UNIVERSIDAD MICHOACANA

DE SAN NICOLAS DE HIDALGO

FACULTAD DE

INGENIERIA ELECTRICA

DETECCION DE ANOMALIAS EN REDES DE COMPUTO UTILIZANDO MODELOS OCULTOS DE MARKOV Y PROGRAMACION EVOLUTIVA

TESIS

QUE PARA OBTENER EL GRADO DE:

DOCTOR EN CIENCIAS EN INGENIERIA ELECTRICA

PRESENTA:

ANASTACIO ANTOLINO HERNANDEZ

DR. JUAN J. FLORES ROMERO

DIRECTOR DE TESIS

DR. JUAN M. GARCIA GARCIA

CO-DIRECTOR DE TESIS

FEBRERO 2011

DETECCIÓN DE ANOMALÍAS EN REDES DE CÓMPUTO UTILIZANDO MODELOS OCULTOS DE MARKOV Y PROGRAMACIÓN EVOLUTIVA

TESIS

Que para obtener el grado de

DOCTOR EN CIENCIAS EN INGENIERÍA ELÉCTRICA

presenta

Anastacio Antolino Hernández

Dr. Juan J. Flores Romero

Director de Tesis

Dr. Juan M. García García

Co-Director de Tesis

Universidad Michoacana de San Nicolás de Hidalgo

Febrero 2011

Dedicación

Dedico este trabajo al amor, apoyo, tolerancia y esfuerzo de toda mi familia. Empezando por mi amada y adorada esposa Betzy, a nuestros hijos: Cristopher, Jonathan y Alexander, así como a su abuelita Betzy.

A mi padres, Juan Antolino Torres y Natividad Hernández Montes[‡], y a todos mis hermanos que se encuentran, al igual que mi padre, distantes geográficamente.

Y sobre todo, a Dios y a la vida por haberme dado esta oportunidad de crecer como persona y profesionalmente en esta área del conocimiento.

¡Gracias!

Agradecimientos

A Dios, a mi Familia y a mis Padres.

Deseo agradecer a mis asesores:

Dr. Juan José Flores Romero por todo su apoyo, amistad, tiempo, asesoría y sabios consejos que me ayudaron a dar buen término este trabajo de investigación.

Al Dr. Juan Manuel García García, amigo, compañero de trabajo y asesor, que cuando acudía a él con algún problema o duda, siempre me daba una idea o me orientaba por dónde buscar una posible solución.

También deseo manifestar mi agradecimiento al Instituto Tecnológico de Morelia por el permiso otorgado para realizar mis estudios de doctorado, así como a la Academia del Departamento de Sistemas, todos ellos amigos y compañeros de trabajo.

Un especial agradecimiento al Ing. Antonio Chávez Garibay encargado del Centro de Cómputo Universitario, quien nos proporcionó los datos del monitoreo de la red, con los cuales probamos y trabajamos en nuestro proyecto. Así como al profesores del tecnológico Emilio Guzmán Pulido, que me apoyó en algunas cuestiones de dudas sobre análisis de datos, y a Heberto Ferreira Medina por su disposición y ayuda desinteresada.

Deseo también agradecer al Conacyt por la beca de apoyo a mis estudios.

A la comunidad de la División de Estudios de Posgrado de la Facultad de Ingeniería Eléctrica, por haberlos conocido y convivido. Entre los que se encuentran todos los Doctores-Investigadores, compañeros estudiantes de posgrado, y personal de apoyo: como las Sras. Yolanda y Tere; y Jorge.

Resumen

La seguridad de la información es una necesidad importante y creciente. Los ataques a los sistemas informáticos son constantes y cada vez en mayor número, a la vez que se vuelven mas complejos y difíciles de detectar. Los sistemas que tratan de detectarlos y prevenirlos están avanzando, pero a paso muy lento. Los esquemas utilizados más comunes y sobre los cuales se basan los sistemas de detección son: los basados en patrones o firmas y los basados en anomalías. Cada esquema tiene sus ventajas y desventas, por ejemplo los basados en patrones son rápidos y cuentan con conocimiento a priori de las amenazas informáticas, pero son incapaces de detectar nuevos ataques o variaciones de ataques conocidos. Los esquemas basados en anomalías utilizan un conjunto de métricas, las cuales perfilan el comportamiento normal del sistema, y cualquier desviación que supere un valor de umbral, respecto al perfil establecido, será tratado como anomalía. Este esquema tiene la desventaja que puede presentar falsos positivos; pero la ventaja que resaltan es que detectan nuevos y variaciones de ataques conocidos.

Este trabajo de tesis contribuye con un esquema de detección de anomalías, basado en el consumo de ancho de banda de red, utilizado en la subred de la División de Estudios de Posgrado de la Facultad de Ingeniería Eléctrica, de la Universidad Michoacana de San Nicolás de Hidalgo (UMSNH). Se determina el modelo de comportamiento normal de uso de ancho de banda, de acuerdo a los datos proporcionados por el centro de cómputo de la universidad. El modelo generado se utiliza en la detección de anomalías o variaciones en el consumo del ancho de banda de red. La presencia de una anomalía indica que se está haciendo un mal uso de la red (existencia de virus, gusanos, denegación de servicio, etc). Las variables a utilizar principalmente serán: la del consumo de ancho de banda de red de entrada, y la del consumo de ancho de banda de red de salida. Pero cabe destacar que el sistema está diseñado para analizar dos o más métricas del comportamiento del sistema.

Otra contribución de esta tesis es la forma de diseñar la arquitectura y optimización de los parámetros de los HMMs (del inglés Hidden Markov Models, Modelos Ocultos de Markov), de manera automática utilizando Programación Evolutiva (EP). Los HMMs son creados y evolucionados para modelar el perfil de comportamiento normal del sistema. Los HMMs se utilizan como modelos probabilísticos que capturan la dinámica del sistema para la deteccción de secuencias anómalas.

Otro punto importante de contribución es que en la tesis se trabajó con un alfabeto real, en vez de uno discreto como en todas las investigaciones sobre HMMs que se mencionan en esta tesis.

La tesis muestra que los HMMs evolucionados, proporcionan un esquema para la detección de anomalías basado en las secuencias de observación del sistema modelado. El HMM es capaz de analizar tanto secuencias de observación univariadas, como secuencias multivariadas.

Abstract

Information security is an important and growing need. Attacks on computer systems are constant and increasing in number, while they become more complex and difficult to detect. Unlike systems that try to detect and prevent, anomaly detection systems advance in shorter steps. The most common schemes used which serve as a basis for detection systems, include *pattern* or *signature-based* and *anomaly-based*.

Each scheme has its advantages and disadvantages, for example pattern-based schemes are fast and have a priori knowledge of the threats, but are unable to detect new attacks or even variations of known attacks. Anomaly-based schemes using a set of metrics, which outline the normal system behavior and any deviation that exceeds a threshold value, corresponding to the profile established, will be treated as an anomaly. This scheme has the disadvantage that false alarms can occur; but the advantage they present is to detect new attacks and also detect variations of known attacks.

This thesis contributes with an anomaly-based scheme that monitors the bandwidth consumption of subnetwork, used in the División de Estudios de Posgrado de la Facultad of Engeniería Electrica de la Universidad Michoacana de San Nicolás de Hidalgo (UMSNH). Normal behavior model is based on bandwidth consumption of subnetwork and the generated model is used to detect anomalies or changes in the bandwidth consumption of network. The presence of an anomaly indicates that it is misusing the network (viruses, worms, denial of service, etc). The variables to be used mainly are: the bandwidth consumption of network IN and OUT of the network. But it is noteworthy that the system is designed to analyze two or more metrics of the system behavior.

Another contribution of this work thesis is to design the architecture and optimizing the parameters of Hidden Markov Models (HMMs), automatically using Evolutionary Programming (EP). HMMs are created and evolved to shape the profile of normal system behavior. The HMMs are used as probabilistic models that capture the dynamics of the system to detect anomalous sequences.

Another important contribution is this work thesis treated with a real alphabet, rather than discrete as in all investigations mentioned about HMMs in this thesis.

The thesis shows that HMMs evolved provide a scheme for anomaly detection based on observation sequences of the system to be modeled. Furthermore, HMMs are able

to analyze both univariate observation sequences, such as multivariate sequences.

Lista de Símbolos

HMM. Modelo Oculto de Markov (del inglés Hidden Markov Model).

HMMCU. HMM continuo univariado.

HMMCM. HMM continuo multivariado.

N. Número de estados del HMM.

A. Matriz de transiciones entre estados de un HMM.

B. Matriz de distribución de probabilidad.

Π. Vector de distribución de probabilidad.

V. Conjunto de símbolos posibles a observar.

P(x). Valor de probabilidad.

O. Secuencia de símbolos observados.

Q. Secuencia de estados.

Ω. Conjunto de operadores evolutivos.

λ. Representa a un modelo HMM.

C. Conjunto de cromosomas.

F(x). Función de distribución de probabilidad.

$f_{\mu\sigma}(x)$. Función de distribución normal.

$f_{\mu\Sigma}(x_1, \ldots, x_d)$. Función de multidistribución gaussiana.

σ. Desviación estándar.

Σ. Matriz de covarianzas.

μ. Media.

t. Tiempo.

d. Dimensionalidad.

X. Variable aleatoria.

x. Valor de la variable aleatoria.

Contenido

Lista de Figuras

Lista de Tablas

Lista de Algoritmos

Capítulo 1

Introducción

En este capítulo se dará una breve introducción a la definición de los HMMs (Hidden Markov Models, Modelos Ocultos de Markov) y se describirán brevemente los Algoritmos Genéticos (GAs), se mencionará el objetivo del presente trabajo, se describirá la justificación, la propuesta de tesis del presente documento y la motivación del desarrollo de la presente investigación.

1.1. Modelos ocultos de Markov

La teoría de los HMMs (Hidden Markov Models, Modelos Ocultos de Markov) fue dada a conocer a finales de los 60s y a inicios de los 70s; los HMMs fueron aplicados inicialmente al reconocimiento del habla. Los HMMs y sus variaciones han sido aplicados, desde entonces, a otras áreas. Son procesos estocásticos que pueden ser utilizados para analizar series de tiempo estadísticas. Los HMMs describen un doble proceso estocástico, donde cada estado del HMM representa alguna condición inobservable del sistema a ser modelado. Cada estado expresa la probabilidad de obtener cierto valor, y existen las probabilidades de transitar a otros estados. Se puede tener en cada estado una distribución de probabilidad diferente, el modelo puede también cambiar el estado inicial en cada ejecución. El sistema no requiere de datos anómalos para "aprender" o distinguirlos de los normales, ni requiere de reglas predictivas explícitas [Warrender99].

Los HMMs son considerados como uno de los componentes básicos de los sistemas de reconocimiento del habla [Rabiner86], [Kurimo92]. Además, los HMMs están basados en máquinas de estado finito o automáta finito probabilístico usados para modelar secuencias

estocásticas [Won05], [Missaoui08]. Los HMMs tienen muchas aplicaciones como en procesamiento de señales [Kirshner05], reconocimiento de patrones [Deng06], secuencias de ADN [Pachter01], reconocimiento de gestos y el habla [Nicholl08], así como en la detección de anomalías [Brewer06].

1.1.1. Componentes de un HMM

Un HMM está formado por un número finito de estados conectados por transiciones, como se observa en la Figura 1.1, donde el número del estado se encuentra encerrado en un rectángulo, y las flechas indican las transiciones entre los estados. Cada transición o flecha está asociada a un peso o valor de probabilidad. El HMM puede generar una secuencia de observaciones dependiendo de las probabilidades del estado inicial y de las transiciones hechas entre los estados.

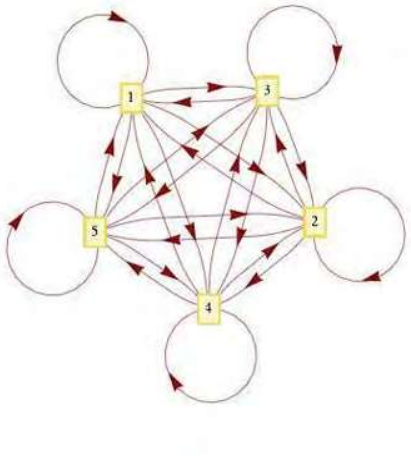

Figura 1.1: Representación general de un HMM.

El Modelo de Markov es oculto porque los estados no se observan directamente. Es decir, no se sabe qué secuencia de observaciones lleva a qué estado. Un HMM está definido básicamente por los parámetros siguientes:

$A = a_{ij}$ es la matriz de probabilidad de transición entre estados. Donde $a_{i,j} = \mathrm{P}(q_{t+1} = \mathrm{S}_j \mid q_t = \mathrm{S}_i)$, $1 \leq \mathrm{i}, \mathrm{j} \leq N$. q_t es el estado actual en el tiempo t.

$B = b_{jk}$ es la matriz de probabilidad de emisión, indicando la probabilidad de observar un símbolo k, en un estado j. Es decir, $b_{j,k} = \mathrm{P}(v_k \text{ en } t \mid q_t = S_j)$. v_k es observar

el símbolo k del alfabeto V.

$\Pi = \pi_i$ es la distribución de probabilidad de iniciar en el estado i. Esto es, $\pi_i = \mathrm{P}(S_i$ en $t = 1)$.

