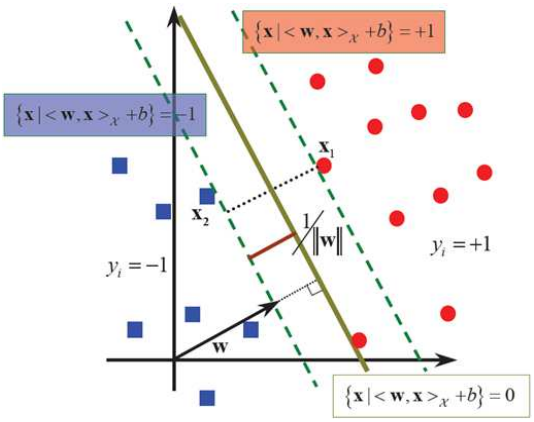



**Fig. 1.** Hiperplano y definición del margen geométrico, adaptado de [6].

Dos hiperplanos paralelos pueden ser representados como

$$\mathbf{y}_i \left(\mathbf{w} \cdot \mathbf{x}_i + b\right) \geq 1 \tag{2}$$

Como se definió anteriormente MSV trata de maximizar la distancia entre dos clases, donde la amplitud del margen entre dos hiperplanos paralelos es $d = \frac{2}{\|\mathbf{w}\|}$, por lo tanto para el caso linealmente separable se puede encontrar el hiperplano óptimo resolviendo el siguiente problema de optimización cuadrático

$$\begin{gathered} min\ \frac{1}{2} \|\mathbf{w}\|^2 \\ \text{sujeto a: } \mathbf{y}_i \left(\mathbf{w} \cdot \mathbf{x}_i + b\right) \geq 1 \end{gathered} \tag{3}$$

Utilizando multiplicadores de Lagrange $\alpha_i (i = 1, 2, ..., n)$ para la restricción, el problema primal (Ecuación (4)) se convierte en encontrar el punto de silla de Lagrange. Por lo tanto se vuelve un problema dual de la forma

$$max\ L(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{n}^{ij} \alpha_i \alpha_j \mathbf{y}_i \mathbf{y}_j$$
$$\text{sujeto a: } \sum_{i=1}^{n} \alpha_i \mathbf{y}_i = 0,\ \alpha_i \geq 0 \tag{4}$$

Aplicando las condiciones de Karush-Kukn-Tucker (KKT), se tiene

$$\alpha_i \left[\mathbf{y}_i \left(\mathbf{w} \cdot \mathbf{x}_i + b\right) - 1\right] = 0 \tag{5}$$

Si $\alpha_i > 0$, los puntos correspondientes a los datos son llamados vectores de soporte. Por lo tanto, la solución óptima para el vector normal está dada por

$$\mathbf{w}^* = \sum_{i=1}^{N} \alpha_i \mathbf{y}_i \mathbf{x}_i \tag{6}$$

Donde $N$ es el número de vectores de soporte. De la ecuación (5), eligiendo cualquier vector de soporte $(\mathbf{x}_k, \mathbf{y}_k)$, se obtiene $b^* = \mathbf{y}_k - \mathbf{w}^* \cdot \mathbf{x}_k$. Después $(\mathbf{w}^*, b^*)$ es determinado. A partir de la ecuación (6) el hiperplano separador óptimo puede escribirse como [7,8]

$$f(\mathbf{x}) = \mathbf{sgn}\left(\sum_{i=1}^{N} \alpha_i \mathbf{y}_i (\mathbf{x} \cdot \mathbf{x}_i) + b^*\right)$$
$$\text{donde sgn(.) es la función sign, y } \mathbf{x} \in \begin{cases} +1 \text{ si } f(\mathbf{x}) > 0 \\ -1 \text{ si } f(\mathbf{x}) < 0 \end{cases} \tag{7}$$

**Caso no linealmente separable.** Las MSV también pueden manejar casos donde los datos no son linealmente separables. Estas intentan mapear el vector de entrada $\mathbf{x}_i \in \mathbb{R}^d$ sobre un espacio de mayor dimensionalidad. Este proceso está basado en la elección de una función kernel. Algunas de las funciones kernel ($\mathbf{K}$) más usadas son [9]:

- Kernel lineal
- Kernel polinomial
- Función de base radial
- Kernel sigmoideo

Por lo tanto el hiperplano óptimo (Ec. (8)) toma la forma

$$f(\mathbf{x}) = \mathbf{sgn}\left(\sum_{i=1}^{N} \alpha_i \mathbf{y}_i \cdot \mathbf{K}(\mathbf{x} \cdot \mathbf{x}_i) + b^*\right) \tag{8}$$

La Figura 2 muestra un ejemplo de una transformación no lineal

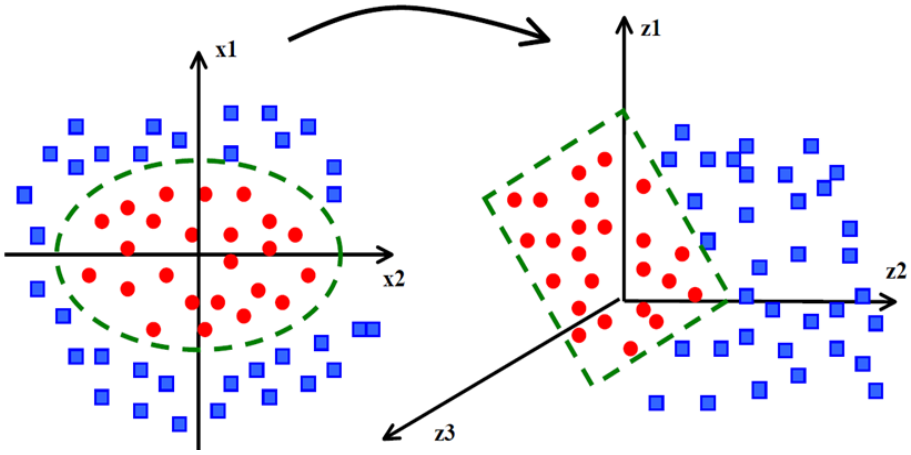



**Fig. 2.** Transformación no lineal, adaptada de [6].

### 2.2. Escalamiento multidimensional (EMD)

La técnica EMD es aplicada cuando para un conjunto de distancias observadas entre cada par de $N$ objetos, se desea encontrar una representación de los mismos en dimensiones menores, tal que las proximidades entre objetos coincidan lo más cerca posible con las originales [1].

Para $N$ objetos obtenemos

$$M = \frac{N(N-1)}{2} \tag{9}$$

Se ordenan las semejanzas $s_{i_1k_1} < s_{i_2k_2} < ... < s_{i_Mk_M}$ donde $s_{i_1k_1}$ es la más pequeña de las $M$ semejanzas. Ahora se calculan las distancias entre objetos $d^{(q)}_{i_1k_1} > d^{(q)}_{i_2k_2} > ... > d^{(q)}_{i_Mk_M}$. Por último se minimiza el *stress*

$$stress = \left[ \frac{\sum_{i<k} \sum \left( d^{(q)}_{ik} - \widehat{d}^{(q)}_{ik} \right)^2}{\sum_{i<k} \sum (d^{(q)}_{ik})^2} \right]^{\frac{1}{2}} \tag{10}$$

Usando las $\widehat{d}^{(q)}_{ik}$ mueve los puntos alrededor para obtener una mejor configuración, el proceso se repite hasta que la mejor representación es obtenida (*stress* mínimo).

### 2.3. Rangos percentiles

Los percentiles se aplican a datos numéricos para describir la dispersión en la muestra de datos. Los percentiles se basan en la división en 100 unidades.

El $P$-ésimo divide los datos de manera que existe un $P$ porcentaje de números por debajo del $P$-ésimo percentil y un porcentaje igual a $100 - P$ por encima del mismo. Los percentiles se calculan ordenando las observaciones en orden descendente, después calcule las posición:

$$L = \frac{P}{100} * N \tag{11}$$

donde $N$ es el número total de observaciones.

Rangos percentiles calculan la diferencia entre dos percentiles de una muestra $N$ y ya que estos brindan información acerca de la dispersión y localización de los datos, esta diferencia es una forma robusta de estimar la dispersión de los datos aun cuando estos no se distribuyan normalmente [1].

### 2.4. Análisis de componentes principales

Considere una matriz de datos $\mathbf{X}$ de tamaño $m \times n$, para $m$ variables del proceso con $n$ mediciones de cada variable. Asumimos que los datos para cada variable han sido escalados con media 0 y varianza 1. La matriz de covarianza de $\mathbf{X}$ esta definida como

$$\sum = \frac{\mathbf{X}^T \cdot \mathbf{X}}{n-1} \tag{12}$$

Sea $\lambda_i (i = 1, 2, \ldots, m)$ los valores propios de la matriz de covarianza, los cuales son organizados en orden descendente para determinar los componentes principales (CPs), y los componentes principales $\mathbf{p}_i$ (*loadings*) los cuales son los correspondientes vectores propios. Entonces, los primeros $K$ CPs son seleccionados para construir el modelo ACP, por lo que la matriz de datos $\mathbf{X}$ puede ser expandida usando los componentes principales $\mathbf{p}_i$, los vectores $\mathbf{t}_i$ (*scores*), y la matriz residual $\mathbf{E}$ [10]

$$\mathbf{X} = \sum_{i=1}^{K} \mathbf{t}_i \mathbf{p}_i + \mathbf{E} \tag{13}$$

Los $\mathbf{t}_i$ contienen información sobre como las muestras estan relacionadas entre si. Mientras los $\mathbf{p}_i$ contienen información sobre la relación existente entre las variables. El número de componentes principales debe ser igual o menor que las variables de $\mathbf{X}$ [1].

### 2.5. Matriz de confusión

En el área de la inteligencia artificial la matriz de confusión es una herramienta de visualización que se usa en el aprendizaje supervisado. Cada columna de la matriz representa el número de predicciones de cada clase, y cada fila representa cada clase real. La matriz de confusión facilita observar si el sistema confunde dos clases.

## 3. Descripción del sistema

Como se mencionó anteriormente el desempeño del sistema de monitoreo propuesto en este artículo será comparado con el sistema fuera de línea propuesto por [1]. A continuación se presenta una breve descripción del funcionamiento de dicho sistema.

El sistema puede ser clasificado como un método basado en datos históricos del proceso, este necesita conjuntos de datos del comportamiento del sistema en estado normal de operación. La etapa de extracción de características se lleva a cabo para que el sistema aprenda el estado normal de operación del proceso. Primero calcula ACP para observar la correlación entre variables. Esto agrupará las variables que estén correlacionadas y separará las que no lo estén. Esto puede revelar la causa raíz de la falla. Después lleva a cabo EMD para localizar variables con correlación positiva y variables con correlación negativa. Estas distancias son utilizadas para obtener límites en los cuales se encuentran los datos de operación normal. De esta manera se localiza la falla cuando esta suceda. Por último calcula RP para obtener un valor que describa la distribución de los datos. Una falla puede ser detectada comparando los RP de los datos de prueba contra los datos de operación normal, si estos son similares se asume que no existe falla de lo contrario se calcula nuevamente ACP para observar las variables que están en falla. Finalmente para localizar la falla se observan las posiciones de las distancias en el vector de salida del EMD, las cuales se encuentren fuera de los limites obtenidos en la fase de aprendizaje [1].