Por lo que un HMM puede ser definido por la tripleta de la Ecuación 1.1 [Nicholl08]:

$$\lambda = (A, B, \Pi) \tag{1.1}$$

Es importante mencionar que en un HMM, la buena estimación de los parámetros del modelo afecta en el rendimiento del proceso de reconocimiento o detección [Korayem07], [Seymore99]. Los parámetros de un HMM pueden ser determinados utilizando un proceso iterativo, a este proceso de optimización de parámetros se le llama "entrenamiento". El algoritmo Baum-Welch [Rabiner89], [Welch03], es un método iterativo aplicado a la re-estimación de los parámetros del HMM. El algoritmo de Baum-Welch, siendo un método basado en gradiente, puede converger en un óptimo local. Se pueden usar técnicas de búsqueda global para optimizar los parámetros de los HMMs.

1.2. Algoritmos genéticos

La *Computación Evolutiva* es una rama de la Inteligencia Artificial que involucra problemas de optimización combinatoria; se inspira en los mecanismos de la Evolución Biológica. Durante los años 60s y 70s, varias corrientes de investigación independientes comenzaron a formar lo que ahora se conoce como *computación evolutiva*: Programación Evolutiva, Estrategias Evolutivas y Algoritmos Genéticos [Jacob01].

En los años 70s, de la mano de John Henry Holland, surgieron los *Algoritmos Genéticos* (GAs, Genetic Algorithms), y son llamados así porque se inspiran en la evolución biológica de la naturaleza. El término GA describe la idea básica: técnicas algorítmicas, inspiradas por principios genéticos, que son usadas para simular procesos evolutivos [Jacob01].

Estos algoritmos hacen evolucionar una población de individuos sometiéndola a acciones aleatorias semejantes a las que actúan en la evolución biológica, así como también a una selección de acuerdo con algún criterio. Es en función de este criterio que se decide cuales son los individuos mejor adaptados y cuales los menos aptos [Barrera08]. Los GAs son procesos de búsqueda basados en las leyes de la selección natural y leyes genéticas. Simulan individuos en un ambiente natural, donde los mecanismos de selección natural hacen de los

individuos más fuertes, más probables de ser ganadores en un ambiente de competencias [Korayem07].

Los GAs son métodos de búsqueda estocástica que pueden efectuar una búsqueda global dentro de un espacio de soluciones definidas [Ogawa06]. Estas técnicas de optimización global pueden ser usadas para optimizar los parámetros de un HMM [Korayem07] y han sido propuestas y utilizadas por M. Slimane [Slimane96] desde el año de 1995. Otro trabajo muy citado, con respecto a la optimización de los parámetros de un HMM utilizando GAs, es el de C. W. Chau [Chau97], el proceso consiste en optimizar un HMM de 5 estados para ser utilizado en el reconocimiento de voz. Sólo se utiliza en el cromosoma la matriz de transiciones y la de distribución de probabilidad para los símbolos discretos.

Los casos de utilización de GAs para entrenar HMMs son relativamente raros, en comparación con el uso de GAs en el entrenamiento de Redes de Neuronas [Won05]. Existen trabajos recientes que utilizan a los GAs en la optimización de los parámetros, como se verá más adelante.

La *Programación Evolutiva* (EP, Evolutionary Programming) nació en la década de 1960 y su creador fue el Dr. Lawrence J. Fogel, considerado el "padre de la inteligencia computacional". Este desarrollo comenzó como un esfuerzo encaminado a crear inteligencia artificial basado en la evolución de máquinas de estado finito.

La programación evolutiva toma como base la siguiente afirmación: si las poblaciones naturales se adaptan a su entorno por evolución, entonces podemos modificar selectivamente las estrategias de Solución de Problemas, codificadas como programas, y progresivamente ajustar estos programas a una tarea predefinida con un conjunto de restricciones del medio ambiente.

1.3. Seguridad informática

La seguridad de la información no es una cuestión reciente, existen varios casos en la historia –sobre todo en las guerras mundiales– que hacen mención a la protección de la información. La preocupación por la seguridad de la información en una red local o en Internet nace desde el surgimiento de las redes de comunicación. Es un tema que los mismos creadores de las redes habían considerado, pero no es hasta 1980 cuando se fundamentan las bases y términos de la seguridad de la información [Spafford08].

Entre los muchos retos que un administrador de red enfrenta, se encuentra la *Se-*

guridad Informática, que es necesaria pero compleja. Es necesaria porque sin ningún sistema de seguridad apropiado la utilización de recursos y el intercambio de información no se puede lograr efectivamente. Es compleja porque actualmente no existe un sistema que cubra todos los requisitos y necesidades a un cien por ciento en cuestiones de seguridad informática. En resumen, podemos decir que el objetivo de cualquier sistema de seguridad es la protección contra intrusiones o anomalías, de acuerdo a sus propiedades y características [He97].

1.3.1. Objetivos de la seguridad informática

Mantener un sistema seguro o confiable, según Esquivel [Esquivel04], significa garantizar los siguientes aspectos: *confidencialidad*, *integridad*, *disponibilidad*; así como también *consistencia*, *autenticidad*, *control de acceso* y *auditoría*. A continuación se describen con más detalle estos aspectos:

Confidencialidad. Los recursos del sistema sólo deben ser accedidos por personas autorizadas para ello, y que esas personas autorizadas no van a convertir esa información en disponible para otras personas.

Integridad. Los archivos sólo pueden ser modificados por personas autorizadas y de una manera controlada.

Disponibilidad. Los recursos del sistema tienen que permanecer accesibles a las personas autorizadas.

Consistencia. Asegura que el sistema al igual que los datos, deben comportarse como uno espera que lo hagan.

Autenticidad. Permite que únicamente personas autorizadas ingresen al sistema, siempre y cuando comprueben que son usuarios legítimos del sistema.

Control de Acceso. Se refiere a que debe conocerse en todo momento quién entra al sistema y de dónde procede.

Auditoría. Indica que debe conocerse en todo momento las actividades de los usuarios dentro del sistema [Esquivel04].

Cumpliendo el sistema estas características, se garantiza que se encuentra fortalecido y resistente a ciertos intentos de ataques externos, pero de ninguna manera se puede garantizar que sea un sistema 100 % inviolable.

1.3.2. Detección de anomalías

El *objetivo principal* de la detección de anomalías es identificar, tanto como sea posible, eventos raros e interesantes con un mínimo de retardo y un número mínimo de alarmas falsas [Singh07].

Definición: El enfoque de la detección de anomalías puede ser visto, según [Dasgupta01], como el filtrado de desviaciones no-permitidas de las propiedades características del sistema monitoreado.

Otra definición de la detección de anomalías nos dice que es un proceso de identificación de comportamiento malicioso que se concentra hacia los sistemas de cómputo y sus recursos [Jemili07].

La *detección de anomalías* se enfoca a la detección de cambios en los patrones de utilización o en el cambio en el comportamiento del sistema. La detección de anomalías se puede efectuar construyendo un modelo estadístico que contenga métricas derivadas de la operación del sistema; el modelo resultante será capaz de registrar una intrusión si alguna métrica observada tiene una desviación estadística significativa del modelo [Islam05].

Otra definición de "anomalía", es una actividad sospechosa en el comportamiento de un usuario, proceso o servicio en comparación con el perfil o modelo clasificado como normal [Khreich09]. En otras palabras, un *sistema de detección de anomalías* usa un modelo de comportamiento "normal" del sistema en cuestión, para compararlo contra el comportamiento actual del sistema. Cualquier comportamiento que varíe de este modelo se considera una anomalía, y cualquier comportamiento que se acerque o coincida con el modelo, se considera "normal". En general, el comportamiento normal de un sistema de cómputo puede ser representado o modelado, observando sus propiedades en el tiempo.

1.3.3. Características de la detección de anomalías

La técnica de *Detección de Anomalías* se basa en un perfil del comportamiento normal del sujeto. Este perfil se compara contra el comportamiento observado del sujeto, e indica anomalía cuando el comportamiento observado del sujeto difiere significativamente de su perfil normal. Es decir, que se encarga de descubrir cambios en los patrones de utilización o del comportamiento usual del sistema.

Ventaja: Esta técnica se basa en el modelado de un patrón de comportamiento normal de un sistema (por ejemplo uso de ancho de banda de red, CPU, RAM, disco duro, etc.), y presenta la ventaja de detectar nuevos tipos de ataques [Shyu07]. Es decir, que tiene la habilidad de detectar ataques aún no registrados, variantes de ataques conocidos y desviaciones del uso normal de los sistemas [Ghosh98].

Desventaja: Estos sistemas presentan la desventaja de poseer un alto porcentaje de falsas alarmas (falsos positivos, reconociendo una anomalía cuando en realidad no existe) [Shyu07], [Ghosh98].

1.3.4. Problemas en la detección de anomalías

La proliferación de redes de computadoras heterogéneas tienen serias implicaciones en el problema de la detección de anomalías. La más importante en esta serie de implicaciones es el incremento en la oportunidad en el acceso no autorizado, a través de la conectividad de las redes [Jemili07].

Entre algunas amenazas informáticas existentes en las redes de comunicación, se pueden mencionar los intentos e intrusiones logradas por atacantes, virus y gusanos informáticos, denegación de servicio, etc. Existen otras técnicas utilizadas por atacantes para recabar información de una red para un uso ilícito; entre estas técnicas se pueden mencionar: barridos de las direcciones IPs de la red, barrido de protocolos de red, barrido de puertos y/o servicios de red, etc. Estas amenazas señaladas en su ejecución o en su actividad de intentar accesar recursos de la red, incrementan el uso o consumo de ancho de banda de la red. Con la existencia de una herramienta que analice el consumo de ancho de banda, se podrían detectar de manera exitosa el comportamiento anómalo de la red.

Cabe destacar que algunos atacantes hacen mimetismo, y realizan sus intentos de intrusión cuando existe más uso del ancho de banda de red, para poder pasar desapercibidos de los detectores de anomalías; o bien, intentan sus ataques de manera controlada y a intervalos de tiempo esporádicos para pasar también desapercibidos bajo los esquemas de detección. En estos tipos de ataques, es muy difícil detectar los intentos de intrusiones.

Otro problema presente en la detección de anomalías es que involucra grandes volúmenes de datos de series de tiempo, los cuales tienen un número significativo de entidades y actividades. Es decir, que para entrenar los modelos que detectarán las posibles anomalías se requiere una serie de datos que representen el comportamiento normal del sis-

tema en cuestión. De estos datos se selecciona únicamente los datos de los cuales aprenderá o tomará en cuenta el sistema de detección.

La detección de anomalías es un problema de gran significado para la seguridad de los sistemas de información, especialmente en vista del incremento de los accidentes mundiales de ciber-ataques en infraestructuras críticas [Mukkamala07].

1.4. Sistemas empleados en la detección de anomalías

Varios algoritmos y técnicas se han desarrollado para identificar diferentes tipos de intrusiones de red, ya que las redes crecen en importancia y tamaño, y se necesita el monitoreo efectivo de la actividad para prevenir accesos ilícitos. La detección de anomalías provee una capa de defensa la cual vigila el tráfico de red para identificar actividad sospechosa o patrones que sugieran un tráfico hostil en potencia.

Se sabe que las técnicas comunes de intrusión empiezan con el escaneo de un conjunto de hosts y puertos para explorar la topología de una red o localizar servidores vulnerables. Esto a menudo es la antesala a ataques más dañinos. La detección correcta y a tiempo de este tipo de anomalías en una red es útil para identificar intentos de intrusiones antes de que cualquier actividad maliciosa pueda ocurrir [Zhang06].

En la Detección de Anomalías se han desarrollado sistemas basados en diversas técnicas o áreas del conocimiento, e incluso se han hecho combinaciones de varias de éstas, para cubrir mejor el objetivo de la seguridad informática. A continuación se describen algunos trabajos que tienen que ver con la detección de anomalías y diversas áreas del conocimiento.

Por ejemplo, algunos trabajos se han enfocado en el uso de la *Inteligencia Artificial* para ciertos sistemas de seguridad computacional. A esta clase de seguridad se le denomina *"Sistemas de Seguridad Inteligente"*, los cuales se han empleado en la detección de intrusos y virus, análisis en tiempo real de bitácoras, etc. Pero donde más se les ha utilizado a estos sistemas, es en los Sistemas de Detección de Intrusos donde se ha combinado la Ingeniería del conocimiento y la Seguridad Informática [Toure94].

Redes Bayesianas. Herramientas de modelado gráfico usadas para problemas de decisión con incertidumbre, que toman como base la regla de Bayes. Jemili [Jemili07] propone un sistema de detección de intrusos adaptativo usando redes Bayesianas, donde el conjunto de datos de aprendizaje se actualiza al agregarle nuevos patrones de anomalías o

intrusiones.

Redes Neuronales. Son entrenadas para aprender el comportamiento normal de los procesos a monitorear y poder detectar potenciales intrusiones o anomalías contra el sistema [Ghosh98].

HMMs (Hidden Markov Models). Este esquema es relativamente nuevo y consiste en modelar la dinámica del sistema y generar un modelo de comportamiento normal, que será utilizado para series de observaciones subsecuentes. Con ayuda del modelo del comportamiento normal se determina la probabilidad que la secuencia tenga un comportamiento normal o anómalo.

Recientemente, la seguridad de las redes se ha convertido en un tema extremadamente vital que necesita del desarrollo de soluciones correctas y eficientes, capaces de defender efectivamente los sistemas de red y la invaluable información que transita en ellos. Esto se debe a que los sistemas de redes juegan un gran rol crítico en la sociedad mo-derna y a que existen datos sensitivos que se encuentran almacenados y que se manipulan a través de las redes. En estas redes existe la posibilidad de intentos de acceso no-autorizados por parte de intrusos, que pueden provocar serios daños a personas, corporaciones y a la sociedad en general. Esto ha provocado que la seguridad en las redes sea un tema de vital importancia en la época actual y futura [Shyu07].

La tecnología en los sistemas de cómputo avanzan a grandes pasos, un dispositivo móvil actual tiene el mismo poder de cómputo, o más, de lo que tenía una computadora de escritorio de hace algunos años. Los sistemas de malware y las técnicas empleadas por los atacantes, han avanzado considerablemente y se han vuelto más complejas y difíciles de detectar. La importancia de intercambiar y transferir información por las redes se ha vuelto una necesidad creciente, y es por esto que surge la necesidad de crear nuevos esquemas de detección de anomalías, herramientas de detección de ataques, detección de intrusiones, etc. Pero estas herramientas y esquemas no avanzan a la misma velocidad que lo hacen las tecnologías de cómputo y los sistemas y esquemas de ataque a los sistemas de información.

En el presente trabajo de tesis se pretende detectar comportamientos anómalos, en el consumo del ancho de banda de la subred del Posgrado en Ingeniería Eléctrica. Para esto se tomarán como métricas los valores del comportamiento del consumo de la red. Las variables a analizar en el consumo del ancho de banda de la red son: los valores del tráfico de red de entrada (consumo de ancho de banda de red utilizado para descargar información desde Internet) y el tráfico de red de salida (consumo de ancho de banda de red utilizado

para peticiones o subir información a Internet), así como el día de consumo del ancho de banda, como caso especial y para fines de prueba.

El fin es detectar comportamiento anómalo en el consumo del ancho de banda de red, auxiliado por un modelo de HMM creado y optimizado por la programación evolutiva, que modela la dinámica del sistema tomando como métricas las variables del consumo de ancho de banda de la red.

1.5. Planteamiento del problema

Los HMMs tratan con los problemas de reconocimiento de secuencias de observación, secuencias probables de estados y los parámetros de diseño óptimos de un HMM. Para poder proporcionar respuesta a estos problemas se necesita conocer a priori la topología del modelo. Es decir, que para construir un HMM se necesita decidir cuántos estados tendrá, qué transiciones existirán entre sus estados y qué función de distribución se usará en los estados involucrados.

El principal problema que esta tesis trata es el de encontrar el mejor diseño o arquitectura de un HMM para realizar la tarea de detección de anomalías. El diseño se efectuará de manera automática, con sólo la secuencia de observación generando todos los parámetros necesarios y óptimos en la construcción del modelo.

Para probar la eficacia y eficiencia de los modelos generados, se utilizan los datos generados del comportamiento del consumo de ancho de banda de red, del posgrado en Ingeniería Eléctrica, en la creación y optimización de los parámetros de un HMM. El HMM modelará el perfil del comportamiento normal del tráfico de red, basado en las métricas del consumo de ancho de banda de salida y entrada de la subred de la universidad.

Para entender lo que se entiende por comportamiento "normal", se puede observar un conjunto de variables del sistema y se puede decir que el sistema exhibe comportamiento normal o no normal, conocido como anómalo. A partir de las observaciones del comportamiento del sistema bajo condiciones normales (considerando que no se encuentra bajo ningún tipo de ataque), se captura la dinámica del sistema en un modelo probabilista. Este modelo probabilista nos indicará cuando el comportamiento del sistema se desvía considerablemente de su comportamiento normal. En estas condiciones decimos que el sistema presenta un comportamiento anómalo.

Una vez encontrado el mejor HMM que modele la secuencia de observación, es

factible encontrar la solución al problema del reconocimiento de una secuencia de observación. El problema a resolver consiste en: dada una secuencia de observaciones O, de una (o varias) variable(s), diseñar de manera automática un HMM λ que maximice la probabilidad de que la secuencia observada haya sido generada por λ. Esto es

$$P(O \mid \lambda)$$

con la finalidad de utilizarlo en la solución del problema de detección de anomalías en sistemas de cómputo.

Con el modelo generado, se efectúan pruebas de detección de comportamiento anómalo en la red. Las pruebas con datos sintéticos representan la detección de anomalías, las cuales pueden ser generadas por un ataque de denegación de servicio, un virus o gusano propagándose por la red, el uso del ancho de banda de la red para bajar música, videos, ataques de barrido de la red, etc.

1.6. Justificación

Uno de los grandes riesgos en Internet es el crecimiento en el número de ataques distribuidos y automatizados que son llevados a cabo por intrusos maliciosos internos y externos.

Gracias a los avances y desarrollo de computadoras más poderosas y con grandes recursos de cómputo, el número de ataques a los sistemas computacionales se ha incrementado y se están tornando cada vez más sofisticados. En la época contemporánea se ha extendido el uso de herramientas de scripts y exploits (vulnerabilidades del sistema) para intentar intrusiones.

Se ha visto que las amenazas de ataques sofisticados a los sistemas está creciendo y desafortunadamente, los sistemas de detección no crecen al mismo nivel que las amenazas actuales.

Los sistemas de detección y prevención de intrusos son efectivos sólo contra patrones bien conocidos de ataques. Nuevos patrones, o incluso hasta la menor modificación de un patrón conocido, no podría ser detectado por alguno de estos sistemas. Es por esto que las Base de Datos de firmas dañinas debe ser actualizada constantemente para reflejar los nuevos modos de ataques [Green07].

El porcentaje de éxito en la intrusión de un atacante tiene relación directa con el intervalo de tiempo que se detecta el ataque y el tiempo que se toma en realizar una medida de respuesta a la intrusión. Si una contramedida se realiza tan pronto como la intrusión o anomalía se detecta, el porcentaje de éxito del ataque descenderá drásticamente [Lang04].

Un punto importante es que los nodos capturados o comprometidos (dispositivos de cómputo utilizados sin autorización por un atacante) pueden ser usados para provocar ataques más dañinos a la red o a otras redes. Es decir, desde los hosts comprometidos, se puede amenazar a otros hosts o a otras redes. Otra amenaza es suplantar a los hosts originales, escuchando los paquetes que se transmiten en la red, y en resumen terminar con la confiabilidad del sistema.

Por último, no hay que olvidar que toda la información digital que se origina desde el hogar, centros de investigación y hasta la información gubernamental de un país, pasa por los sistemas de redes.

Dado esta problemática, esta tesis se enfoca al desarrollo de un esquema de detección basado en anomalías. El esquema propuesto se basará en lo siguiente:

- Un conjunto de series de tiempo (secuencia de observaciones) que representa la actividad normal de un sistema, este conjunto de datos, constituirán los valores de variables del sistema a monitorear, a intervalos de tiempo dados. Los valores de las mediciones serán valores continuos.

- Se incluirá un sistema inteligente que nos ayude en la creación y el desarrollo del Modelo Matemático Probabilístico a utilizar para entrenarlo con las series de tiempo. Con las series de tiempo dadas, el modelo creado aprenderá el comportamiento considerado como normal del sistema.

- Se desarrollarán y emplearán los Modelos Ocultos de Markov como modelos probabilísticos en la detección de las anomalías. Este modelo una vez entrenado con las series de datos de las variables aleatorias, será capaz de detectar comportamientos normales y anormales del sistema.

1.7. Objetivos

En esta tesis se presenta el uso de EP (Programación Evolutiva) en el diseño y optimización de los parámetros de un HMM de manera automática, sin conocimiento a

priori del diseño, sólo se necesitarán los datos de una serie de tiempo. Una vez obtenido el HMM con el proceso de optimización de la EP, se utilizará en la detección de anomalías en el tráfico de red. Por lo que se pretende:

- Desarrollar un esquema de seguridad que nos permita detectar anomalías del sistema en cuestión, basado en un Modelo de Análisis de Series de Tiempo.
- Utilizar vectores de variables aleatorias de series de tiempo, en el modelado, construcción y entrenamiento de nuestros HMMs.
- Construir HMMs que modelen el comportamiento normal del sistema, entrenados con las series de tiempo dadas, para poder detectar desviaciones o anomalías en las observaciones posteriores.
- Modelar, crear, entrenar y utilizar HMMs multivariables en la detección de anomalías de series de tiempo.
- Mostrar evidencia que sugiera que este enfoque, del uso de los HMMs, puede ser satisfactorio para detectar anomalías, ataques y abuso de los sistemas de cómputo.

1.8. Tesis

Dada una serie de tiempo de valores continuos, y tomando como base la programación evolutiva, es posible desarrollar HMMs que modelen la dinámica del proceso observado y mediante los cuales se pueda lograr una eficaz y eficiente detección de anomalías en un sistema de cómputo.

1.9. Contenido de la Tesis

El Capítulo 2 presenta el estado del arte de los HMMs usados en la detección de anomalías, así como el uso de GAs en la optimización de parámetros de los HMMs. En el Capítulo 3 se muestran los conceptos preliminares de los HMMs, clasificación y tipos, así como su uso en la detección de anomalías. En el Capítulo 4 se describe el ambiente de trabajo donde se evolucionaron los HMMs, empleados en la detección de anomalías. En el Capítulo 5 se presentan los resultados obtenidos al aplicar los diferentes modelos de HMMs a secuencias de observación, de una y dos variables aleatorias continuas, para la detección

de comportamiento anómalo. Por último, las conclusiones de los resultados observados en el uso de los HMMs en la detección de anomalías en series de tiempo, se muestran en el Capítulo 6, así como el planteamiento de nuevos desarrollos futuros o ampliaciones al presente trabajo de tesis.

Capítulo 2

Estado del Arte

En este capítulo se dará una descripción del estado del arte de los trabajos basados en HMMs (Hidden Markov Models) y su uso aplicado a la detección de anomalías, así como la descripción de los métodos basados en GAs para la optimización de los parámetros de un HMM. Se describirán los esquemas principales utilizados en la detección de intrusos y, por último, se describirá de manera breve los problemas involucrados en el tema de Seguridad Informática.

2.1. HMMs utilizados en la detección de anomalías.

En un inicio, la mayoría de los trabajos sobre HMMs (Hidden Markov Models, Modelos Ocultos de Markov), estaban orientados a secuencias de ADN, reconocimiento de voz, etc. En la actualidad, el área de trabajo de los HMMs se ha ampliado a un gran campo de acción.

La utilización de los HMMs aplicados a la detección de anomalías es relativamente nuevo, pero existen trabajos relacionados a esta área, como se verá en los párrafos siguientes. Los trabajos se presentan en orden cronológico descendente, para mostrar las investigaciones más recientes con respecto al campo de acción que estoy abarcando al momento. Todos los trabajos que menciono basan su diseño en los HMMs, con la característica que para su correcta funcionalidad, son creados y entrenados por personas con experiencia en el área, es decir, con conocimiento a priori.

La mayoría de las investigaciones están orientados a observaciones discretas, y algunas de las estructuras de los HMMs comprenden de dos a cuatro estados únicamente.

El entrenamiento de los HMMs se basa en las llamadas al sistema y en el comportamiento o perfil de usuario. A continuación se describen algunos de los trabajos de investigación sobre los HMMs, y su aplicación a la seguridad de sistemas informáticos.

En el documento de Xiuqing et al. [Xiuqing10] se propone un sistema de detección de intrusos (IDS), basado en HMMs y en un algoritmo de agrupamiento de adaptación rápida (FACA, Fast Adaptive Clustering Algorithm). En el documento redactan cuatro métodos de detección basados en: HMM, FACA y luego se combinan estos dos detectores con los modelos basados en: detecccón de anomalías y deteccíon de mal uso. Cabe mencionar que cada método de detección está asociado a un HMM en particular. En esta investigación la aplicación de los HMM inicia con la fase de entrenamiento, donde los datos del clustering son transformadas a secuencias de observación de donde se diseñan los HMMs. En el siguiente paso, la fase de prueba, los datos de clustering son transformados a secuencias cortas, las cuales se pasan a los HMMs para determinar la probabilidad de la secuencia y poder seleccionar, de los cuatro, el mejor modelo para utilizarlo en la detección de la anomalía. Del documento se puede destacar que cada esquema propuesto está asociado a un HMM, y la arquitectura de estos se infiere en base a las secuencias de observación. Esto implícitamente significa que el diseño del HMM está hecho en base a conocimiento a priori sobre la arquitectura del HMM. Además los esquemas contemplan sólo valores discretos y una sola variable.

En la investigación de Dan et al. [Dan10], se sugiere un nuevo método de detección de intrusos basado en un HMM, al calcular la probabilidad de salida de secuencia corta de las llamadas al sistema. La investigación se basa en un HMM de dos estados, donde el estado 0 representa el comportamiento normal del sistema, y el estado 1, representa un comportamiento anómalo. El conjunto de símbolos que maneja el HMM es discreto, y se basa en un conjunto de comportamientos del sistema observado, es decir, la secuencia de llamadas a determinadas funciones del kernel. El sistema requiere que la longitud de las secuencias sean cortas y del mismo tamaño, ya que el sistema calcula la probabilidad media de la ocurrencia de una secuencia de observación, con una longitud dada. Esta probabilidad media se puede usar para detectar si secuencias posteriores, del mismo tamaño, son normales o no. Si las características estadísticas o la secuencia de llamadas enviada por un proceso es diferente, se considerará una anomalía.