Una vez descrito el funcionamiento de este sistema, se describe el propuesto en el presente artículo. El sistema de monitoreo basado en Máquinas de Soporte Vectorial necesita bases de datos tanto en modo de operación normal como en modo de falla para el aprendizaje. Como se muestra en la Figura (3) el sistema no cuenta con ningún tipo de pre procesamiento. Una vez generadas las bases de datos estas sirven como entrada para el entrenamiento y validación del sistema. Si el desempeño del sistema es adecuado este se encuentra listo para recibir datos de prueba. A continuación se muestra un resumen del funcionamiento del sistema propuesto.

1. Se genera una base de datos en modo de operación normal
2. Se genera una base de datos en modo de falla
3. Se entrena a las Máquinas de Soporte Vectorial
4. Se evalúa su desempeño, si no es aceptable regrese al paso anterior, si es aceptable continúe
5. Se toman datos de prueba
6. Se prueban los datos con MSV, si no detecta alguna falla regrese al paso anterior, de lo contrario continue
7. Diagnóstico, muestra la variable en falla y su localización.

## 4. Caso de estudio

Seis variables son supervisadas, estas corresponden a seis sensores encargados de monitorear lo siguiente:

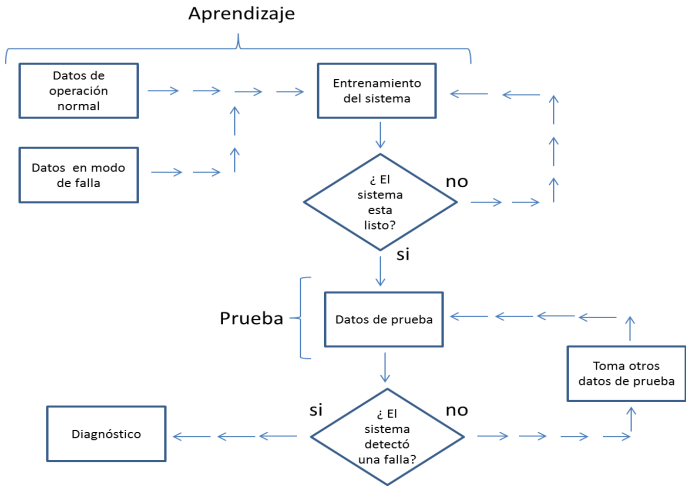



**Fig. 3.** Metodología basada en MSV.

1. Velocidad angular de la llanta frontal izquierda (FI)
2. Velocidad angular de la llanta frontal derecha (FD)
3. Velocidad angular de la llanta trasera izquierda (TI)
4. Velocidad angular de la llanta trasera derecha (TD)
5. Velocidad Longitudinal (VL)
6. Giro del vehículo sobre su propio eje (Yaw)

Las simulaciones se llevaron a cabo sobre la maniobra vehicular conocida como chicane. Chicane se refiere a un cambio en la trayectoria recta del vehículo, por ejemplo, cuando el automóvil realiza una maniobra de rebase. En el análisis realizado por [1] se encontró que para la variable yaw solo es necesario monitorear que en todo momento su valor sea diferente de cero, de lo contrario existirá una falla. Por lo tanto el análisis propuesto se llevará a cabo con las cinco variables restantes. Por lo descrito anterior se concluye que el análisis cuenta con seis estados posibles, operación normal y cinco estados de falla correspondientes a las cinco variables en análisis. La idea central del sistema propuesto es utilizar MSV como clasificador para poder detectar y localizar alguna posible falla, sin embargo como se mostró en la sección 2.1 las MSV estándar clasifica únicamente entre dos clases, por lo tanto se debe utilizar el enfoque de MSV multiclase (MSVMC) y así clasificar en seis distintas clases. Las cuales corresponden a cada uno de los estados mencionados anteriormente.

### 4.1. MSV multiclase (MSVMC)

Entre los métodos más aplicados para lograr una clasificación multiclase con MSV se encuentran los algoritmos conocidos como: uno contra todos (UCT)

el cual construye $M$ MSV donde $M$ es igual al número de clases a clasificar, después cada uno de estos SVM separa una clase del resto. Es decir el $i$-ésimo MSV es entrenado con todas las muestras de entrenamiento de la $i$-ésima clase con una etiqueta distinta a las otras. Otro método es el conocido como uno contra uno (UCU), el cual construye $M * (M-1)/2$ MSV. Estos combinan su función de clasificación para determinar la clase a la que pertenece la muestra de prueba, esto mediante la acumulación de predicciones de clasificación (votos), la predicción con más votos corresponderá a la clasificación final [11,12]. La Figura (4) muestra gráficamente estos dos métodos, las áreas rojas corresponden a las regiones no separables de cada método. Esto no se detalla en este artículo.

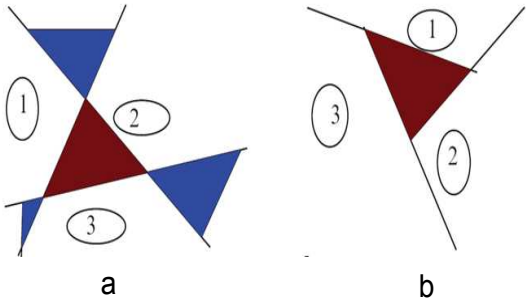



**Fig. 4.** a) UCT, b) UCU, adaptado de [12].

En esta propuesta se utilizará MSVMC usando el algoritmo UCU.

## 5. Análisis y resultados

Se procede a generar las bases de datos tanto en modo de operación normal como en modo de falla, la primera se compone de 4400 observaciones para las 5 variables, la segunda se construye a partir de datos para cada una de las fallas. Se tomaron 1600 observaciones para cada variable en modo de falla, generando una matriz de tamaño $12400 \times 5$. Para entrenamiento y validación del sistema se utilizó validación cruzada.

Como datos de entrada se utilizan los datos de entrenamiento antes descritos, además de un vector de etiquetas de tamaño $12400 \times 1$, el cual contiene seis etiquetas correspondientes a los seis estados posibles. El tiempo aproximado de cómputo para el entrenamiento fue de 6.3 segundos, mientras el desempeño del clasificador fue del 99.95 %. El modelo para la clasificación utilizó un kernel radial y encontró un total de 1612 vectores de soporte. La Tabla 1 muestra la matriz de confusión resultante de la validación del sistema.

Se puede observar que el algoritmo clasifica un dato erróneamente, de 333 datos correspondientes a la clase 4 clasifica un dato de ellos en la clase 0, aun

**Tabla 1.** Matriz de confusión del clasificador MSV.

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| 0 | 899 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 313 | 0 | 0 | 0 | 0 |
| 2 | 0 | | 324 | 0 | 0 | 0 |
| 3 | 0 | 0 | 0 | 331 | 0 | 0 |
| 4 | 1 | 0 | 0 | 0 | 322 | 0 |
| 5 | 0 | 0 | 0 | 0 | 0 | 290 |

así su desempeño es bastante bueno por lo tanto se prosigue con la siguiente etapa del sistema. A continuación se obtienen observaciones para su monitoreo, del total de estas se toman ventanas de 100 muestras para su análisis. Dicho tamaño de muestra se utilizó al encontrar que es un valor estándar empleado y recomendado en la revisión del estado del arte [1].

### 5.1. Presencia de falla

Se considera que existe falla en un sensor cuando en él se presenten cambios en su valor nominal fuera de un rango de $\pm 2\,\%$. La Figura 5 muestra una falla presente en el sensor encargado de monitorear la velocidad angular de la llanta delantera izquierda. Esta falla representó un cambio de menos 5 % en la medición con respecto a su valor nominal y se presentó durante 12 unidades de tiempo (segundos). Al monitorear estos datos con el sistema propuesto, éste detecta la falla en la variable 1 (FI) a partir de la observación 21 y durante 12 unidades de tiempo. Como se mencionó anteriormente el análisis se lleva a cabo en ventanas de 100 datos. En el análisis anterior la ventana se tomó a partir de la muestra 160 hasta la 259, lo cual corresponde con la localización mostrada por el sistema.

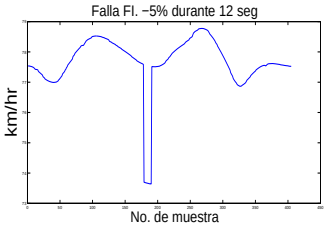



**Fig. 5.** Falla presente en sensor de llanta delantera izquierda.

A continuación se muestran los resultados obtenidos por el sistema propuesto en este artículo en comparación con el sistema propuesto por [1]. Tal comparación se obtuvo al realizar doscientas simulaciones de fallas simples (presencia de falla en una sola variable a la vez) en distintas localizaciones y a diferentes magnitudes

de desviación con respecto a los valores nominales de las variables monitoreadas. En las siguientes tablas se puede observar la comparación de la nueva propuesta contra la metodología empleada en [1]. La Tabla 2 muestra los porcentajes de detección, la Tabla 3 los porcentajes para la identificación, la Tabla 4 lo hace para la localización y la Tabla 5 muestra las comparaciones cualitativas de ambos métodos.

**Tabla 2.** Exactitud de algoritmos en detección de fallas.

| Desviación | Muestras en falla | Detección con MSV | Detección con [1] |
|---|---|---|---|
| 5 % | 15 | 100 % | 100 % |
| 5 % | 10 | 100 % | 96.66 % |
| 5 % | 5 | 100 % | 66.66 % |
| 3 % | 15 | 100 % | 76.66 % |
| 3 % | 10 | 100 % | 63.33 % |
| 3 % | 5 | 100 % | 36.66 % |

**Tabla 3.** Exactitud de algoritmos en identificación de fallas.

| Desviación | Muestras en falla | Identificación con MSV | Identificación con [1] |
|---|---|---|---|
| 5 % | 15 | 100 % | 93.33 % |
| 5 % | 10 | 99.6 % | 86.66 % |
| 5 % | 5 | 100 % | 51.66 % |
| 3 % | 15 | 99.4 % | 65 % |
| 3 % | 10 | 100 % | 55 % |
| 3 % | 5 | 100 % | 21.66 % |

**Tabla 4.** Exactitud de algoritmos en Localización de fallas.

| Desviación | Muestras en falla | Localización con MSV | Localización con [1] |
|---|---|---|---|
| 5 % | 15 | 100 % | 91.66 % |
| 5 % | 10 | 100 % | 90 % |
| 5 % | 5 | 100 % | 56.66 % |
| 3 % | 15 | 100 % | 71.66 % |
| 3 % | 10 | 100 % | 61.66 % |
| 3 % | 5 | 100 % | 26.66 % |

La gran eficiencia de clasificación de las máquinas de soporte vectorial así como su capacidad de trabajar con gran número de datos da como resultado un

**Tabla 5.** Características de ambos métodos.

| Método propuesto (MSVMC) | Método [1] |
|---|---|
| • Necesita bases de datos en modo de operación normal y en modo de falla | • Solo necesita base de datos en modo de operación normal |
| • Requiere tiempo de entrenamiento | • Capaz de aprender apartir de los datos en operación normal |
| • Capaz de manejar ruido | • Capaz de manejar ruido |
| • Capaz de operar en linea | • Operación fuera de linea |

desempeño superior del sistema propuesto en este artículo para la detección, clasificación y localización de fallas en comparación con el método propuesto por [1]. Debido a que este último utiliza solo datos de operación normal para su aprendizaje y su funcionamiento se basa en la variabilidad de los mismos. Dando como resultado un sistema el cual su exactitud depende del tamaño de muestra a analizar. Esto significa que el sistema es incapaz de trabajar en línea.