Zihui et al. [Zihui10] proponen un método para la detección de instrusos basado en un HMM y la teoría de conjuntos imprecisos (rough sets). El proceso inicia con la

extracción de un conjunto pequeño de datos de las secuencias normales de las llamadas al sistema, estos datos son transformados en una tabla de decisión. Ésta se reduce y se extraen las reglas más simples que presentan un modo de comportamiento normal, por medio de las teorías de conjuntos imprecisos. Las reglas se utilizan para detectar anomalías. Para realizar detecciones rápidas de intrusiones conocidas, se utiliza un motor de detección basado en un HMM. Este método propuesto se basa, particularmente, en llamadas al sistema, el HMM está diseñado con 30 estados, y la longitud de la secuencia de observación es de 182, la que equivale a las llamadas del sistema de SUN-OS.

Ye et al. [Ye10] investigan la utilización de un HMM en la detección de anomalías en una MANET (Mobil Ad hoc NETwork). Estas redes presentan características especiales como: sin estructura fija, comunicación inalámbrica, topología dinámica, etc., estas mismas características hacen a estas redes muy vulnerables a un ataque malicioso. En el documento de Ye et al. proponen un método de detección de anomalías basado en un HMM. Es un sistema diseñado con conocimiento a priori, que consta de dos estados: donde el estado 0, representa un comportamiento normal y el estado 1, representa un comportamiento anormal. Las secuencias de observación son monitoreadas por un nodo de la red, que registra el comportamiento de ruteo de los mensajes entre un grupo de nodos (cluster) en tiempos definidos. Cualquier cambio de tiempo en la entrega de mensajes entre los nodos del cluster, se considerará una anomalía.

Wang et al. [Wang10] presentan un sistema de detección de anomalías basado en llamadas al sistema. En este trabajo, diseñan el número de estados del HMM, realizando experimentos entre 10 y 40 estados, de manera manual. De esta forma establecen el HMM en 15 estados, el cual entrenan con la secuencia de llamadas al sistema. El proceso de detección de anomalías lo efectúan comparando la secuencia de estados observados, contra la que se tiene almacenada, la cual se obtuvo en el momento del entrenamiento.

En la investigación de Zhicai et al. [Zhicai10] se propone detectar ataques complejos a las redes de computadoras. Definiendo como ataque complejo (DDOS, phishing, etc.) aquel tipo de ataque que finaliza con una serie de pasos discretos, bajo el control del atacante o de un programa malicioso. Además, cada paso del ataque puede completarse con diferentes métodos y pueden ocurrir en diferentes tiempos. En esta investigación se utiliza un HMM por cada tipo de ataque complejo, y los estados, que constituyen al HMM, representan los pasos discretos del ataque. El sistema se apoya en el uso de monitores de red, que clasifican y almacenan los eventos anómalos. En el proceso de entrenamiento de los HMMs, toman

las secuencias de los eventos observados y clasificados de la base de datos. En la fase de prueba, cada secuencia de alertas se le proporciona a los modelos, y aquel modelo que genere la mayor probabilidad con la secuencia dada, es el tipo de ataque que se pronostica en la red. Los tipos de eventos de advertencia en el sistema, de los diferentes monitores de la red, son correlacionados y su relación se analiza para detectar los tipos de ataques complejos y predecir su grado de amenaza al sistema. Los tipos de ataques se clasifican en un rango de 1 a 7, de acuerdo al grado de peligrosidad, generado por el sistema BLADE IDS Informer. Cada uno de estos 7 tipos de ataque corresponde a un HMM, y con las secuencias de advertencias, que corresponde a un tipo de ataque, se entrenan a los HMMs. Los HMMs tienen 7 estados, tienen un conjunto de 11 símbolos o valores discretos a observar, y en general, están diseñados en base a la experiencia.

En el trabajo de Zaki et al. [Zaki10] presentan un HMM que combina dos tecnologías para el modelado de patrones complejos en datos secuenciales: Minería de Patrón y Modelado de Datos. El sistema se puede aplicar en la clasificación de secuencia de cadenas de proteínas, registro de uso de la web, detección de intrusiones y en la corrección de ortografía. El sistema primero extrae las patrones de secuencia frecuente vía minería de secuencia, y con estos datos se construye el HMM, con un número de estados entre dos y cuatro. El HMM se utiliza para descubrir e interpretar rangos temporales grandes y cortos, y dependencias en los datos analizados. En el aspecto de detección de intrusos, los datos se basaron en prácticamente los perfiles de los usuarios, utilizando comandos de Unix.

Sugaya et al. [Sugaya09] presentan un sistema de detección de anomalías ligero, que se encarga de monitorear los recursos del sistema computacional para la detección de anomalías, especialmente ataques. Los ataques son síntomas de abuso del recurso, y para esta detección se utilizan los patrones de uso del recurso de cada proceso. El sistema se basa en métodos de aprendizaje de máquina. Utiliza el método de clustering para cuantificar el uso del recurso y poder aprender los patrones normales con un HMM. En la fase de ejecución, el sistema encuentra las anomalías comparando el uso del recurso por la aplicación contra el modelo aprendido. Reduciendo el tiempo de análisis y haciendo posible el monitorear el proceso en tiempo real. El prototipo es capaz de detectar anomalías tales como Inyección de SQL y el desbordamiento de buffer con un mínimo de falsos positivos.

Zeng et al. [Zeng09] mencionan que los HMMs se han aplicado con éxito en la detección de intrusiones y han probado ser una buena herramienta para modelar el comportamiento normal de procesos privilegiados. Sin embargo, un gran problema con este

enfoque es que demandan un excesivo uso de recursos de cómputo para modelos que requieren gran tiempo de entrenamiento. En este artículo se presenta un nuevo método de reducción del conjunto bruto de las observaciones en el HMM para superar este defecto. El enfoque propuesto clasifica y simplifica las grandes secuencias de observación en virtud de la reducción del conjunto bruto, y las condiciones de decisión se podrían utilizar en la detección de anomalías.

Li et al. [Li09] describen el excelente funcionamiento de los HMMs y su gran uso en el reconocimiento de patrones. Pero debido al alto porcentaje de falsas alarmas en los clásicos sistemas de detección de intrusos basados en HMMs, se propone un enfoque borroso para el HMM, llamado HMM Borroso. Donde se introduce la lógica Borrosa al HMM, originando que el porcentaje de robustez y precisión de los sistemas de detección basados en HMM Borrosos, sean mejores en gran manera. El método es eficiente para clasificar entre perfil anómalo y perfil normal, teniendo bajo porcentaje de falsos positivos, y con un gran porcentaje de aciertos.

Existe un trabajo de Zhao et al. [Zhao09], que es un protocolo de detección de anomalías que utiliza un HMM, el cual verifica e identifica entre tráfico normal y anormal. Utilizan un conjunto de datos del Laboratorio Lincoln del Instituto Tecnológico de Massachusetts, para probar el correcto funcionamiento del modelo HMM, en la evaluación de detección.

En el trabajo de Shivaraj et al. [Shivaraj08] se propone un enfoque basado en estadística para detectar puntos de acceso no autorizados usando un HMM, el cual se aplica a los encabezados de los datos medidos pasivamente, recolectados de un ruteador. El enfoque se basa en las características del tráfico asociado individualmente a los hosts finales. El HMM, con tres estados, se entrena con un conjunto de datos obtenidos de actividades normales de Internet y ataques de negación de servicio. Una vez entrenado el HMM, se encarga de monitorear el tráfico de paquetes que llegan a intervalos de tiempo por flujos diferentes al límite de la red. Donde una variación en el flujo de paquetes origina un cambio de estado del HMM.

Srivastava et al. [Srivastava08] utilizan HMMs para modelar la secuencia de operaciones en el proceso de la transacción de una tarjeta de crédito, y se muestra cómo puede ser usado para la detección de fraudes. El HMM se entrena inicialmente con el comportamiento normal de un cuentahabiente. Si una una nueva transacción de una tarjeta de crédito no es aceptada por el HMM entrenado, con la suficiente probabilidad, se considera un fraude. Al

mismo tiempo, se trata de asegurar que las transacciones genuinas no sean rechazadas.

En la investigación de Singh et al. [Singh07] se ilustran las capacidades de los HMMs, combinados con el seguimiento asistido por características para la detección de amenazas asimétricas (lo cual se refiere a las tácticas empleadas por sujetos para efectuar ataques contra un oponente superior). Este enfoque combina el modelo probabilístico basado en transacción con los HMMs y el monitoreo apoyado en características. El HMM describe la dinámica de una red terrorista, se incluye información de la gente involucrada, la localización geográfica, etc. Los estados del HMM contienen directamente estas características, que pueden ser usadas para distinguir los objetivos de interés.

En Yasami et al. [Yasami07] se trabaja con la detección de anomalías de la red. Donde se utiliza el proceso de reconocimiento del comportamiento del tráfico de un host, el cual se monitorea y su comportamiento se compara contra un modelo (normal) dado. La detección de anomalías de red es un área de investigación activa y dinámica. Existen varios métodos utilizados en el reconocimiento del comportamiento, pero la incorporación de HMMs para la detección de anomalías (especialmente en la detección de anomalías ARP, Protocolo de Resolución de Dirección) es un método nuevo. Este documento se concentra en la clasificación del tráfico ARP en la red como anormal y normal usando un HMM especial. El principal objetivo de este trabajo es construir un sistema de detección de anomalías estadístico, un modelo predictivo capaz de descriminar entre comportamiento normal y anormal del tráfico ARP de la red. El método propuesto es único en este aspecto donde se utiliza un HMM modificado, además el algoritmo detecta anomalías ARP basadas en host.

En Lee et al. [hyeonLee07] se propone un diseño mejorado de un sistema de detección de intrusos basado en agente móvil, usando un HMM para la detección. El HMM se usa para detectar un patrón de comportamiento anormal al analizar la información de la bitácora del sistema. Al usar agentes distribuidos, se puede reducir el retardo y la carga de la red, y cada agente puede operar independientemente. Adoptando este sistema, se puede mejorar el funcionamiento de detección.

En la investigación de García et al. [Garcia06] presentan un enfoque de detección de anomalías basado en la construcción, modelo creado con conocimiento a priori, de un HMM, entrenado en la carga de trabajo de un procesador. El HMM se utilizó para observar las secuencias de las medidas de carga de trabajo del procesador, donde si la probabilidad de la observación de ser generada por el modelo era más baja de lo que había sido estimada con el HMM, se consideraba una anomalía. En este trabajo los valores son discretizados y

se maneja una sola variable.

En el documento de Khanna et al. [Khanna06], presentan una estrategia para la detección de intrusiones usando un modelo con mezcla de funciones multigaussianas en las observaciones, las cuales son usadas después en la predicción de un ataque que existe en la forma de un estado oculto. El trabajo se enfoca en la detección de anomalías usando usuarios y perfiles de tendencia. Donde cualquier comportamiento fuera de este perfil se considera como una actividad inusual, la cual se registra como una observación en la secuencia. Una vez que se tienen estas observaciones, se utiliza un HMM para predecir una posible intrusión, basado en sus transiciones de estados ocultos determina la secuencia de estado más probable en la intrusión. Estos autores están trabajando en arquitectura ad hoc, por lo que utilizan sensor de datos, los cuales coleccionan y analizan datos. Este enfoque basado en HMM (con el modelo diseñado con conocimiento previo) correlaciona las observaciones del sistema (uso y perfil de actividad) y las transiciones de estados para predecir la secuencia de estados más probable de intrusión.

En el trabajo de Jha et al. [Jha01] presentan un algoritmo de detección de anomalías estadística basado en Cadenas de Markov. El algoritmo propuesto puede ser aplicado directamente en la detección de intrusiones para descubrir actividades anómalas. Este algoritmo usa la secuencia de las llamadas del sistema que utilizan los procesos así como el seguimiento de sus actividades. El algoritmo utiliza un conjunto normal de registro de las llamadas al sistema y se construye un modelo estadístico, después se usa éste para construir un clasificador capaz de reconocer un comportamiento normal de uno anormal.

En la investigación de Warrender et al. [Warrender99] se estudian los datos generados por las secuencias de las llamadas al sistema dentro del kernel del sistema operativo. Comparan la habilidad de diversos métodos: Secuencias Enumeradas, Frecuencias, Minería de Datos y Máquinas de Estado Finito, tanto en el modelado de los datos y representación del comportamiento normal del sistema, así como para reconocer intrusiones. El trabajo mostró que el comportamiento normal de un programa puede ser caracterizado por patrones, y desviaciones de estos patrones pueden ser identificados como violaciones de seguridad de un proceso. En esta investigación se determinó el número entre 20 y 60 estados en los HMMs, de acuerdo al número de llamadas al sistema hechas por los programas.

En resumen podemos decir que este trabajo de tesis no tiene por el momento comparación, ya que todas las investigaciones mencionadas construyen sus modelos de HMM basados en la experiencia y conocimiento del diseñador. Todos utilizan valores discretos y

casi todos toman como referencia una sola variable en la observación.

2.2. Optimización de parámetros de un HMM utilizando GAs

Un método utilizado para lograr encontrar los mejores parámetros de un HMM, son los GAs (Genetic Algorithms, Algoritmos Genéticos), y como se verá, el encontrar estos parámetros óptimos de los HMMs, es todo un reto. El adecuado inicio de valores de los parámetros de un HMM, repercute considerablemente en el desempeño del modelo.