## 6. Conclusiones

En este artículo se presentó un sistema para la detección y diagnóstico de fallas para la dinámica lateral de un automóvil basado en Máquinas de Soporte Vectorial Multiclase. El desempeño de dicho sistema fue comparado con el método propuesto por [1], el cual combina Escalamiento Multidimensional, Rangos Percentiles y Análisis de Componentes Principales. Los resultados mostraron que el sistema propuesto tiene un mejor desempeño en tareas como detección, identificación y localización. Debido a la gran eficiencia de clasificación de las máquinas de soporte vectorial así como su capacidad de trabajar con gran número de datos. Además se observó que la exactitud del método basado en MSV no depende de la cantidad de muestras a analizar, por lo que al realizar pruebas con solo una observación los resultados fueron prácticamente iguales a los realizados con 15, 10 y 5 observaciones. Por lo tanto al respecto se concluye que el sistema propuesto es capaz de trabajar en línea, a diferencia del método [1]. Las principales desventajas del método con MSV frente al método mencionado, radican en la necesidad de bases de datos en modo de falla y al tiempo computacional generado por entrenamiento, siendo esta última menos importante ya que el tiempo de entrenamiento fueron aproximadamente seis segundos, un tiempo relativamente corto. De acuerdo a los resultados presentados en este artículo se concluye que las Máquinas de Soporte Vectorial son una técnica robusta para tareas de clasificación, y presenta excelentes resultados en el campo del diagnóstico y detección de fallas.

## Referencias

1. Juan Pablo Nieto González, Luis Eduardo Garza Castañon, Abdelhamid Rabhi, Ahmed El Hajjaji: Fault Detection and Diagnosis of a Vehicle Combining Multidi-

mensional Scaling, Percentiles Range and PCA. In: 9th international conference on Sciences and Techniques of Automatic control & computer engineering, Túnez, pp. 1–11 (2008)
2. Baoling Liu, Dibo Hou, Pingjie Huang, Banteng Liu, Huayi Tang, Wubo Zhang, Peihua Chen, Guangxin Zhang. An improved PSO-SVM model for online recognition defects in eddy current testing. Nondestructive Testing and Evaluation, 28(4), 367–385 (2013)
3. Esraa Elhariri, Nashwa El-Bendary, Mohamed Mostafa M. Fouad, Jan Platos, Aboul Ella Hassanien, Ahmed M.M. Hussein: Multi-class SVM Based Classification Approach for Tomato Ripeness. Innovations in Bio-inspired Computing and Applications, Advances in Intelligent Systems and Computing, 175–186 (2014)
4. Amaury B. Andre, Eduardo Beltrame, Jacques Wainer: A combination of support vector machine and K-nearest neighbors for machine fault detection. Applied Artificial Intelligence: An International Journal. Vol. 27, 36–49 (2013)
5. N.M. Khan, R.Ksantini, I.S.Ahmad, B.Boufama: A novel SVM + NDA model for classification with an application to face recognition. Pattern Recognition, Vol. 45, 66–79 (2011)
6. Elkin Eduardo García Díaz: Boosting support vector machines. Tesis de maestria, Bogotá, Colombia, Universidad de los Andes, facultad de ingeniería (2005)
7. Gong Yanjie, Gao Xuejin, Wang Pu, Qi Yongsheng: Online Modeling Method Based on Dynamic Time Warping and Least Squares Support Vector Machine for Fermentation Process. In: 8th World Congress on Intelligent Control and Automation. inan, China, pp. 481–485 (2010)
8. Nicholas I. Sapankevich, Ravi Sankar: Time series prediction using Support Vector Machines. IEEE computational intelligence magazine, 24–37 (2009)
9. Hsu Chun-Chin, Chen Mu-Chen, Chen Long-Sheng: Intelligent ICA-SVM fault detector for non-Gaussian multivariate process monitoring. Expert Systems with Applications, Vol. 37, 3264–3273 (2010)
10. Zhiqiang Ge, Zhihuan Song: Multivariate Statistical Process Control. Springer-Verlag, London (2013)
11. Hassen Keskes, Ahmed Brahama, Zied Lachiri: Broken rotor bar diagnosis in induction machines through stationary wavelet packet transform and multiclass wavelet SVM. Electric Power Systems Research, Vol. 97, 151–157 (2013)
12. Oscar Castillo, Li Xu, Sio-Long Ao: Trends in Intelligent Systems and Computer Engineering. Springer Science + Business Media, New York, USA (2008)

# Modelado de un problema de iluminación inteligente de una oficina usando un enfoque de actualización basado en Answer Set Programming

Mayra Cordero, Claudia Zepeda, José Luis Carballido

Benemérita Universidad Atónoma de Puebla,
Facultad de Ciencias de la Computación,
Puebla, Puebla, México

{maycorderor,czepedac,jlcarballido7}@gmail.com

**Resumen.** La actualización es una acción que nos permite renovar información, es decir, consiste en convertir información retrasada en información actual. Por otra parte, Answer Set Programming es un lenguaje de programación lógico y declarativo basado en la lógica. El propósito de este artículo es utilizar el enfoque de actualización de secuencias de programas lógicos basado en Answer Set Programming para modelar y resolver el problema de iluminación inteligente de una oficina. Este problema consiste en controlar la iluminación de una oficina donde el encendido y apagado de cuatro lámparas dependa de la ubicación de una persona.

**Palabras clave:** Actualización, programación lógica, Answer Set Programming.

## 1. Introducción

La programación lógica (LP) ofrece de manera natural representar conocimiento declarativo utilizando el lenguaje de la lógica matemática. La finalidad de un programa lógico es buscar todos los valores que hacen verdadero a un programa y como resultado de las combinaciones de dichos valores obtener una respuesta [9]. Para que un programa lógico tenga un significado y pueda ser interpretado por programas computacionales se le debe asignar una semántica, dicha semántica permite determinar el tipo de conclusiones que se pueden establecer a partir de un conjunto de reglas [9].

Debido a esto surgen lenguajes que permiten escribir programas lógicos tales como Answer Set Programming (ASP) [2,10]. ASP es un lenguaje de programación lógico y declarativo para la representación del conocimiento que nos permite manejar problemas con conocimiento por defecto y razonamiento no-monotóno mediante el concepto de negación como falla [14]. El éxito de ASP se debe en gran medida a su capacidad para resolver problemas [6]. Dentro de la gama de problemas que pueden enfrentarse con ASP se encuentran problemas de planeación, modelado de agentes lógicos, actualización y aplicaciones de inteligencia artificial en general [7,3,8,13].

La contribución de este trabajo es modelar y resolver el problema de iluminación inteligente de una oficina utilizando el enfoque de actualización basado en secuencias de programas lógicos y ASP [14,12]. Nos interesamos en este problema ya que una de las causas principales del excesivo consumo energetico es el tiempo que permanecen encendidas las lámparas sin ningún beneficio [4].

Existen diferentes tipos de oficinas, en esta ocasión tomaremos como modelo una oficina conformada por cuatro lámparas.

El Modelo del sistema consiste en controlar el estado de encendido y apagado de las cuatro lámparas dentro de la oficina usando cuatro sensores de presencia. Para este modelo proponemos que a dicha oficina solo tenga acceso una sola persona. Como las actividades de la persona dentro de la oficina no son estáticas el estado de encendido y apagado de las lámparas debe mantenerse actualizando para cambiar de un estado a otro según los detectores de presencia.

En la sección 2 llamada Marco teórico describimos los conceptos que se consideran importantes para resolver el problema de iluminación inteligente de oficinas. La sección 3 corresponde a la descripción de problema, el modelado del problema, la aplicación del enfoque de actualización, discusión de resultados y finalmente la sección 4 corresponde a las conclusiones.

## 2. Marco teórico

En esta sección presentamos un panorama general de las ideas y conceptos principales tratados para el desarrollo del problema de iluminación inteligente de una oficina. Incluimos los conceptos principales como Programas Lógicos y ASP, y Enfoque de Actualización.

### 2.1. Programas lógicos y ASP

Entendemos a los programas lógicos como teorías proposicionales. Una teoría proposicional es sólo un conjunto finito de fórmulas bien formadas. Se le puede llamar teoría o programa en el que no se presenta ninguna ambigüedad. Utilizamos el lenguaje de la lógica proposicional de la forma habitual para el modelado del problema de iluminación inteligente de una oficina. Consideramos símbolos proposicionales: $m, n, p, q, r, s, t, ...,$ conectivos proposicionales: $\wedge, \vee, \leftarrow, \bot, -, \neg$ y símbolos auxiliares: '(',')','.'. También consideramos los dos tipos de negación: negación fuerte o clásica escrita como $(-)$ y negación débil o fracaso escrita como $(\neg)$. Asumimos que para cualquier fórmula proposicional $f$, $\neg f$ es una abreviatura de $f \leftarrow \bot$. Un átomo es un símbolo proposicional. Un literal es un átomo $a$ o la negación de un átomo $-a$. Dado un conjunto de átomos $A$, definimos $-A = \{-l \,|\, l \in A\}, \neg A = \{\neg l \,|\, l \in A\}$, y $Lit_A = A \cup -A$. La signatura de un programa lógico $P$ denotada como $\mathcal{L}_P$, es el conjunto de átomos que se producen en $P$. Por lo general, la sintaxis de fórmulas bien formadas dentro de los programas lógicos se ha limitado a la forma $f \leftarrow g$ que es sólo otra manera de escribir $g \rightarrow f$. Las fórmulas bien formadas $f \leftarrow g$ se llaman reglas donde $f$

se llama la cabeza y $g$ el cuerpo de la regla. Una regla con un cuerpo vacío, $f \leftarrow$, se llama un hecho, y una regla sin cabeza $\bot \leftarrow g$ se llama restricción. Hechos y restricciones también se denominan $f$ y $\leftarrow g$ respectivamente. Queremos hacer hincapié en el hecho de que en el enfoque de actualización, un programa lógico se interpreta como una teoría proposicional. Como es habitual en ASP, utilizaremos programas proposicionales. En este trabajo utilizamos la semántica de answer sets definida en términos de la llamada reducción Gelfond-Lifschitz [5]. Vale la pena mencionar que en el enfoque de actualización un átomo y su contraparte negada nunca pueden ocurrir en el mismo answer set. Por último, se dice que $B$ es un subconjunto de $A$ (escrito como $B \subseteq A$) si y sólo si todos los miembros de $B$ son miembros de $A$, y decimos que $B$ es un subconjunto propio de $A$ (escrito como $B \subset A$) si y sólo si $B \subseteq A$ y $B \neq A$.