Determinar los mejores parámetros de un HMM es un problema de optimización difícil, debido a que los HMMs describen un doble proceso estocástico [Warrender99].

Todos estos sistemas, al igual que este trabajo de tesis, también generan los HMMs de manera automática. La diferencia consiste en que todos los trabajos mencionados utilizan HMMs basados en secuencias de observación de valores discretos y orientada a símboles o alfabetos pequeños (secuencias de ADN o alfabetos de algún idioma), sólo observan un valor en la secuencia de observación y están construidos utilizando GAs. La diferencia con el trabajo de esta tesis es que utiliza EP en la evolución del HMM, el HMM generado es capaz de observar secuencias de vector de datos, observa datos con valores de tipo real y el modelo se aplica a la detección de anomalías.

Se debe de tomar en cuenta que el desempeño de un HMM dependerá en gran medida de sus parámetros iniciales, sobre todo al momento de ser entrenado con el algoritmo de Baum-WElch. Existen trabajos de investigación en esta área, donde se aplican los GAs como un método de búsqueda para encontrar los mejores parámetros para un HMM. A continuación se mencionan algunos trabajos relacionados con la búsqueda de tales parámetros.

Oudelha et al. [Oudelha10] combinan el algoritmo de Baum-Welch (BW) y los GAs, para la obtención de los mejores parámetros de un HMM, que se usará en el proceso del reconocimiento automático del habla. Este trabajo se auxilia de la herramienta y ambiente de trabajo de MatLab GA ToolBox, para la optimización de los parámetros del HMM. En el trabajo comparan el refinamiento de los parámetros usando el algoritmo iterativo de BW contra el de GA-BW, dando mejores resultados el método de optimización de parámetros del algoritmo híbrido de GA-BW.

Cheshomi et al. [Cheshomi10] presentan un algoritmo híbrido de Optimización de Caos (OC) y Baum-Welch para el refinamiento de los parámetro de un HMM, utilizado

para el reconocimiento del habla. El OC, es un método de búsqueda estocástico, y a menudo existe en los sistemas no lineales. Con el algoritmo de OC se pretende optimizar los valores iniciales del algoritmo de BW, así como evitar que BW caiga en un óptimo local. Además, OC, es rápido y requiere de menos almacenamiento que otros algoritmos de optimización híbridos. Se probó con el lenguaje persa, se cambió la función de entrenamiento en el HTK Toolkit, por el algoritmo híbrido. El proceso de optimización determina los parámetros con OC, y posteriormente se pasa el HMM a un refinamiento con el algoritmo iterativo de BW. El HMM creado y optimizado fue un HMM con transiciones de izquierda-derecha, con 5 estados.

En Bhuriyakorn et al. [Bhuriyakorn08] se muestran algunos enfoques que son utilizados exhaustivamente para la generación topológica de HMMs. Un algoritmo, por ejemplo, predefine rígidamente la conectividad entre los estados del HMM, por lo que este algoritmo sólo estima el número de estados y mezclas gaussianas. Otro algoritmo, llamado División de Estados Sucesivos, comienza permitiendo una sola transición en el estado, y crece de un estado a la topología óptima. Esto es, que empieza con un estado hasta crear una topología de un HMM completa. Existe otro algoritmo que desarrollaron, Reducción de Estados, el cual comienza con una topología de un HMM con sus estados completamente conectados, con un número predefinido de estados, y después, de manera iterativa reduce las transiciones entre los estados hasta que el proceso termina. También describen otro algoritmo que utiliza diferentes caminos, entre los estados, de un HMM. La idea con este algoritmo es el de mejorar las topologías en vez de usar esfuerzos significativos en ajustar un solo camino. La investigación se centra básicamente en el problema de la selección de topologías de un HMM. Cabe mencionar que estos algoritmos, utilizados para determinar los parámetros óptimos de un HMM, se centran en la actividad del reconocimiento de los fonemas en tailandés.

En Yang et al. [Yang08a] se aborda el problema de la optimización de los parámetros de un HMM. Donde primero, se usa un GA para obtener un espacio de soluciones corto, donde la Búsqueda Tabú (TS, Tabu Search) se aplica como una memoria de corto plazo, almacenando en una lista tabú los operadores genéticos usados, y la descendencia generada. Asi que el algoritmo GATS (GA + TS), garantiza que el GA no caerá en una convergencia prematura, debido a que los operadores genéticos usados y la descendencia generada se registra en una lista tabú, la cual puede ser usada para impedir que se utilicen los mismos elementos en un futuro cercano. Para mejorar la velocidad de convergencia en la

optimización de un HMM basado en GATS, se le combina con el algoritmo de Baum-Welch (BW). Generando el algoritmo híbrido GATSBW, el cual combina estos tres algoritmos mencionados, con el propósito de generar y entrenar un HMM en el reconocimiento del habla continua. El algortimo GATSBW supera la deficiencia de la lenta velocidad de convergencia del algortimo GATS y ayuda al algoritmo de BW a escapar de un óptimo local.

En el documento de Yang et al. [Yang08b], se consideran a los HMMs como la tecnología dominante en el reconocimiento del habla. El problema de la optimización de los parámetros del modelo es de gran interés para los investigadores de esta área. El algoritmo de BW (Baum-Welch) es el método popular de estimación debido a su confiabilidad y eficiencia. Sin embargo, es fácil que quede atrapado en un óptimo local. Recientemente, los GAs y la Optimización de Partículas (PSO, Particle Swarm Optimization) son consideradas, entre otras, técnicas modernas de optimización heurística interesantes. Se sabe que los dos enfoques se centran en encontrar una solución dada una función objetivo, pero emplean estrategias y esfuerzo computacional diferentes. La investigación se centra en la aplicación y desempeño de los PSO y los GAs para la optimización de un HMM continuo en el reconocimiento del habla. Los resultados experimentales demuestran que los PSO son superiores a los GAs respecto al desempeño de reconocimiento.

En el trabajo de Bhowmik et al. [Bhowmik07], se presenta un sistema de reconocimiento de la escritura a mano aislada de palabras en Bangladesh, con un léxico fijo, usando los HMMs. Se utiliza el método estocástico de los GAs para entrenar los HMMs. El HMM utilizado es de los clasificados como de izquierda a derecha. Para la extracción de las características, el contorno de la imagen se maneja en dirección de las manecillas del reloj y en sentido contrario a éstas, y los cambios significativos son usados por el modelo propuesto.

En Ogawa et al. [Ogawa06] utilizan un GA basado en el método de optimización de estructura para un HMM Parcial (PHMM, Partly HMM), en el que la dependencia de los estados y de las observaciones del PHMM se selecciona de manera óptima de acuerdo a cada modelo, usando el criterio descriminativo de maximización del cociente de probabilidad ponderada. El trabajo se encuentra enfocado hacia el modelado acústico para el reconocimiento del habla.

En el trabajo de Won et al. [Won05] se presenta el uso de GAs para evolucionar HMMs en la predicción de información de estructuras secundarias para la secuencias de proteínas. Este GA híbrido fue ejecutado en un cluster de computadoras como un algoritmo

genético paralelo. Este artículo también menciona que la topología del HMM fue restringida principalmente a la parte biológica.

La mayoría de los trabajos en GAs se enfocan en determinar los mejores parámetros del HMM, otros se concentran en, una vez obtenidos los mejores parámetros, utilizar el HMM en la investigación de secuencias de proteínas o en fonética. La mayoría de los HMMs usados en la detección de anomalías, descritos previamente, fueron diseñados con conocimiento a priori de los modelos a utilizar.

El diseño de los HMMs con conocimiento a priori, implica implícitamente conocimiento derivado de experiencia probabilística previa de los modelos a crear. Es decir, se requiere de conocimiento previo del comportamiento de los datos, y la consideración de la teoría general de los HMMs. Este trabajo propuesto combina dos técnicas: la programación evolutiva para la optimización de los parámetros de un HMM, y el uso del HMM en la detección de anomalías. Donde una vez que se obtiene el HMM, evolucionado con los GAs, lo aplicamos en la Detección de anomalías. Este enfoque permite que cualquier usuario utilice el sistema, sin la asunción de conocimiento previo en probabilidad y teoría de los HMMs.

2.3. Esquemas de detección de intrusos

El propósito de un Sistema de Detección de Intrusos (IDS, Intrusion Detection System) es el de proveer de un marco estructurado que permita descubrir, reportar y posiblemente reparar intrusiones pasadas y recientes en tiempo real [Foo04].

Se debe mencionar que el presente trabajo de investigación no se considera un IDS. Se puede ver, si cabe, como la parte o módulo correspondiente a la creación del sistema detector de anomalías únicamente.

Gran parte de los actuales IDS y de los Sistemas de Respuesta a Intrusos (IRS, Intrusion Response System), se basan en el reconocimiento o descubrimiento de patrones o estados que detectan un comportamiento intrusivo o anómalo. Los IRS se pueden catalogar como sistemas de sólo notificación, sistema de respuesta manual o sistema de respuesta automática. De esta clasificación un gran número de IDS e IRS sólo hacen notificaciones. Son sistemas que solamente hacen reportes y alarmas [Ragsdale00].

Estos esquemas o técnicas se implementan tanto en redes cableadas, como en redes inalámbricas. Y se desarrollan tanto en plataformas fijas como en dispositivos móviles, esto debido a que los intentos de intrusiones se dan en todos los ambientes. Por eso es que se

realiza investigación en todas estas áreas, como se verá a continuación.

En el trabajo de Ido Green et al. [Green07] se mencionan dos sistemas importantes en la detección y prevención de ataques a los sistemas de cómputo, particularmente a las redes de computadoras: los IDS, que se encargan de monitorear la actividad de la red en busca de anomalías, y los Sistemas de Prevención de Intrusiones (IPS, Intrusion Prevention System), los cuales tienen como actividad principal el bloquear los ataques en tiempo real. Ambos sistemas trabajan en línea y se basan en el uso de patrones de actividad de la red para la detección de firmas de modos conocidoss de ataques. Estos sistemas mencionados, IDS e IPS, tienen una tendencia hacia los falsos positivos, se estima que es entre 30 y 60 % las alertas que son alarmas de falsos positivos. De este modo, las alertas válidas se pierden en la gran cantidad de información.

En Green at al. [Green07] también se cita un reciente y nuevo enfoque a la seguridad computacional, la Prevención Activa de Intrusiones (AIP, Active Intrusion Prevention), que consiste en examinar constantemente la actividad en la red, buscando solicitudes de datos que pueden ser usadas para entrar a la red o para otros propósitos hostiles similares. En estas situaciones, el sistema proporcionará los datos solicitados, con la característica que serán registrados. Y cuando el atacante trate de usar los datos marcados, el sistema conocerá con certeza que es un ataque válido, y bloqueará al atacante de manera indirecta o directa. El sistema virtualmente se encuentra libre de falsos positivos.

En el trabajo de Chunyue et al. [Chunyue06], se propone un IDS basado en la detección de patrones. Este sistema consiste de los siguientes módulos: Colección, Análisis, Respuesta, y Ataque. Donde el primero se encarga de la recolección de paquetes, el siguiente es responsable del preprocesamiento y detección. El tercero es responsable de registrar el ataque, almacenar los datos capturado, enviar advertencias, reconfigurar el ruteador, etc., y el último módulo se auxilia de Snort, ya que este sistema cuenta con la base de datos de reglas o patrones de ataque. Es un sistema basado en el reconocimiento de firmas o patrones conocidos dados por Snort.

El trabajo de Kannadiga et al. [Kannadiga05] trata de la detección de intrusiones en ambientes de cómputo móvil, consistente de nodos estáticos y móviles heterogéneos, donde se hace uso de agentes de software móviles, que son enviados desde un nodo estático hacia los nodos inalámbricos con comportamiento sospechoso, y que una vez instalados, colaboran en la detección de intrusiones. Estos agentes móviles se encargan de reunir datos y analizarlos.

En la investigación de Li et al. [Li04] se presenta una herramienta poderosa para el modelado estadístico y procesamiento de coeficientes de wavelet. Captura la dependencia de los coeficientes wavelet y los coeficientes de la escala de una variable de proceso medible, respectivamente. El documento presenta la manera de seleccionar los coeficientes de wavelet y construir un HMM con los coeficientes wavelet seleccionados y los coeficientes de la escala. El método que se presenta usa detección en línea de tendencia de proceso, donde todos los coeficientes de la escala y varios coeficientes seleccionados wavelet son tomados en cuenta.

2.4. Seguridad informática

Desde la concepción de las redes de computadoras, la seguridad ha sido un tópico ampliamente discutido. La protección del proceso de la información se requiere porque la información puede ser comprometida por ignorancia, inadvertencia, accidente o malicia. El objetivo fundamental de la seguridad es preservar la integridad de la información que ha sido enviada o almacenada a través del sistema. Una definición más precisa es: el estado de certeza de que los archivos de datos y programas computarizados no pueden ser accesados, obtenidos, o modificados por personal no autorizado [Wadron92].

Para lograr la seguridad de la información se deben contemplar los mecanismos de seguridad –procedimientos operativos, características de hardware y software, procedimientos administrativos y una combinación de estos– implementados para detectar y prevenir amenazas pasivas o activas en cualquier componente del sistema de información. Mantener un sistema seguro (o confiable) significa garantizar los siguientes aspectos: confidencialidad, integridad, disponibilidad, consistencia, autenticidad, control de acceso y auditoría [Esquivel04].