### 2.2. Mínimos generalizados asnwer sets

En esta sección recordamos la sintaxis y la semántica de los programas lógicos abductivos como se presenta en [11].

**Definición 2.21 (Programa Abductivo Lógico)** *[1] Un* programa abductivo lógico *es un par $\langle P, A\rangle$ donde $P$ es un programa arbitrario y $A$ es un conjunto de literales, llamados* abductivos.

En [1], la noción de minimo generalizdo answer set se utiliza para definir la semántica de un programa lógico abductivo.

**Definición 2.22 (Generalizados Answer Sets)** *[1] $\langle M, \Delta\rangle$ es un generalizado answer set de un programa abductivo $\langle P, A\rangle$ si y sólo si $\Delta \subseteq A$ y $M$ es un answer set de $P \cup \Delta$.*

En [1] también se presenta un orden entre generalizados answer sets a fin de obtener los mínimos generalizados answer sets de un programa lgico abductivo. Veremos que un mínimo generalizado answer set es un par $\langle M, \Delta\rangle$, nosotros sólo estamos interesados en $M$ .

**Definición 2.23 (Orden Abductivo de Inclusión)** *[1] Podemos establecer un orden entre generalizados answer sets de la siguiente manera: Sea $\langle M_1, \Delta_1\rangle$ y $\langle M_2, \Delta_2\rangle$ generalizados answer sets de $\langle P, A\rangle$, definimos $\langle M_1, \Delta_1\rangle < \langle M_2, \Delta_2\rangle$ si $\Delta_1 \subset \Delta_2$.*

**Definición 2.24 (Mínimos Generalizados Answer Sets)** *[1] $\langle M, \Delta\rangle$ es un mínimo generalizado answer set de $\langle P, A\rangle$, si y sólo si $\langle M, \Delta\rangle$ es un generalizado answer set de $\langle P, A\rangle$ y es el mínimo abductivo por orden de inclusión.*

### 2.3. Mínimos extendidos generalizados Answer Sets

La semántica para actualizaciones que consiste en una secuencia de programas se basa en una extensión de la definición de mínimos generalizados answer

sets. La extensión se llama mínimos extendidos generalizados answer sets (MEG) [11]. También el enfoque de actualización utiliza la definición de un programa extendido abductivo lógico (EAL). Esta definición es similar a la definición 1, pero se agrega una función sobreyectiva. De un programa EAL obtenemos sus extendidos explícitos generalizados answer sets (EEG). Utilizamos la función sobreyectiva de un programa EAL para definir un orden entre sus EEG answer sets y obtener sus MEG answer sets [11].

**Definición 2.31 (Programa Extendido Abductivo Lógico (EAL))** *[11] Un* programa extendido abductivo lógico (EAL) *es un triple $\langle P, A, f\rangle$ tal que $P$ es un programa arbitrario, $A$ es un conjunto de átomos, y $f$ es alguna función sobreyectiva con dominio $A$ y codominio $\{1, \ldots, N\}$, $N > 0$.*

***Ejemplo* 1** *Podemos definir un programa EAL $\langle P, A, f\rangle$ tal que $P$ es el siguiente programa:*

$$\begin{aligned} a &\leftarrow \neg x_1^1. \\ b &\leftarrow a, \neg x_2^1. \\ -a &\leftarrow \neg x_1^2. \\ c &\leftarrow . \end{aligned}$$

*$A$ es el siguiente conjunto de átomos: $\{x_1^1, x_2^1, x_1^2\}$; y $f : A \to \{1, 2\}$ es la función tal que $f(x_j^i) = i$.*

**Definición 2.32 (Extendidos Generalizados answer sets (EG))** *[11] Sea $\langle P, A, f\rangle$ un programa EAL, $M$ un conjunto de literales, y $\Delta \subseteq A$. Un* extendido generalizado (EG) *answer set de $\langle P, A, f\rangle$ es un par $\langle M, \Delta\rangle$ si $M$ es un answer set de $P \cup \Delta$.*

***Ejemplo* 2** *Vamos a considerar el programa EAL $\langle P, A, f\rangle$ del Ejemplo 1. El cuadro 1 muestra los diferentes EG answer sets de $\langle P, A, f\rangle$ con su respectivo $\Delta \subseteq A$ (los subconjuntos que no aparecen en la tabla no tienen EG answer sets).*

**Tabla 1.** Los EG answer sets del programa $\langle P, A, f\rangle$ del Ejemplo 1.

| $\Delta$ | $\langle M, \Delta\rangle$ |
|---|---|
| $\{x_1^1, x_2^1\}$ | $\langle\{x_1^1, x_2^1, -a, -b, c\}, \{x_1^1, x_2^1\}\rangle$ |
| $\{x_1^1, x_2^1, x_1^2\}$ | $\langle\{x_1^1, x_2^1, x_1^2, -a, c\}, \{x_1^1, x_2^1, x_1^2\}\rangle$ |
| $\{x_1^1, x_2^1, x_2^2\}$ | $\langle\{x_1^1, x_2^1, x_2^2, -b, c\}, \{x_1^1, x_2^1, x_2^2\}\rangle$ |
| $\{x_1^1, x_2^1, x_1^2, x_2^2\}$ | $\langle\{x_1^1, x_2^1, x_1^2, x_2^2, c\}, \{x_1^1, x_2^1, x_1^2, x_2^2\}\rangle$ |
| $\{x_1^1, x_1^2, x_2^2\}$ | $\langle\{x_1^1, x_1^2, x_2^2, c\}, \{x_1^1, x_1^2, x_2^2\}\rangle$ |
| $\{x_2^1, x_1^2, x_2^2\}$ | $\langle\{x_2^1, x_1^2, x_2^2, a, c\}, \{x_2^1, x_1^2, x_2^2\}\rangle$ |
| $\{x_1^2, x_2^2\}$ | $\langle\{x_1^2, x_2^2, a, b, c\}, \{x_1^2, x_2^2\}\rangle$ |

Ahora presentamos un orden entre los EG answer sets de un programa de EAL.

**Definición 2.33 (Orden de inclusión entre los EG answer sets)** *[11] Sea $\langle P, A, f\rangle$ un programa EAL donde el codominio de $f$ es el conjunto $\{1, \ldots, N\}$, $N > 0$. Sea $A_i = \{a \in A | f(a) = i\}$. Establecemos una orden de inclusión entre EG answer sets de $\langle P, A, f\rangle$ como sigue: Sea $\langle M_1, \Delta_1\rangle$ y $\langle M_2, \Delta_2\rangle$ EG answer sets de $\langle P, A, f\rangle$. Definimos $\langle M_1, \Delta_1\rangle \leq_{inclu} \langle M_2, \Delta_2\rangle$ si y sólo si hay $k$, $1 \leq k \leq N$ tal que $(\Delta_1 \cap A_k) \subset (\Delta_2 \cap A_k)$, y para todo $j$, $k < j \leq N$, $(\Delta_1 \cap A_j) = (\Delta_2 \cap A_j)$.*

***Ejemplo* 3** *Vamos a considerar el programa EAL $\langle P, A, f\rangle$ del ejemplo 1 y sus EG answer sets del cuadro 1. Recordamos que $N = 2$ y $f : A \to \{1, 2\}$ es la función en la que $f(x^i_j) = i$ . Podemos comprobar que $A_1 = \{x^1_1, x^1_2\}$ y $A_2 = \{x^2_1\}$. Así, de acuerdo al cuadro 1 y definición 2.33, podemos comprobar que*
$\langle\{x^1_2, x^2_1, x^2_2, a, c\}, \{x^1_2, x^2_1, x^2_2\}\rangle \leq_{inclu} \langle\{x^1_1, x^1_2, x^2_1, x^2_2, c\}, \{x^1_1, x^1_2, x^2_1, x^2_2\}\rangle$*; ya que $k = 1$ tal que $(\{x^1_2, x^2_1, x^2_2\} \cap A_1) \subset (\{x^1_1, x^1_2, x^2_1, x^2_2\} \cap A_1)$, i.e, $\{x^1_2\} \subset \{x^1_1, x^1_2\}$, y para todo $j$, $1 < j \leq 2$ tenemos que*
$(\{x^1_2, x^2_1, x^2_2\} \cap A_2) = (\{x^1_1, x^1_2, x^2_1, x^2_2\} \cap A_2)$, *i.e,* $\{x^2_1, x^2_2\} = \{x^2_1, x^2_2\}$.
*De manera similar , podemos comprobar que*
$\langle\{x^1_1, x^1_2, -a, -b, c\}, \{x^1_1, x^1_2\}\rangle \leq_{inclu} \langle\{x^1_1, x^1_2, x^2_1, -a, c\}, \{x^1_1, x^1_2, x^2_1\}\rangle$*, ya que $K = 2$ de tal manera que $(\{x^1_1, x^1_2\} \cap A_2) \subset (\{x^1_1, x^1_2, x^2_1\} \cap A_2)$, i.e., $\emptyset \subset \{x^2_1\}$, y no hay $j$, $2 < j \leq 2$.*

Notemos que también podemos definir un orden de cardinalidad entre los EG answer sets de un programa EAL $\langle P, A, f\rangle$, denotado por $\leq_{card}$. Esta definición puede obtenerse a partir de la Definición 2.33 reemplazando el conjunto de criterio de inclusión por conjunto de criterios de cardinalidad. En el resto de este trabajo se utilizará sólo orden de inclusión entre los EG answer sets y vamos a escribir $\leq$ para denotar este orden. También vale la pena mencionar que en Definicón 2.33 el programa de $P$ se utiliza para obtener los EG answer sets, si bien no es utilizado para definir el orden entre los EG answer sets. El orden entre EG answer sets se define en términos de los subconjuntos de $A$ ($\Delta_i$ and $A_j$). Por otra parte, este orden se puede utilizar para obtener los mínimos extendidos generalizados answer sets de un programa EAL $\langle P, A, f\rangle$. Veremos que un mínimo extendido generalizado answer set es un par $\langle M, \Delta\rangle$, por esta ocasión, sólo estamos interesados en $M$.