En la década pasada, las organizaciones vieron que serían más eficientes y productivas si adoptaban tecnologías de información y comunicaciones. Estudios previos demostraron que las organizaciones, en el manejo de controles de seguridad de la información, se habían concentrado en la presencia o ausencia de estos, en lugar de enfocarse a la calidad y eficacia [Baker07]. El amplio uso de las redes de computadoras e Internet han ocasionado un incremento paralelo en lo concerniente a la seguridad, particularmente en la protección contra actividades hostiles [Green07], [Chunyue06].

Las amenazas de seguridad en los sistemas de cómputo se han incrementado considerablemente con la inclusión de virus informáticos, negación de servicios, explotación de

vulnerabilidades, etc. Muchos mecanismos de seguridad han sido introducidos para neutralizar estas amenazas, pero ninguno puede prevenirlo todo de manera completa. Las amenazas de seguridad a los sistemas de cómputo han resaltado la importancia de la detección de anomalías [Islam05].

La Detección de Anomalías (DA) es el proceso del registro de eventos que ocurren en un host o en una red de computadoras, y efectuar el análisis de estos para detectar signos de anomalías. Una Anomalía se puede definir como un intento a comprometer la confidencialidad, integridad, disponibilidad, o traspasar los mecanismos de seguridad de un sistema de cómputo.

Los Sistemas de Detección de Intrusos (IDS, Intrusion Detection Systems), tienen el propósito de proveer de un marco estructurado que permita descubrir, reportar y posiblemente reparar intrusiones pasadas y recientes en tiempo real. Y se recomienda que para ataques en progreso, el IDS debe implementarse en conjunto con el sistema operativo en el host [Foo04].

- En el presente trabajo de tesis se propone la optimización de los parámetros de un HMM utilizando programación evolutiva. El sistema evolutivo se encargará del diseño automático del modelo.

- Donde una vez evolucionado el modelo HMM, se le usará como una herramienta de apoyo en la de detección de anomalías en secuencias de observación.

- Se tiene la hipótesis que los modelos probabilísticos son herramientas de gran ayuda en la detección de anomalías en secuencias de datos.

- En esta tesis se presentará toda la información que se ha utilizado en el desarrollo e implementación de los HMM, evolución y aplicación en la detección de anomalías, así como los resultados que se esperaban.

Capítulo 3

Conceptos preliminares

En este capítulo se describirá en detalle a los HMMs, su uso o aplicaciones en diversas áreas del conocimiento, los tipos y aplicaciones de estos, la arquitectura y componentes que los forman. También se abarcarán los conceptos básicos de los Algoritmos Genéticos (GAs), la relación que existe entre estos y la optimización de los parámetros de un HMM. Así como una breve descripción de la Programación Evolutiva (EP).

3.1. Modelos ocultos de Markov (HMMs)

En esta sección se hará una mención de los orígenes de los HMMs, su clasificación y aplicaciones, así como la arquitectura que poseen. Más adelante se tratarán los HMMs Univariados y los HMMs Multivariados, así como HMMs Discretos y HMMs Continuos.

3.1.1. Inicios de los HMMs

Los HMMs derivan de los Procesos de Markov o Cadenas de Markov, que son un proceso estocástico. Se pueden ver como una serie de eventos, en la cual la probabilidad de que ocurra un evento depende del evento inmediato anterior. Una cadena de Markov se puede denotar por $\{X_k\}_{k \geq 0}$, donde k es un índice entero [Taylor98], [Cappé05].

Los HMMs no requieren de datos anómalos para "aprender" o distinguirlos de los normales, ni requiere de reglas predictivas explícitas [Warrender99]. Los HMMs también son conocidos como Procesos Ocultos de Markov (HMP, Hidden Markov Processes), que son Cadenas de Markov homogéneas con estados finitos de tiempo discreto observadas a través de canales invariantes sin memoria en tiempo discreto. Donde el canal se considera

formado por un conjunto finito de transacciones indexados por los estados de la cadena de Markov [Ephraim02].

3.1.2. Tipos de HMMs

En [Rabiner89] se describen a detalle los tipos de HMMs básicos, como son los Ergódicos, No Ergódicos, y los Autoregresivos. Así como también describe los modelos "izquierda-derecha". Para evitar repetir lo que en muchos documentos y trabajos se informa, nos permitiremos remitir al lector interesado en estos aspectos a la fuente original de esta clasificación, por lo que recomendamos que para más detalle ver [Rabiner89].

En el trabajo de Nicholl et al. [Nicholl08] se mencionan los diversos enfoques que los HMMs han adoptado, para ser utilizados en el reconocimiento de rostros. Estos enfoques tratan de representar la estructura de dos dimensiones de la imagen de un rostro:

- Uno de estos enfoques es el 1D-DHMM, de HMM Discreto de una dimensión, el cual modela la imagen de la cara usando dos HMM estándar. Uno para las observaciones en dirección vertical y otro para las observaciones en dirección horizontal.

- Otro modelo es el Pseudo-2D HMM (2D-PHMM), el cual se utiliza para modelar la secuencia de columnas de una imagen. Cada estado que forma al modelo 2D-PHMM, es un 1D-DHMM.

- En el mismo documento de Nicholl et al. [Nicholl08], mencionan también un enfoque llamado Baja-Complejidad 2D-HMM (LC 2D-HMM), en este modelo se permiten las transiciones entre estados de manera horizontal y vertical. Se describe que la complejidad del LC 2D-HMM es más sencilla que la de 2D-PHMM. Sin embargo, su certeza en el reconocimiento es bajo.

- Existe otro enfoque llamado HMM Jerárquico (HHMM, Hierarchical), que se utiliza en el análisis de contenido de video. Este enfoque se utiliza principalmente para modelar dominios con estructura jerárquica. El problema que presenta el algoritmo es que toma mucho tiempo de procesamiento en el análisis de las observaciones, haciéndolo impráctico para algunos dominios.

- Un último enfoque es el HMM Estructural (SHMM), el cual maneja procesos de clustering sin supervisión, enfocado al reconocimiento de manuscritos. Este enfoque

permite que las propiedades estadísticas y estructurales de un patrón, puedan ser representados en el mismo marco probabilístico.

En general, todos estos enfoques y estructuras de HMMs, mencionados en [Nicholl08], fueron empleados para el reconocimiento de rostros y texto manuscrito, así como para el análisis de contenidos de video. Por lo que los autores vieron la necesidad de modificar o ampliar la estructura de los HMMs estándar. Este trabajo propone arquitecturas orientadas al análisis de datos discretos y diseñados por los expertos del área, en base a su experiencia.

En el trabajo de Pachter et al. [Pachter01] tratan la alineación de genes con el uso de un Par de HMM (PHMM), y para encontrar la secuencia de genes utilizan un HMM Generalizado (GHMM). En su marco de trabajo hacen una extensión de ambos modelos para generar el Par Generalizado de HMMs (GPHMM), que lo utilizan para hallar genes en cruza de especies y alineación de proteínas en el campo de la biología molecular. En esta investigación, los modelos propuestos, son diseñados en base a la experiencia. Las observaciones con las que se trabajan son secuencias de genes perfectamente determinados, es decir, son valores discretos.

En el documento de Brewer et al. [Brewer06], se mencionan otras extensiones a la arquitectura de los HMMs, como: HMM Factorial (FHMM), Árboles de Decisión Ocultos de Markov (HMDT, HM Decision Trees) y HMMs Emparejados (CHMMs, Coupled HMMs). Brewer y sus compañeros utilizan este último esquema para un análisis digital forense en interacciones sospechosas, apoyados de un toolbox de MatLab para la creación del CHMM. Proponen el esquema de CHMM como un posible método para el modelo de sospechas múltiples en un caso de de forensia digital. Concluyendo podemos decir que este trabajo se basa en el diseño automático del CHMM por una herramienta comercial. La arquitectura del CHMM se encuentra definida por el conocimiento de los investigadores, además, el modelo se basa en observaciones de valores discretos.

3.1.3. HMM discretos y HMM continuos

Los procesos del mundo real producen secuencias de símbolos observables. Estos símbolos pueden ser discretos (lado de una moneda, puntos de la cara de un dado, etc.) o continuos (temperatura, velocidad de viento, ejemplos del habla, etc.). El problema consiste en modelar la señal que explique y caracterice la ocurrencia de los símbolos observados. Si tal modelo existe, entonces se puede utilizar para identificar o reconocer otras secuencias

de observación. En base a la señal y la dinámica del sistema se puede decidir la forma del modelo y el tipo de observación a estudiar: determinística o estocástica. [Rabiner86].

En base a lo anterior, existen en la literatura HMMs Discretos, caracterizados porque su función de densidad de probabilidad de observación es discreta, y los HMMs Continuos, que tienen una función de densidad de probabilidad de observación continua. En seguida, se describen las características de estos modelos.

HMM Discreto (HMMD). Este HMM tiene un alfabeto, o conjunto de símbolos, discreto. Sus valores u observaciones se encuentran perfectamente definidos. Por lo que en cada estado del modelo, se encuentra definida una función de distribución de probabilidad discreta. Cada símbolo observado X tiene una distribución de probabilidad dada por la expresión $\mathrm{F}(X) = \mathrm{P}(X = x)$.

HMM Continuos (HMMC). En este tipo de HMM los valores a observar son de tipo continuo. Se considera que existe un espacio de observación infinito. El alfabeto no es un conjunto numerable, es decir, el conjunto de posibles valores de una observación X, abarca todo un intervalo de números reales. Además, se utilizan funciones de densidad de probabilidad (*pdf*) continuos en cada estado del modelo. Para calcular la distribución de probabilidad para una observación X continua, se utiliza la siguiente expresión $\mathrm{F}(X) = \int_{X-\epsilon}^{X+\epsilon} f(x)dx$.

Una descripción de estos modelos, HMMD y HMMC, se verá en secciones posteriores, enfocados al análisis de series de tiempo univariadas y multivariadas.

3.2. HMMs univariados y multivariados

En esta sección se presentan los conceptos de HMMs Continuos Univariados y HMMs Continuos Multivariados. También se proporciona una breve distinción entre Mezcla Gaussiana y Multidistribución Gaussiana.

3.2.1. Procesos estocásticos

El campo de la estadística tiene sus fundamentos en la teoría matemática de la ley de los grandes números y en la teoría analítica de las probabilidades. La estadística y la probabilidad tienen que ver con la incertidumbre: la probabilidad con eventos inciertos; la

estadística con mecanismos inciertos. La probabilidad se hace preguntas como: "dadas estas condiciones, ¿qué tipos de salidas se pueden anticipar?". La estadística trabaja en sentido opuesto: "¿Qué tipo de condiciones deben darse para producir estas salidas?" [Link10].

Desde nuestro caso de estudio en particular, nos interesa conocer que dados ciertos eventos, o comportamientos de objetos de estudio, qué se espera o qué se puede anticipar como salida. Es decir, nos interesa estudiar el comportamiento de un conjunto de variables aleatorias a lo largo del tiempo, y sabemos que estaremos trabajando con procesos estocásticos.

En general las investigaciones presentadas en el estado del arte trabajan con procesos estocásticos (variables aleatorias), para producir modelos que nos permitan hacer predicciones sobre el comportamiento futuro de un proceso. Es decir, es una sucesión de variables aleatorias que evolucionan en función de otra variable, generalmente el tiempo.

Un ejemplo particularmente importante lo proporcionan las denominadas "Series de Tiempo" o "Series Temporales", que registran observaciones de determinado proceso en función del tiempo. Podemos definir un proceso estocástico como una familia {X(t), t ∈ T} de variables aleatorias, clasificadas mediante un parámetro t, que varía en un conjunto T.

Ahora bien, un fenómeno físico complejo se describe con un modelo consistente en un conjunto de variables, definidas por funciones reales del tiempo. Así, llamamos determinística a una señal o magnitud que podemos predecir en cualquier instante del tiempo (dentro del intervalo de interés) y la representamos con una función real que, por lo general, es continua (en segmentos), por ejemplo:

$$X_d = t_0 \ Sin(2\pi); \ t > t_0 \geq 0$$

donde t_0 es el tiempo inicial.

Por el contrario, si la señal es impredecible en cualquier instante del tiempo (dentro del intervalo de interés) la llamamos aleatoria y la representamos con una variable aleatoria que se da en función del tiempo y, por lo general, es discontinua, por ejemplo:

$$X_a = t_0 \ Sin(2\pi + \tau); \ t > t_0 \geq 0$$

donde t_0 es el tiempo inicial y τ es la función de densidad de probabilidad uniforme.

Entonces, podemos deducir que a un conjunto de variables aleatorias dependientes del tiempo se le llama proceso estocástico (aleatorio) y es posible representarlo por el vector

del proceso estocástico v(t) = $[v_1(t), v_2(t), \ldots, v_n(t)]$ T con dimensión n [Link10].

Se puede decir que las series de tiempo univariadas han sido estudiadas ampliamente, en cambio las series de tiempo multivariadas han sido menos estudiadas [Kirshner05]. El término Multivariada o Multivariable se usa cuando existen dos o más variables siendo analizadas simultáneamente [Kirshner05].