**Definición 2.34 (Mínimos Extendidos Generalizados answer sets (MEG))** *[11] Sea $\langle P, A, f\rangle$ un programa EAL. $\langle M, \Delta\rangle$ es un* mínimo extendido generalizado answer set (MEG) *de $\langle P, A, f\rangle$ si $\langle M, \Delta\rangle$ es un EG answer set de $P$ y no hay EG answer set $\langle M', \Delta'\rangle$ de $\langle P, A, f\rangle$ tal que $\langle M', \Delta'\rangle \leq \langle M, \Delta\rangle$.*

***Ejemplo* 4** *Vamos a considerar el programa EAL $\langle P, A, f\rangle$ del Ejemplo 1. De acuerdo con el cuadro 1 y la definición 2.34, podemos comprobar que*
$\langle\{x^1_1, x^1_2, -a, -b, c\}, \{x^1_1, x^1_2\}\rangle$ *es el único MEG answer set de $\langle P, A, f\rangle$ ya que no hay EG answer set $\langle M', \Delta'\rangle$ de $\langle P, A, f\rangle$ tal que*
$\langle M', \Delta'\rangle \leq \langle\{x^1_1, x^1_2, -a, -b, c\}, \{x^1_1, x^1_2\}\rangle$.

### 2.4. Enfoque de actualización utilizando MEG answer sets

Consideramos las secuencias de programas de actualización [11] . Formalmente una secuencia de programas de actualización, entendemos una secuencia $(P_1, \ldots, P_n)$ de programas lógicos donde $N_{P_i}$ es el número de reglas de cada programa lógico. Decimos que $\mathbf{P}$ is una secuencia de actualización de programas sobre $\mathcal{L}_{\mathbf{P}}$ si y solo si $\mathcal{L}_{\mathbf{P}}$ representa un conjunto de átomos que aparecen en $\bigcup_{1 \leq i \leq n} P_i$ [9].

**Definición 2.41 (Programa de actualización de una secuencia de (LP))** *[11] Dada una secuencia de actualización de programas $\boldsymbol{P} = (P_1, \ldots, P_n)$ sobre $\mathcal{L}_{\boldsymbol{P}}$, definimos una* secuencia de programas de actualización *$\boldsymbol{P}_{\oslash} = P_1 \oslash \cdots \oslash P_n$ sobre $\mathcal{L}^*_{\boldsymbol{P}}$ (que se extiende $\mathcal{L}_{\boldsymbol{P}}$ por nuevos átomos abducibles) constituidos por los siguientes elementos:*

1. *Todas las restricciones en $P_1, \ldots, P_{n-1}$,*
2. *Para cada $r_j \in P_i, 1 \leq i \leq n-1$, $1 \leq j \leq N_{P_i}$ añadimos la regla $r_j \leftarrow \neg b^i_j$, donde $b^i_j$ es un nuevo átomo abducible,*
3. *Todas las reglas y restricciones $r \in P_n$.*

Definimos el programa EAL de $\mathbf{P}$ como triple $\langle \mathbf{P}_{\oslash}, B, f \rangle$ donde $B$ es el conjunto de nuevos átomos abducibles de $\mathbf{P}_{\oslash}$, i.e., $B = \{b^i_j \mid b^i_j \in \mathbf{P}_{\oslash}, 1 \leq i \leq n-1, 1 \leq j \leq N_{P_i}\}$; y $f : B \to \{1, \ldots, n-1\}$ es la función sobreyectiva donde $f(b^i_j = i)$.

La semántica de nuestro operador de actualización se basa en los MEG answer sets de un programa EAL de una secuencia de actualización determinada.

**Definición 2.42** *Sea $\langle \boldsymbol{P}_{\oslash}, B, f \rangle$ el programa EAL de una secuencia de actualización $\boldsymbol{P}$ sobre $\mathcal{L}_{\boldsymbol{P}}$. $M$ es un $\oslash$-actualización answer set de $\boldsymbol{P}$, si $M'$ es un MEG answer set de $\langle \boldsymbol{P}_{\oslash}, B, f \rangle$ y $M = M' \cap \mathcal{L}_{\boldsymbol{P}}$.*

***Ejemplo* 5** *Sea $\boldsymbol{P} = (P_1, P_2, P_3)$ una secuencia de actualización a través de $\mathcal{L}_{\boldsymbol{P}} = \{a, b, c\}$ donde: $P_1$ es el siguiente programa lógico,*

$$a \leftarrow .$$
$$b \leftarrow a.$$

*$P_2$ es el siguiente programa lógico,*

$$-a \leftarrow .$$
$$-b \leftarrow .$$

*and $P_3$ es el siguiente programa lógico,*

$$c \leftarrow .$$

*Por lo tanto el programa de actualización $\boldsymbol{P}_{\oslash} = P_1 \oslash P_2 \oslash P_3$ sobre $\mathcal{L}^*_{\boldsymbol{P}}$ (que se extiende de $\mathcal{L}_{\boldsymbol{P}}$ por nuevos átomos abducibles $\{x^1_1, x^1_2, x^2_1, x^2_2\}$ ). Así el programa lógico de actualización $\boldsymbol{P}_{\oslash} = P_1 \oslash P_2 \oslash P_3$ es el siguiente programa lógico:*

$$
\begin{aligned}
a &\leftarrow \neg x_1^1.\\
b &\leftarrow a, \neg x_2^1.\\
-a &\leftarrow \neg x_1^2.\\
-b &\leftarrow \neg x_2^2.\\
c &\leftarrow .
\end{aligned}
$$

El programa EAL de $\mathbf{P}$ es el triple $\langle P_{\oslash}, B, f\rangle$, donde $f : B \rightarrow \{1, 2\}$ es la función $f(b_j^i = i)$. Por lo tanto sabemos que su único MEG answer set es $\{x_1^1, -a, -b, c\}$. Por lo tanto, $\{-a, -b, c\}$ es el único $\oslash$-actualización answer set de $\mathbf{P}$.

## 3. Descripción del problema

Como ya mencionamos el problema consiste en modelar y resolver el problema de iluminación inteligente de una oficina. El problema está enfocado en una oficina conformada por cuatro lamparas l1, l2, l3 y l4 que se controlan por cuatro sensores de presencia s1, s2, s3 y s4. Los sensores s1,s2,s3 y s4 son los encargados de mantener el estado de encendido y apagado de las lamparas l1,l2,l3 y l4 respectivamente.

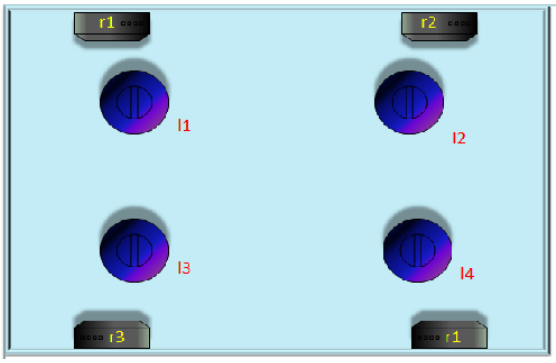



**Fig. 1.** Sistema de iluminación

Como se muestra en la figura 1 Los sensores s1 y s2 detectan presencia cuando una persona se encuentre en la oficina enfrente de lado derecho o izquierdo respectivamente y los sensores s3 y s4 detectan presencia cuando una persona se encuentre en la oficina atrás de lado derecho o izquierdo respectivamente. Los sensores son los encargados de actualizar el estado de las lámparas a encendido o apagado dependiendo si detectan personas o no. Si s1 detecta presencia el estado de l1 debe ser encendido, si s2 detecta presencia el estado de l2 debe ser encendido, si s3 detecta presencia el estado de l3 debe ser encendido y finalmente si s4 detecta presencia el estado de l4 debe ser encendido. Por otra parte si s1 no

detecta presencia el estado de l1 debe ser apagado, si s2 detecta no presencia el estado de l2 debe ser apagado, si s3 no detecta presencia el estado de l3 debe ser apagado y finalmente si s4 no detecta presencia el estado de l4 debe ser apagado.

### 3.1. Modelado del problema

Una ves que establecimos como pueden ser los estados de encendido y apagado de lámparas proponemos el siguiente escenario para ver el funcionamiento del sistema. El escenario consiste en suponer que inicialmente en un tiempo t1 no hay ninguna persona dentro de la oficina esto significa que ningún sensor ha detectado presencia y por lo tanto las cuatro lámparas se encuentran en un estado de apagado, posteriormente en un tiempo t2 ingresa una persona, obviamente esta persona se posiciona dentro de la oficina y en este caso suponemos que se posiciona atrás de lado izquierdo, por lo tanto el sensor encargado de esa posición es s3 y en un tiempo t3 eperamo que el sistema actualice el estado de los sensores y de las lámparas para poder encender l3. De esta manera el estado final que esperamos de nuestro sistema es que l3 se encuentre en un estado de encendido y l1,l2 y l4 se encuentren en une estado de apagado. A continuación presentamos tres diagramas que representan graficamente el escenario:

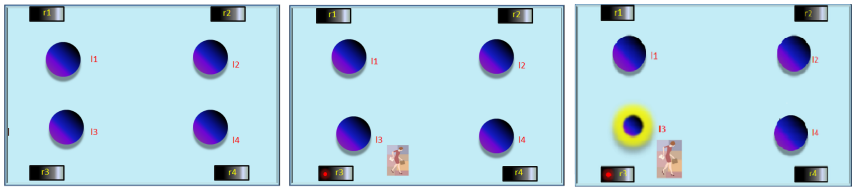


**Fig. 2.** Escenario en lo diferentes tiempos

Para modelar nuestro problema representamos la información por medio de programas lógicos, nosotros para este problema utilizamos tres programas lógicos $P_1$, $P_2$ y $P_3$ pero el enfoque de actualización permite utilizar los n-programas lógicos que se necesiten.

En esto tres programas lógicos representamos el comportamiento y las acciones que debe realizar nuestro sistema. Para representar el estado de encendido de las lámparas utilizamos el valor 1, de igual forma para representar si un sensor detecta presencia utilizamos también el valor 1. El programa $P_1$ lo utilizamos para representar la base de conocimiento inicial y corresponde al momento en el cúal no se encuentra la persona en la oficina y por lo tanto el estado de las lámparas es apagado, para las reglas de este programa utilizamos el concepto de negación clásica.

El programa $P_2$ lo utilizamos para representar el estado de las lámparas según los sensores de presencia. En las reglas de este programa utilizamos la negación como falla que nos permite representar que si no tenemos evidencia de que un sensor detecta presencia el estado de las lámparas debe ser apagado.

El programa $P3$ lo utilizamos para representar la información que indica que sucede cuando un sensor detecta presencia que lámpara es la que debe cambiar su estado a encendido. Finalmente representamos la situaciónes que se le pueden presentar al sistema, en este caso vamos a asumir que s3 detectó presencia. A continuación motramo los programas con sus reglas correspondientes:

$P_1$ :

$$r1 : -l1(1).$$
$$r2 : -l2(1).$$
$$r3 : -l3(1).$$
$$r4 : -l4(1).$$

$P_2$ :

$$r1 : -l1(1) \leftarrow \neg s1(1).$$
$$r2 : -l2(1) \leftarrow \neg s2(1).$$
$$r3 : -l3(1) \leftarrow \neg s3(1).$$
$$r4 : -l4(1) \leftarrow \neg s4(1).$$

$P_3$ :

$$r1 : l1(1) \leftarrow s1(1).$$
$$r4 : l2(1) \leftarrow s2(1).$$
$$r6 : l3(1) \leftarrow s3(1).$$
$$r8 : l4(1) \leftarrow s4(1).$$
$$r9 : s3(1).$$

### 3.2. Aplicación del enfoque de actualización

De manera formal decimos:
Sea $\mathbf{P} = P_1, P_2, P_3$ una secuencia de programas, donde $P_1, P_2, P_3$. Y sea $B$ el conjunto de nuevos átomos abductivos $\{x_1^1, x_2^1, x_3^1, x_4^1, x_1^2, x_2^2, x_3^2, x_4^2\}$. El nuevo programa lógico de actualización $P_{\oslash}= P_1 \oslash P_2 \oslash P_3$ el siguiente programa:

$$r1 : -l1(1) \leftarrow \neg x_1^1.$$
$$r2 : -l2(1) \leftarrow \neg x_2^1.$$
$$r3 : -l3(1) \leftarrow \neg x_3^1.$$
$$r4 : -l4(1) \leftarrow \neg x_4^1.$$
$$r1 : -l1(1) \leftarrow \neg s1(1), \neg x_1^2.$$
$$r2 : -l2(1) \leftarrow \neg s2(1), \neg x_2^2.$$
$$r3 : -l3(1) \leftarrow \neg s3(1), \neg x_3^2.$$
$$r4 : -l4(1) \leftarrow \neg s4(1), \neg x_4^2.$$
$$r1 : l1(1) \leftarrow s1(1).$$
$$r4 : l2(1) \leftarrow s2(1).$$
$$r6 : l3(1) \leftarrow s3(1).$$
$$r8 : l4(1) \leftarrow s4(1).$$
$$r9 : s3(1).$$

Una vez obtenido el nuevo programa actualizado, obtuvimos el conjunto de EEG answer sets. De todo el conjunto de EEG answer sets obtenemos el MEG answer set por medio del orden de inclusion. Por lo tanto el programa EAL de $\mathbf{P}$

es un triple $\langle P_{\oslash}, B, f\rangle$ donde $f{:}\rightarrow \{1,2\}$es la función $f(b_j^i = i)$. El MEG answer set de $\langle P_{\oslash}, B, f\rangle$ es:
$\{x_3^1, s3(1), -l1(1), -l2(1), -l4(1), l3(1)\}$ y como solo estamos interesados en $M$ restamos el abductivo que nos permitió la actualización y por lo tanto, el answer set $\oslash$-actualización es: $\{s3(1), -l1(1), -l2(1), -l4(1), l3(1)\}$

### 3.3. Discusión de resultados

Observando el único MEG answer set $\{s3(1), -l1(1), -l2(1), -l4(1), l3(1)\}$ nos damos cuenta como el sistema efectivamente actualiza los estados de los sensores y la lámparas y una vez percibida presencia por el sensor s3 presencia el sistema actualiza el estado de l3 de un estado apagado a un estado encendido. con este resultado observamos que el sistema si logró actualizar los estados de las lámparas en este caso de l3 según el sensor s3 que fue el que detecto presencia.

## 4. Conclusiones

Dentro de las conclusiones de este trabajo podemos decir que cumplimos con el objetivo de modelar y resolver el problema de iluminación inteligente de una oficina utilizando el enfoque de actualización de secuencias de programas lógicos basado en ASP. Cabe mencionar que este escenario es solo un ejemplo que utilizamos para representar el comportamiento del enfoque de actualización. Ya que se puede utilizar cualquier escenario donde sea necesaria la actualización.

Por otra los resultados que obtuvimos fueron resultados satisfactorios por lo tanto concluimos que usando este enfoque es posible modelar problemas con información cambiante o información contradictoria gracias al potencial de ASP. Finalmente podemos decir que con respecto al enfoque de actualización no solo observamos la utilidad que tiene en este sistema sino también podría ser util en cualquier otro sistema o área donde exista la necesidad de actualización.

**Agradecimientos.** Este trabajo ha sido apoyado por el Proyecto de Ciencia Básica del Fondo Sectorial SEP-CONACyT con número de registro 101581.

## Referencias

1. Balduccini, M., Gelfond, M.: Logic programs with consistency-restoring rules. In: on Logical Formalization of Commonsense Reasoning, I.S. (ed.) AAAI 2003 Spring Symposium. In P. Doherty, J. McCarthy, and M.-A. Williams, editors (1988)
2. Baral, C.: Knowledge Representation, reasoning and declarative problem solving with Answer Sets. Cambridge University Press, Cambridge (2003)
3. Carmen L. Garca M., P.R.M.G., Portillo, R.B.J.J.: A. robot platform motion planning using answer set programming. In: Latin American Workshop on Non-Monotonic Reasoning (2011)
4. de Colombia, S.E.: Soluciones en Control de Iluminación

5. Gelfond, M., Lifschitz: The Stable Model Semantics for Logic Programming. In: Kowalski, R., Bowen, K. (eds.) 5th Conference on Logic Programming. pp. 1070–1080. MIT Press (1988)
6. Gelfond, M., Lifschitz, V.: The Stable Model Semantics for Logic Programming. In: Kowalski, R., Bowen, K. (eds.) 5th Conference on Logic Programming. pp. 1070–1080. MIT Press (1988)
7. Lifschitz, V.: What is answer set programming? In: Proceedings of AAAI-08 (2008)
8. Michael Gelfond, R.S., Baral, C.: Answer set programming as the basis for a homeland security qask. In: AAAI Spring Symposium: AI Technologies for Homeland Security (2005)
9. Navarro, J.A.: Tesis profesional de Licenciatura: Lógica Aplicada a Answer Sets (Universidad de las Américas, Puebla, 2003)
10. Niemelä, I.: Logic programming with stable model semantics as a constraint programming paradigm. In: Annals of Mathematics and Artificial Intelligence. pp. 241–273 (1996)
11. Osorio, M., Zepeda, C.: Minimal Extended Generalized Answer Sets and their Applications. In: Proceedings of the Workshop in Logic, Language and Computation (LoLaCOM06), Fifth Mexican Internantional Conference on Artificial Intelligence. vol. 220 (2006)
12. Romero, M.C.: Tesis profesional de Licenciatura: Modelado de problemas de actualización utilizando un enfoque de actualización de secuencias de programas lógicos basados en Answer Set Programming (Ciencias de la Computación, Benemérita Universidad Autónoma de Puebla 2014)
13. You, J.H., Jia, L.Y.Y.X.: Adding Domain Dependent Knowledge into Answer Set Programs for Planning. In: Logic Programming, Lecture Notes in Computer Science. vol. 3132 (2004)
14. Zepeda, C.: Evacuation Planning using Answer Set Programming. Ph.D. thesis, Universidad de las Americas, Puebla and Institut National des Sciences Appliquées de Lyon (2005)

# Detección de barras rotas en motores de inducción utilizando SMCSA (Square Motor Current Signature Analysis)

David Alejandro Fernández Tavitas, Juan Pablo Nieto González

Corporación Mexicana de Investigación en Materiales, S.A. de C.V COMIMSA

david.fdz@comimsa.com, juan.nieto@comimsa.com

**Resumen.** Los motores de inducción actualmente son la máquina más ampliamente usada en el campo industrial, por su robustez y fácil instalación. Es por eso que la detección de fallas toma un papel importante para un correcto mantenimiento y por consecuente el alargamiento de la vida útil del motor. Una de las principales técnicas existentes para la detección de fallas en motores de inducción es MCSA (*motor current signature analysis*). Dicha técnica basa su funcionamiento en el análisis de la corriente de un motor y observación de patrones que nos indican la existencia de una falla. En este artículo se presenta una comparación entre las técnicas MCSA y SMCSA (*Square motor current signature analysis*) y la ventaja que presenta la técnica para la detección de pequeñas fallas.

**Palabras clave:** Motores de inducción, barras rotas. SMCSA.

## 1. Introducción

Uno de los elementos más importantes dentro del ámbito industrial son los motores de inducción. Dichas máquinas usualmente trabajan bajo severas condiciones (térmicas, eléctricas, mecánicas, ambientales, etc.) Lo cual puede provocar la presencia de fallas en el estator y/o el rotor.

Estadísticas de fallas [1] reportan que los componentes del motor que tienden a fallar con más frecuencia son: Estator (38%), Rotor (10%), Rodamientos (40%), Otros (12%)

La ruptura de barras en el rotor representa un 10% de las fallas de un motor, dependiendo de su severidad pueden ir desde el mal funcionamiento hasta la detención total del motor y a su vez un paro de línea causando pérdidas económicas dentro de la empresa. [2, 3] Existen diversas técnicas y metodologías para la detección de fallas en el motor como la Impedancia de secuencias negativas, análisis del espectro de frecuencias, el vector de Perk [4], la transformada de Hilbert [1] análisis de la firma eléctrica [6] entre otros. Una de las técnicas más ampliamente usadas es MCSA, consiste en el análisis de la corriente del motor detectando bandas

laterales sobre la frecuencia principal que señalan la presencia de una anomalía. Sin embargo la técnica MCSA presenta limitaciones para detectar fallas en caso de que la falla sea muy pequeña o la corriente del motor sea muy reducida. En este artículo se realiza la comparación de la técnica MCSA con SMCSA para la detección de barras rotas del rotor, presentando ventaja la segunda técnica para corrientes de reducido tamaño o para un número reducido de barras.

La experimentación fue realizada mediante la simulación de un motor trifásico de 220v, 30Hz y poniendo en cortocircuito las resistencias de una fase en representación de la perdida de corriente por las barras rotas del motor.

El presente artículo está organizado de la siguiente forma: la sección 2 y 3 presentan una breve explicación de las teorías a comparar, la sección número 4 muestra los preliminares matemáticos utilizados para el pos-procesamiento de las señales obtenidas, la sección número 5 señala la fase de experimentación, en la sección número 6 los resultados obtenidos y la sección número 7 presenta las conclusiones del proyecto.

## 2. MCSA (Motor Current Signature Analysis)

Para la detección de fallas del rotor es necesario utilizar un espectro de frecuencia, el cual contiene importante información acerca del funcionamiento del motor. Al observar los armónicos obtenidos de la corriente es posible detectar anomalías o variaciones presentes en la frecuencia la cual indica un mal funcionamiento, dichos armónicos muestran diferentes espectros dependiendo de la falla del motor así como de su severidad [6, 7].