3.2.2. HMMs continuos univariados

En general la arquitectura mostrada en Rabiner [Rabiner89], es el ejemplo sencillo para un HMM Discreto orientado a una serie de observación univariable. Las características son las que a continuación se describen.

El modelo consta de:

N, un número dado de estados que forman al HMM.

V, número de símbolos que componen el alfabeto.

A = a_{ij}, una matriz con una distribución de probabilidad de transición entre estados.

B = $b_{j,k}$, una matriz con una distribución de probabilidad de emisión de símbolos. Indicando la probabilidad de observar un símbolo k, en un estado j.

Π = π_i, que es la distribución de probabilidad de iniciar algún estado i.

La única característica que marca la diferencia entre un HMM Discreto de un HMM Continuo, es la matriz *B*. Para un HMMD, existe en el modelo un conjunto finito de V símbolos o alfabeto. Con una distribución de probabilidad definida para cada símbolo o elemento observado.

En cambio, para un HMMC el alfabeto V no es un conjunto numerable. Es decir, que los valores que cada símbolo observado puede tomar, abarca un intervalo de números reales, y por lo tanto no se puede utilizar una probabilidad de distribución similar a la de uno discreto. Por esto se utiliza una función de densidad de probabilidad (*pdf*) para valores continuos.

En nuestro caso de estudio se utilizó, como nuestra *pdf* en los HMMs Univariados, la *Distribución Normal* de una sola dimensión para el cálculo de la probabilidad de una secuencia de observación. La Distribución Normal, también conocida como Distribución de

Gauss o Distribución Gaussiana, es la distribución de probabilidad que con más frecuencia aparece en estadística y teoría de probabilidades. Esta función tiene dos propiedades fundamentales [Link10]:

- Su función de densidad es simétrica y con forma de campana, lo que favorece su aplicación como modelo a gran número de variables estadísticas.
- Es límite de otras distribuciones y aparece relacionada con multitud de resultados ligados a la teoría de probabilidades, gracias a sus propiedades matemáticas.

Su función de densidad está dada por (3.1):

$$f_{\mu,\sigma}(x) = \frac{1}{\sigma\sqrt{2\pi}}\, e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{3.1}$$

donde μ es la *Media* y σ es la *Desviación Estándar* (σ^2 es la Varianza). Cuando $\mu = 0$ y $\sigma = 1$, la distribución se conoce como *Normal Estándar*, dada por la función (3.2).

$$f_{ne}(x) = \frac{1}{\sqrt{2\pi}}\, e^{-\frac{x^2}{2}} \tag{3.2}$$

Muchas variables aleatorias continuas presentan una función de densidad cuya gráfica tiene forma de campana. La importancia de la distribución normal se debe principalmente a que hay muchas variables asociadas a fenómenos naturales que siguen el modelo de la normal: caracteres morfológicos de invididuos, caracteres Fisiológicos, caracteres sociológicos, caracteres psicológicos, nivel de ruido en telecomunicaciones, errores en mediciones de magnitudes y valores estadísticos muestrales como la media.

En el presente trabajo de tesis se tomó como referencia la función de distribución normal por sus características mencionadas y por el teorema del límite central (que indica que en condiciones muy generales la suma de k variables aleatorias independientes, se aproxima a la función de distribución normal). Se utilizó la distribución normal en los HMMs para el cálculo de la probabilidad de los valores observados. Pero no se descarta el posible uso de otras funciones para el cálculo de de probabilidad en los HMMs, incluso con la posibilidad de tener mejores resultados, de acuerdo a las series de tiempo y al análisis de los datos, que la función de distribución normal utilizada en este trabajo, para la detección de anomalías.

Se definió en cada estado del HMMC una *pdf*, que es en realidad una función de distribución normal. En cada estado del HMM, se tiene a la matriz B = $b_{j,k}$, definida de la siguiente manera:

$$b_{j,O} = \Theta[O,\ \mu_j,\ \sigma_j];\ 1 \leq j \leq N \tag{3.3}$$

donde μ_j es la *media* y σ_j es la *desviación estándar* para ese estado j en particular. Θ, representa nuestra *pdf*, que proporcionará el valor de probabilidad que se obtiene para el valor O en el estado dado.

Para concluir, podemos decir que estos modelos de HMMs univariables únicamente toman como referencia de observación un solo símbolo, en este caso un valor real, en cada marca de tiempo t. Esto es, la serie de tiempo con la que trabajan es una secuencia sencilla de valores reales de una variable aleatoria.

3.2.3. HMMs continuos multivariados

Los modelos multivariables o multivariados tratan con un conjunto de valores de variables que son analizadas de manera simultánea. Los conjuntos de datos multivariables de varias dimensiones ocurren en un gran número de dominios de problemas. Por ejemplo el comportamiento de la bolsa de valores, el patrón del habla y de la música, ciencias atmosféricas, detección de anomalías, etc. El modelado y predicción de tales datos es un problema importante en dichas áreas. Por eso es importante proponer y desarrollar, técnicas de análisis de datos confiables y flexibles para series de tiempo de datos multivariables.

Existe un número significativo de literatura acerca de series de tiempo univariables. El modelado del vector de series para dominios finitos es un área de investigación relativamente nueva. El modelado de datos mixtos (valores discretos y contínuos) es todavía menos explorado debido a la complejidad de moderar interacciones entre variables finitas y valores reales [Kirshner05].

Un HMM Multivariado es un HMM que observa un vector de variables, en lugar de un solo valor (HMM Univariable). El espacio de búsqueda en un ambiente multivariado se vuelve más complejo.

3.2.4. Multidistribución gaussiana

La distribución de probabilidad más usada en el análisis de datos multivariados es aquella derivada de la Distribución Multinormal (también conocida como Distribución Gaussiana Multivariable) [Wolfram09]. La distribución normal multivariable es una generalización de la distribución normal unidimensional o univariable a dimensiones superiores.

$$f(x_1, x_2, \ldots, x_d) = \frac{1}{(2\pi)^{d/2}\ |\Sigma|^{1/2}}\ e^{[-\frac{1}{2}(x_i-\mu)^T\ \Sigma^{-1}\ (x_i-\mu)]} \tag{3.4}$$

donde Σ es la *matriz de covarianza de tamaño* $d \times d$, $|\Sigma|$ es su determinante, μ es el vector de tamaño d de valores esperados; y x es el vector de tamaño también d conteniendo los valores de las variables. Si $|\Sigma| = 0$, el vector x tiene una *distribución degenerada*; lo que significa que no existe variabilidad en algunos de sus componentes. Con un análisis previo a los datos de las variables aleatorias, se podría determinar qué variables podrían ser susceptibles de eliminarse haciendo un análisis de componentes principales [Shlens09]. Si algunas variables aleatorias se mantienen constantes, entonces se podría hacer el estudio de la dinámica de un sistema basándose en las variables que sí muestran cambios en el tiempo y reflejan el comportamiento del sistema a modelar.

Una función de distribución de probabilidad multivariable está compuesta de:

- μ, un vector de d medias. Donde d es igual al número de variables que intervienen en el análisis.

- Σ, la matriz de Covarianzas simétrica de dimensión $d \times d$, que guarda la varianza y la correlación que existe entre las variables que participan.

En nuestro trabajo se plantea utilizar a la matriz de distribución de probabilidad $B = b_{j,O_d}$, donde para calcular la probabilidad del vector de valores O_d de dimensión d, se tendrá la Ecuación (3.5).

$$b_{j,O_d} = \Theta[O_d,\ \mu_j,\ \Sigma_j];\ 1 \leq j \leq N. \tag{3.5}$$

La función *pdf* multivariable Θ se encuentra representada en la Ecuación 3.4. Donde la observación O_d es igual a $[x_1,\ x_2, \ldots,\ x_d]^T$, tomando también a μ_j y Σ_j como valores dentro de la ecuación. Es decir, que para nuestro modelo HMMC el conjunto de *pdfs*

estará conformado por funciones de Distribución Normal Multivariable (ver la Ecuación 3.4), de cada uno de los estados que forman al HMMC.

La Distribución Normal Multivariable, a diferencia de la Mezcla Gaussiana, no necesita estimar los coeficientes de peso o ponderación. Pero si se necesitarán estimar las medias (μ) y matrices de covarianza (Σ) de cada *pdf multigaussiana*, en cada estado del HMMC.

3.3. Aplicación de los HMMs

Los Modelos Ocultos de Markov (HMMs) son procesos doblemente estocásticos con un proceso estocástico subyacente que no es observable directamente, pero que puede ser observable a través de otro conjunto de procesos estocásticos producidos por la secuencia de símbolos observados [Juang85], [Warrender99], [Kotsalis06].

La aplicación inicial y más común de los HMMs fue la del reconocimiento automático del habla, donde los HMMs fueron utilizados para caracterizar las propiedades estadísticas de una señal [Rabiner86], [Rabiner89], [Dai94], [Van-Le99].

Los HMMs son ampliamente usados para el reconocimiento del habla y en el modelado de secuencias de ADN. También han sido empleados en el área de bioinformática y a una gran variedad de aplicaciones fuera del área del reconocimiento del habla, tales como reconocimiento de texto manuscrito, reconocimiento de patrones en biología molecular [Won04], [Won05], reconocimiento de rostros [Chien08], y reconocimiento de expresión facial [Miners05]. Variantes y extensiones de los HMMs incluyen econométricas, series de tiempo y procesamiento de señales [Bengio97].

También se han utilizado en el estudio de los errores en los enlaces de red inalámbricos 802.11b, se han utilizado en proyectos de robots asistentes participando en el área de visión y navegación [López05], al monitoreo de actividades humanas [Al-ani07], se han utilizado en sistemas de detección de intrusos [Zhecheva07], [Lane03], [Khanna06], en detección de fallas y procesamiento de imágenes [Deng06], [Vanluyten07].

Otros ejemplos de las diversas áreas donde se pueden emplear los HMMs son: bioinformática, visión, segmentación de música, modelado de tránsito carretero, predicción de precipitaciones, reconocimiento de rostros, secuencias de ADN, clasificación por categorías y análisis forense digital [Pachter01], [Kirshner05], [Deng06],[Brewer06], [Nicholl08].

3.4. Arquitectura de un HMM

Un HMM está formado por un número finito de estados conectados por transiciones. En la Figura 3.1 se pueden observar los estados, representado por rectángulos, que conforman al HMM. Las líneas que interconectan a los estados son las transiciones con un peso o valor de probabilidad de transitar de un estado a otro. El modelo HMM puede generar una secuencia de observaciones dependiendo de las probabilidades de su estado inicial y de sus transiciones. Esto significa que un HMM está representado por tres conjuntos de probabilidades (probabilidad del estado inicial, probabilidad de transición entre estados, y la probabilidad de observar un símbolo en cada estado).

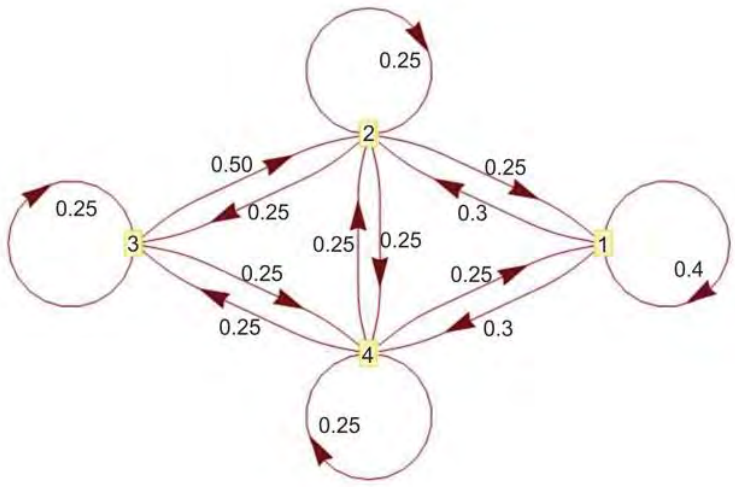

Figura 3.1: Representación de un HMM con 4 estados.

El Modelo de Markov es oculto porque no sabemos qué observación lleva a qué estado. Es decir, se desconocen los estados y las interconexiones, sólo se observan los símbolos generados por los estados. Un HMM está definido, entre otros, básicamente por estos tres parámetros:

$A = a_{i,j}$ es la matriz de probabilidad de transición entre estados. Donde $a_{i,j} = \mathrm{P}(q_{t+1} = \mathrm{S}_j \mid q_t = \mathrm{S}_i)$, $1 \leq \mathrm{i}, \mathrm{j} \leq N$.

$B = b_{j,k}$ es la matriz de probabilidad de emisión, indicando la probabilidad de

observar un símbolo k, en un estado j. Es decir, $b_{j,k}$ es igual a la $\mathrm{P}(v_k \text{ en } t \mid q_t = S_j)$. Para el caso continuo univariado, en este trabajo de tesis, se utiliza la Ecuación 3.3.

$\Pi = \pi_i$ es la distribución de probabilidad de iniciar en el estado i. Esto es, $\pi_i = \mathrm{P}(S_i \text{ en } t = 1)$.

Por lo que un HMM puede ser definido por la tripleta [Nicholl08]:

$$\lambda = (A, B, \Pi) \tag{3.6}$$

En el documento de Rabiner [Rabiner86] se mencionan otras características que definen a un HMM discreto:

T = Longitud de la secuencia de observación.

N = Número de estados en el modelo.

M = Número de símbolos a observar.

$S = S_1, S_2, \ldots, S_N$ representación de los estados.