### 2.1. Estado de operación normal

Considerando a un motor trifásico en condiciones ideales de voltaje y potencia podemos analizar la corriente en estado de operación normal mediante la fórmula (1)

$$i_a(t) = \sqrt{2} I_s \sin(\omega_s t - \varphi) \quad (1)$$

donde:
$\varphi$ Es el ángulo de fase entre el voltaje y la línea de corriente,
$f_s$ Es la frecuencia fundamental,
$\omega_s$ Es la pulsación fundamental $\omega_s = 2\pi f_s$.

El espectro de la corriente de la fase solo presentará el componente principal de la frecuencia.

### 2.2. Estado de falla

Manteniendo las mismas condiciones del motor es posible detectar la falla examinando los componentes laterales en el armónico donde son resaltados componentes en las frecuencias $(1 + 2s)f_s$ como se muestra en la Fig. 1. La corriente del motor es expresada por:

$$i_a(t) = I_{max}\cos(\omega t) + I_{lsb}\cos[(1-2s)\,\omega t] + I_{usb}\cos[(1+2s)\omega t] \qquad (2)$$

$I_{max}$ Es el valor máximo de la corriente fundamental.
$I_{lsb}$ Es el valor máximo del componente fundamental más pequeño.
$I_{usb}$ Es el valor máximo del componente fundamental más grande.
$s$ Es el deslizamiento.

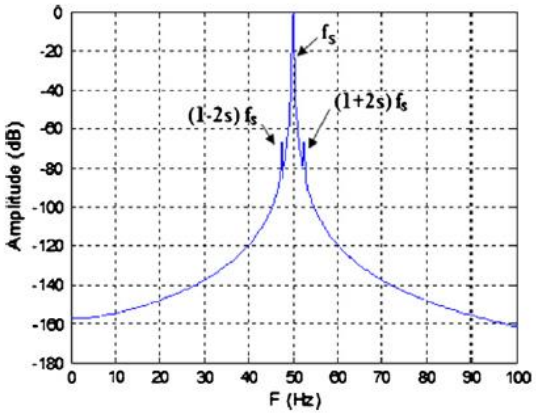



**Fig. 1.** Espectro de frecuencia barras rotas del rotor (adaptada de [8]).

## 3. SMCSA (Square Motor Current Signature Analysis)

Tal como en la técnica MCSA se realiza un análisis del espectro de frecuencias sobre la corriente del estator, sin embargo la técnica presentada utiliza el cuadrado de la corriente, lo cual facilita la detección de patrones indicadores de fallas existentes [8].

La técnica basa su metodología en 3 pasos principales. 1) La adquisición de la corriente de fase del estator en funcionamiento, 2) El procesamiento del cuadrado de la corriente obtenida mediante las fórmulas (3) y (4) y 3) El análisis del cuadrado de la corriente mediante espectros de frecuencia con el uso de la transformada de Fourier (sección 4.1).

### 3.1. Estado de operación normal

Mediante la fórmula (3) es posible hacer un procesamiento de la corriente de fase para conocer su comportamiento en su modo de operación normal.

$$i^2{}_a(t) = \frac{I^2{}_{max}}{2} + \frac{I^2{}_{max}\cos(\omega t)}{2} \qquad (3)$$

donde:

$I_{max}$ Es el valor máximo de la corriente fundamental,

$\omega$ Es la pulsación fundamental $\omega_s = 2\pi f_s$.

Mediante la técnica SMCSA es posible detectar en el armónico una única corriente fundamental en la frecuencia $2f_s$ adjunta a un componente DC como se presenta en la Fig. 2.

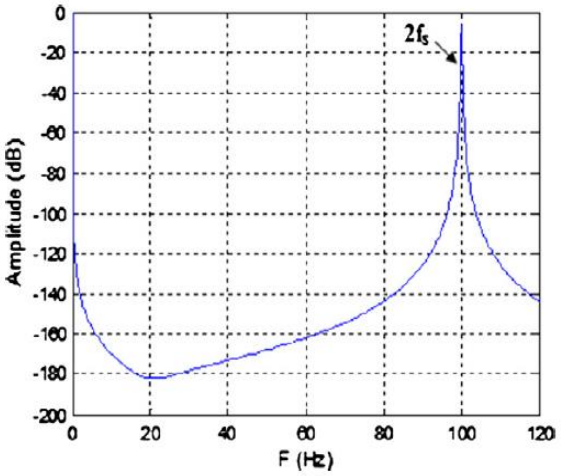



**Fig. 2.** Espectro de frecuencia modo de operación normal (adaptada de [8]).

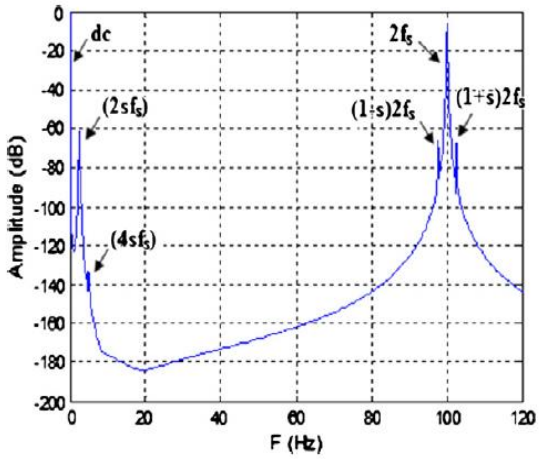



**Fig. 3.** Espectro de frecuencia modo de operación normal (adaptada de [8]).

### 3.2. Estado de falla

Bajo las mismas condiciones un motor de inducción que presenta falla en el rotor, específicamente con barras rotas presentes permite determinar la corriente mediante la fórmula (4), [8].

$$i^2{}_a(t) = \left(\frac{I^2{}_{max}}{2} + \frac{I^2{}_{lsb}}{2} + \frac{I^2{}_{usb}}{2}\right) + \left(\frac{I^2{}_{max}}{2} + I_{lsb}I_{usb}\right)\cos(2\omega t)$$

$$+ (I_{max}I_{lsb} + I_{max}I_{usb})\cos(\omega t) + I_{lsb}I_{usb}\cos(4s\omega t) + I_{max}I_{lsb}\cos(2(1-s)\omega t)$$

$$+ I_{max}I_{usb}\cos(2(1+s)\omega t) + \frac{I^2{}_{lsb}}{2}\cos(2(1-2s)\omega t) + \frac{I^2{}_{usb}}{2}\cos(2(1+2s)\omega t) \quad (4)$$

En la Fig. 3 se muestra un componente DC, un componente en la frecuencia $2f_s$, componentes laterales en las $2(1 \pm 2s)f_s$ y adicionalmente se muestran componentes laterales en las frecuencias $2sf_s$ y $4sf_s$ dependiendo de la severidad de la falla presente.

## 4. Preliminares matemáticos

### 4.1. Transformada de Fourier

La transformada de Fourier es una herramienta matemática utilizada para convertir una señal en el dominio del tiempo a una señal en el dominio de frecuencia con el fin de observar el comportamiento de una función en un tiempo específico [10]. Es denotada $h(t)$ Como una señal continua en el tiempo la cual se encuentra entre 2 señales consecutivas en $T$ segundos, es posible obtener su transformada mediante la fórmula (5)

$$H(\omega) = \sum_{n=-\infty}^{\infty} h(nT)e^{-j\omega n} \quad (5)$$

donde

$\omega$ es la frecuencia normalizada en radianes ($\omega = \Omega T$),

$\Omega$ es la frecuencia en radianes/segundos ($\Omega = 2\pi f$),

$T$ es un periodo de tiempo.

### 4.2. Transformada rápida de Fourier

La transformada rápida de Fourier es un algoritmo que permite evaluar los componentes armónicos de una señal discreta periódica, puede ser evaluada mediante la fórmula (6)

$$H(k) = \sum_{n=0}^{N-1} h(n) e^{\frac{j2\pi}{N} nk} \tag{6}$$

donde
$N$ es el número de coeficientes,
$k$ es un integrador $k = 0, \dots, N-1$.

### 4.3. Velocidad de deslizamiento

El voltaje inducido en una barra del rotor de un motor de inducción depende de la velocidad del rotor con respecto a los campos magnéticos .El comportamiento de un motor de inducción depende del voltaje y la corriente del rotor, conocido como la velocidad relativa. [11] menciona que el deslizamiento se define como la velocidad relativa expresada sobre una base por unidad o porcentaje, el deslizamiento se expresa mediante la fórmula (7)

$$s = \frac{n_{sinc} - n_m}{n_{sinc}} (\times 100\%) \tag{7}$$

donde $s$ es el deslizamiento,
$n_{sinc}$ velocidad de los campos magnéticos,
$n_m$ velocidad del eje del motor.

## 5. Fase de experimentación

Para la experimentación fue realizada la simulación de un motor trifásico de inducción de 220V, 2.5A, dos polos, y una frecuencia de 30 Hz, conexión tipo estrella. En una de las fases del motor se establecieron una serie de cortocircuitos entre las resistencias de dicha fase las cuales fueron utilizadas para simular barras rotas en el rotor mediante el cambio de fluctuación de la corriente, además se definieron diferentes puntos de conexión para cada una de las resistencias, denotando así el grado de severidad en las fallas, Fig. 4 [10].

**Tabla 1.** Puntos de conexion y severidad de fallas.

| Severidad | Porcentaje de corto | Puntos de conexión |
|---|---|---|
| Sin falla | 0% | 1-2, 3-4, 5-6 |
| Severidad baja | 5% | 3-4-5-6, 7-8 |
| Severidad alta | 10% | 3-4-7-8, 5-6 |

Se hicieron pruebas en estado de operación normal y estado de falla para diversos niveles de severidad aplicando ambas técnicas de detección.

Los valores adquiridos del motor fueron pre-procesados mediante el análisis de corriente con el uso de la técnica MCSA y analizadas en base a la frecuencia mediante

el uso de la transformada rápida de Fourier (FFT), en donde se resaltan la frecuencia principal y sus componentes laterales (definidos como aumentos o decrementos de amplitud en la corriente presentes en los costados de frecuencia principal), para cada uno de los grados de operación. Posteriormente se realizó el mismo procedimiento para la técnica SMCSA poniendo en comparación los resultados obtenidos por ambas metodologías.

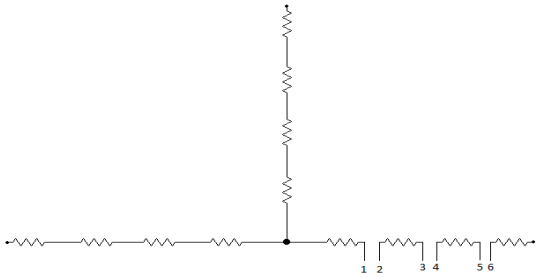



**Fig. 4.** Conexión estrella con cortocircuitos presentes en una de las fases.

## 6. Resultados del experimento

Con el propósito de demostrar la eficiencia de una técnica sobre otra se realizaron diversas pruebas para cada una de las metodologías mencionadas obteniendo los siguientes resultados.

En la Fig. 5 se muestran los espectros de frecuencia en estado de operación normal para ambas técnicas, en donde es posible detectar un aumento de frecuencia en el valor $f_s$ para la técnica MCSA, y un aumento de frecuencia en $2f_s$ así como un componente DC para la técnica SMCSA.

En la figura (6) fue realizado el mismo procedimiento, análisis de ambas técnicas para un grado de severidad alto (10%) presentando en ambas técnicas buenos resultados, siendo favorecida la técnica SMCSA en referencia a la resolución del análisis. En la figura (7) se puede observar un aumento de amplitud en $f_s$ y componentes laterales en las frecuencias $(1-2s)f_s$ y $(1+2s)f_s$ para la técnica MCSA, indicando la presencia de una falla en el motor, específicamente barras rotas en el rotor. En el caso de la técnica SMCSA es posible detectar un aumento de amplitud en el cuadrado de la corriente equivalente a $2f_s$ y componentes laterales en las frecuencias $(1 \pm 2s), 2f_s(1 \pm 4s), 2f_s(1 \pm 6s)f_s$.

La ventaja que muestra la técnica SMCSA sobre MCSA se presenta al momento de trabajar con niveles de severidad bajos en donde el análisis por MCSA no es capaz de resaltar señales que indiquen la presencia de una falla mientras que el cuadrado de dicha técnica muestra componentes relevantes para detección de la falla. En la figura (7) se presenta el análisis de frecuencias con un grado de severidad al 5% y un nivel de corriente bajo denotando aumentos de amplitud en los valores $2f_s$, $4f_s$, $6f_s$ y $8f_s$ para la técnica SMCSA los cuales nos indican la existencia de barras rotas en el rotor.

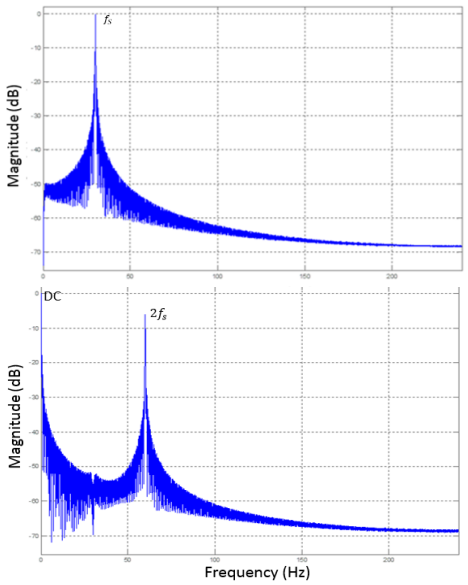



**Fig. 5.** Espectro de frecuencia estado de operación normal a) MCSA b) SMCSA.

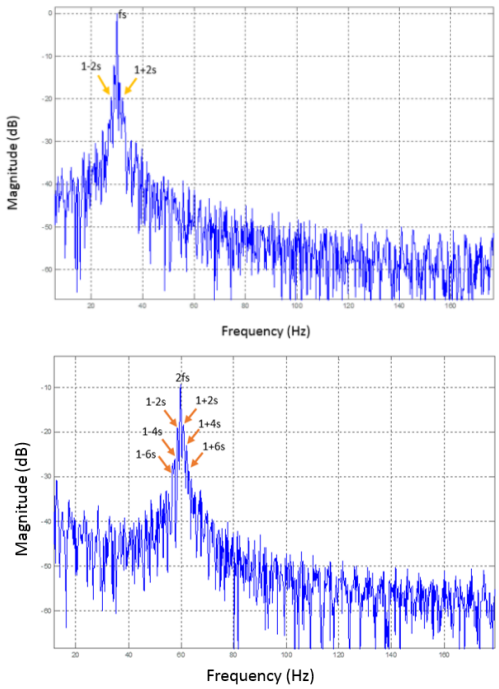



**Fig. 6.** Espectro de frecuencia 10% de severidad a) MCSA b) SMCSA.

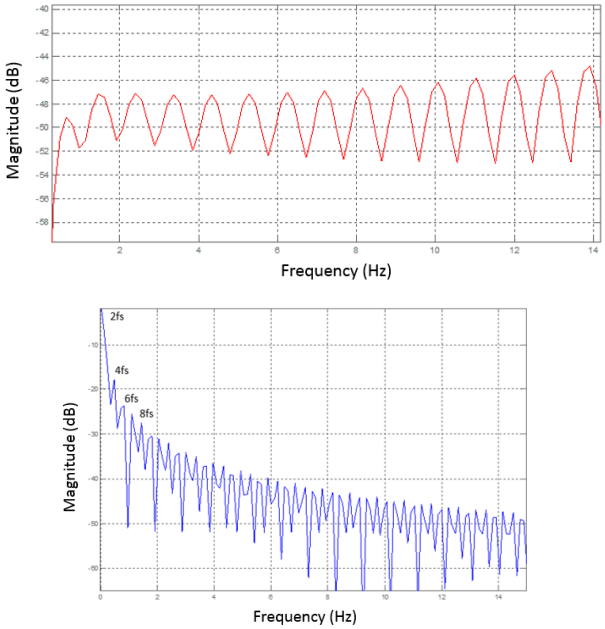



**Fig. 7.** Espectro de frecuencia 5% de severidad a) MCSA b) SMCSA.

## 7. Conclusiones

En este artículo se presenta una comparación entre las técnicas MCSA y SMCSA para el análisis de fallas en motores eléctricos, específicamente detección de barras rotas en el rotor. Las técnicas expuestas basan su funcionamiento en 3 pasos principales, la adquisición de la corriente de fase del motor, el procesamiento de la corriente para la obtención de los componentes principales y finalmente el análisis de las corrientes obtenidas del procesamiento de ellas.

Fueron presentadas diversos modos de operación, desde modo de operación normal hasta modo de falla en diversos niveles de severidad, demostrando resultados notables en ventaja de la técnica SMCSA para fallas con altos niveles de severidad, en los cuales los patrones que nos indican la existencia de una falla se presentan en mayor cantidad, demostrando una mejor resolución al momento del análisis, y principalmente para fallas con bajos niveles de severidad o de condiciones de corriente reducida, en los cuales la técnica MCSA no es capaz de señalar la presencia de alguna falla, en caso contrario, la técnica SMCSA presenta patrones en frecuencias específicas, las cuales indican la existencia de una falla.

## Referencias

1. R. Puche-Panadero, M. Pineda-Sanchez, M. Rierra-Guasp, J. Roger-Folch, E. Hurtado-Pérez and J. Pérez-Cruz: Improved Resolution of the MCSA Method Via Hilbert

Transform, Enabling the Diagnosis of Rotor Asymmetries at Very Low Slip. IEEE Transactions on Energy Conversion, (2009) pp. 52–59.

2. S. Guedidi, S.E.Zouzou, W. Laala, M. Sahraoui, K. Yahia: Broken Bar Fault Diagnosis of Induction Motors Using MCSA and Neural Network. In: 2011 IEEE International Symposium on Diagnostics for Electric Machines, Power Electronics & Drives (SDEMPED), (2011) pp. 632–637.

3. Jose Rangel-Magdaleno, Juan Ramirez-Cortes and Hayde Peregrina-Barreto: Broken Bars Detection of Induction Motor Using MCSA and Mathematical Morphology: An Experimental Study. In: 2013 IEEE International Instrumentation and Measurment Technology Conference (I2MTC), (2013) pp. 825–829.

4. C. J. Verucchi and G. G. Acosta: Técnicas de Detección y Diagnóstico de Fallos en Máquinas Eléctricas de Inducción. En: IEEE Latin American Transactions (2007).

5. Prabhakar Neti, Manoj R. Shah, Karim Younsi, John Krahn, Joe Yingneng Zhou and C. David Whitefield: Motor Current Signature Analysis During Accelerated Life Testing of Form Wound Induction Motors. In: 2010 IEEE International Power Modulator and High Voltage Conference (IPMHVC), (2010) pp. 106–109.

6. Prabhakar Neti, Pinjia Zhang, Manoj Shah and Karim Younsi: Electrical Signature Analysis Based Online Monitoring of Drive-trains for Doubly-fed Wind Generators. In: IECON 2012 - 38th Annual Conference on IEEE Industrial Electronics Society, (2012) pp. 1764–1769.

7. G. Didier, E. Ternisien, O. Caspary and H. Razik: A new Approach to detect broken rotor bars in induction machnes by current spectrum analysis. Mechanical Systems and Signal Processing, Elsevier, (2007) pp. 1127–1142.

8. R.Saravana Kumar, Dr K.K.Ray and K.Vinoth Kumar: Fault Diagnosis of Industrial Drives Using MCSA Techniques. In: 2009 International Conference on Control, Automation, Communication and Energy Conservation, INCACEC, (2009) pp. 1–7.

9. V. Fernão Pires, Manuel Kadivonga, J.F. Martins and A.J. Pires: Motor square current signature analysis for induction motor rotor diagnosis. Measurment, (2013) pp. 942–948.

10. Reddy, V. U.: On Fast Fourier Transform, A popular tool for spectrum analysis. General Article (1998).

11. Stephen J. Chapman: Máquinas Eléctricas. Australia: Mc. Graw Hill, Tercera Edición, (2000).

12. Jee-Hoon Jung, Lee Jong-Jae and Bong-Hwan Kwon: Online Diagnosis of Induction Motors Using MCSA. IEEE Transactions on Industrial Electronics, Vol. 53, (2006) pp. 1842–1852.

13. M. Sahraoui, S.E. Zouzou, A. Ghoggal, S. Guedidi and H. Derghal: An improved Algorithm for Detection of Rotor Faults in Squirrel Cage Induction Motors Based on a New Fault Indicator. In: 2012 XXth International Conference ON Electrical Machines (ICEM), (2012) pp. 1572–1578.

# Reviewing Committee

Noé Alejandro Castro-Sánchez
Jair Cervantes
William De La Cruz De Los Santos
Anilu Franco-Arcega
Sofia N. Galicia-Haro
Alexander Gelbukh
David Gonzalez
Miguel Gonzalez-Mendoza
Oscar Herrera
Hector Jimenez Salazar
Asdrúbal López
Lourdes Martínez
Sabino Miranda
Raul Monroy
Antonio Neme
Obdulia Pichardo-Lagunas
Rafael Rojas-Hernández
Grigori Sidorov
Israel Tabarez
Valentín Trujillo Mora
Edgar Vallejo
Nestor Velasco Bermeo

# Additional reviewers

Jorge Rodriguez Ruiz
Roberto Alonso
Soujanya Poria
José Camiña
Victor Ferman
Gabriela Ramírez-De-La-Rosa

Impreso en los Talleres Gráficos
de la Dirección de Publicaciones
del Instituto Politécnico Nacional
Tresguerras 27, Centro Histórico, México, D.F.
mayo de 2014
Printing 500 / Edición 500 ejemplares