$V = v_1, v_2, \ldots, v_M$ conjunto discreto de posibles símbolos a observar.

q_t = Representación el estado actual en el tiempo t.

Estas especificaciones envuelven la selección de un HMM, como el número de estados N, los M símbolos y la especificación de λ. De todas las especificaciones π es la menos importante, y la distribución de probabilidad B es la más importante. La distribución A para algunos problemas también es importante [Rabiner86].

Existen tres problemas básicos de interés que deben ser resueltos por los HMMs, para ser útiles en las aplicaciones del mundo real:

1. Dada la secuencia de observación $\mathrm{O} = o_1, o_2, \ldots, o_T$, y el modelo λ, ¿cómo procesamos eficientemente $\mathrm{P}(\mathrm{O} \mid \lambda)$, la probabilidad de la secuencia de observación, dado el modelo? Es decir, la probabilidad de que una secuencia de observación haya sido generada por el modelo λ.

2. Dada la secuencia de observación $\mathrm{O} = o_1, o_2, \ldots, o_T$, y el modelo λ, ¿cómo escogemos una correspondiente secuencia de estados $\mathrm{Q} = S_1, S_2, \ldots, S_T$ la cual es óptima en algún sentido (i. e. la que mejor explica las observaciones)? En otras palabras, determinar la probabilidad más alta de secuencias de estados, dada una secuencia de observación y el modelo λ.

3. ¿Cómo ajustar o re-estimar los parámetros de un modelo λ para maximizar $\mathrm{P}(\mathrm{O} \mid \lambda)$?

Los HMMs tratan con estos problemas dentro de un marco probabilístico; ofrecen la ventaja de tener fuertes fundamentos estadísticos y ser computacionalmente eficientes. Pero tienen una desventaja: se necesita conocer de antemano la topología del modelo. Esto es, que para construir un HMM necesitamos decidir cuántos estados tendrá y qué transiciones entre estados existirán. Una vez que la estructura del modelo está diseñada, los parámetros de transición y emisión necesitarán ser estimados con los datos de entrenamiento.

El algoritmo de Baum-Welch (BW) [Rabiner89], [Welch03] se usa para entrenar los parámetros del modelo. Este algoritmo utiliza un método de Maximización de Expectativa, el cual dada una configuración inicial, ajusta los parámetros del modelo a una maximización local de probabilidad con los datos. El entrenamiento con el algoritmo de BW sufre del hecho de que encuentra un máximo local, y esto es sensible a la configuración inicial de los parámetros y el rendimiento del modelo. Para más información al respecto de los HMMs, véase [Rabiner89].

La problemática que estamos abarcando y para la cual proporcionamos una alternativa de solución en esta tesis, es con respecto al problema número 3 "Ajustar los parámetros de un modelo λ para maximizar $\mathrm{P}(\mathrm{O} \mid \lambda)$". Estamos partiendo desde cero en el diseño de la estructura de un HMM multivariado, el sistema sólo necesita la secuencia de observación como dato de entrada principal, para generar automáticamente un HMM. El sistema se encarga de ajustar, durante el proceso de diseño, los valores óptimos de los parámetros del HMM. Estos parámetros estimados son las matrices: A, B y, como estamos trabajando con vectores de observación, también determina μ y Σ.

Las restricciones que se le necesitan proporcionar al sistema evolutivo son: número de población de individuos, número de generaciones por iterar, y el número de estados mínimo y máximo que podría tener el HMM; y con estos valores el sistema sólo necesita, como comentamos anteriormente, la serie de tiempo y tiempo máquina, para poder procesar las generaciones de poblaciones de individuos y obtener de éstas el mejor HMM evolucionado.

Cabe mencionar que una vez generado nuestro HMM, obtenido a través del proceso evolutivo, se encuentra preparado y listo para detectar las probabilidades de las secuencias de observación subsecuentes. Esto significa que también estamos proporcionando una alternativa de solución a la problemática número 1 de los modelos HMMs "Dada la secuencia de observación $\mathrm{O} = o_1, o_2, \ldots, o_T$, y el modelo λ, ¿cómo procesamos eficientemente $\mathrm{P}(\mathrm{O} \mid \lambda)$, la probabilidad de la secuencia de observación, dado el modelo?". Esta respuesta la podemos responder debido a que contamos con el HMM que modela la dinámica normal del

sistema, y que el modelo proporcionará las probabilidades de las secuencias de observación posteriores, indicando la certeza que tiene cada serie de observación de haber sido generada por el modelo.

3.5. Algoritmos genéticos

En esta sección se describirá brevemente la historia de los GAs, la relación que existe entre los GAs y los HMMs; y una breve descripción de la Programación Evolutiva, tema que se utiliza en la implementación y desarrollo de este trabajo de tesis.

3.5.1. Historia

En los primeros días de la Inteligencia Artificial (AI) durante 1950 y 1960, la investigación sobre el entendimiento de la inteligencia por la construcción de sistemas computacionales siguió dos tipos de pensamientos diferentes llamados: de Abajo Hacia Arriba (Bottom-Up) y de Arriba Hacia Abajo (Top-Down) para el modelado del comportamiento inteligente.

Pensamiento de Abajo Hacia Arriba la implementación de la inteligencia comienza a nivel neuronal a fin de diseñar cerebros artificiales. Una vez entendida la construcción de redes neuronales, mecanismos tales como aprendizaje, memorización, asociación de patrones, planificación y control –todas las tareas inteligentes realizadas por un cerebro natural–, se podrían construir sobre máquinas adaptativas. Un número enorme de arquitecturas de redes neuronales y algoritmos de aprendizaje han demostrado un acercamiento subsimbólico, pero aún limitado en cómo crear sistemas inteligentes.

Pensamiento de Arriba Hacia Abajo este enfoque toma la aproximación desde una perspectiva psicológica. La inteligencia artificial se logra por el análisis y modelado de procesos a través de un conjunto de reglas estructuradas como se refleja, por ejemplo, en los sistemas expertos o demostradores de teoremas matemáticos. Este enfoque simbólico conocido como *programación heurística*, confía principalmente en reglas útiles que finalmente fueron difíciles de aplicar y ajustar a ambientes ruidosos y dinámicos.

Un elemento importante del comportamiento inteligente de un organismo, es su capacidad de modelar su ambiente externo basado en observaciones. Por lo tanto un organismo

es capaz entonces, de fabricar predicciones sobre su ambiente en futuros acontecimientos, y poder adaptarse y evolucionar [Jacob01].

La evolución en la naturaleza, un ejemplo del proceso natural de adaptación, ha dado como resultado una fantástica diversidad de formas de vida. La población de organismos, adaptándose a las condiciones de su entorno particular, forman equipos de cooperación y competencias en una interacción de mecanismos de selección y variación. Las características de los organismos, codificados en sus genes, varían de generación en generación. Finalmente los individuos que prevalecerán –en gran medida por sus habilidades y características especiales– serán los más aptos a las condiciones de su medio ambiente.

En los años de 1970, de la mano de John Henry Holland, surgió una de las líneas más prometedoras de la inteligencia artificial: la de los Algoritmos Genéticos (GAs, Genetic Algorithms), y son llamados así porque se inspiran en la evolución biológica de la naturaleza.

El término GA describe la idea básica: técnicas algorítmicas, inspiradas por principios genéticos, que son usadas para simular procesos evolutivos. El objetivo es diseñar e implementar sistemas robustos y adaptables, siguiendo el paradigma de la naturaleza para la evolución de estructuras genéticas. John H. Holland propuso los primeros modelos del proceso de adaptación; definiendo adaptación como un proceso de modificación de estructura progresiva, el cual lleva a mejorar el rendimiento de la interacción del sistema con su entorno [Jacob01].

Estos algoritmos hacen evolucionar una población de individuos sometiéndola a acciones aleatorias semejantes a las que actúan en la evolución biológica (mutaciones y recombinaciones genéticas), así como también a una selección de acuerdo con algún criterio, en función del cual se decide cuales son los individuos mejor adaptados –los que sobreviven– y cuales los menos aptos –mismos que son descartados [Barrera08].

Un algoritmo genético es un método de búsqueda dirigida basada en probabilidad, bajo una condición muy débil, que el algoritmo mantenga elitismo. Es decir, que guarde siempre al mejor elemento de la población sin hacerle ningún cambio [Martínez09].

Los GAs han sorprendido porque se han aplicado a un gran rango de dominios del conocimiento, desde resolver problemas de optimización técnicos, optimización de funciones matemáticas, hasta simular la evolución del comportamiento social de hormigas [Jacob01].

3.5.2. Algoritmo genéticos y HMMs

Los parámetros iniciales de los HMMs se determinan durante un proceso iterativo llamado "entrenamiento". Existe un método iterativo para la optimización de los parámetros de un HMM, conocido como Algoritmo de Baum-Welch (BW). El algoritmo de BW realmente trabaja refinando o ajustando los parámetros que se han asignado al HMM. Es decir, el usuario debe diseñar la arquitectura y las conexiones entre los estados definidos, y el algoritmo de BW ajusta estos parámetros.

Otro problema que se presenta es que el algoritmo de BW puede converger en un máximo local, sin llegar a entrenar adecuadamente al modelo HMM, lo cual no es conveniente.

Una posible solución al problema de la convergencia a un máximo local, es utilizar los Algoritmos Genéticos (GAs, Genetic Algorithms). Los GAs utilizan una técnica de optimización global que puede ser usada en la optimización de los parámetros de un HMM [Korayem07], [Bhuriyakorn08].

Los GAs simulan el fenómeno de la evolución de la naturaleza. Su espacio de búsqueda se mapea dentro de un espacio genético. La posible solución se codifica dentro de un vector (cromosoma), y cada elemento de este vector se le llama gen. Se calcula, en cada generación, la aptitud de cada cromosoma, los mejores cromosomas se seleccionan y, después de realizar operaciones genéticas con estos, se obtiene la solución óptima [Zhang07]. Los GAs son una técnica de optimización robusta, la cual envuelve una población de soluciones, así como un conjunto de operaciones para moverse en su espacio de búsqueda [Korayem07].

La *selección*, *cruza* y *mutación* son tres operadore principales de los GAs, donde un individuo es el objeto de los operadores. Los GAs tienen buenas características que otros métodos clásicos no tienen –por ejemplo, los métodos de Avance-Retroceso (Forward-Backward) y el de Gradiente [Chau97]–. Se puede encontrar los pasos generales de los GAs en [Zhang07].

3.5.3. Programación evolutiva

Es indudable que la teoría de la evolución, enunciada por Charles Darwin a mediados del siglo XIX, provocó una revolución científica. Los conceptos de esta teoría, como el de selección natural, han permeado desde entonces casi todas las disciplinas científicas. Así, puede constatarse cómo han surgido áreas de estudio que van desde la evolución química

prebiótica y la evolución molecular hasta la evolución tecnológica, pasando por ideas de la evolución aplicadas a las ciencias sociales y a las ciencias cognitivas [Martínez09].

La *Computación Evolutiva* es una rama de la Inteligencia Artificial que involucra problemas de optimización combinatoria; se inspira en los mecanismos de la Evolución Biológica. Durante los años 50 se comenzaron a aplicar los principios de Charles Darwin en la resolución de problemas. Durante los años 60 y 70, varias corrientes de investigación independientes comenzaron a formar lo que ahora se conoce como *computación evolutiva* [Jacob01]:

- *Programación Evolutiva.* Nació en la década de 1960 y su creador fue el Dr. Lawrence J. Fogel, considerado el "padre de la inteligencia computacional". Este desarrollo comenzó como un esfuerzo encaminado a crear inteligencia artificial basado en la evolución de máquinas de estado finito.

- *Estrategias Evolutivas.* Fueron propuestas por Ingo Rechenberg y Hans-Paul Schwefel en la década de 1970. Su principal objetivo era el de optimizar parámetros.

- *Algoritmos Genéticos.* Fueron propuestos por John Henry Holland en 1975 y su motivación inicial fue la de proponer un modelo general de proceso adaptable [Martínez09].

La programación evolutiva toma como base la siguiente afirmación: si las poblaciones naturales se adaptan a su entorno por evolución, entonces podemos modificar selectivamente las estrategias de Solución de Problemas, codificadas como programas, y progresivamente ajustar estos programas a una tarea predefinida con un conjunto de restricciones del medio ambiente.

La evolución implica la adaptación de las poblaciones de organismos a su medio ambiente a menudo a través de métodos sofisticados de experimentación, desarrollados en muchos casos en cientos de millones de años.

La evolución, en los sistemas computacionales, toma los principios de adaptación de la naturaleza, basado en la mutación iterada y la selección. La evolución –forma de selección iterada y variada–, es el mecanismo principal en la naturaleza de adaptar individuos a un entorno específico. Donde la adaptación es un proceso de cambio de estructura progresivo, el cual resulta en un mejor comportamiento de un individuo en su interacción con un medio ambiente [Jacob01].

3.6. HMMs en la detección de anomalías

En el desarrollo de esta tesis se utilizaron mecanismos de *Programación Evolutiva*, logrando que un HMM, creado aleatoriamente, evolucionara para proporcionarnos un modelo confiable en la detección de comportamiento anómalo en un conjunto de series de datos.

La forma como se logra la obtención del HMM Continuo Multivariable (HMMCM), consiste en lo siguiente:

- Se inicia el sistema con una población, generada de manera aleatoria, con un número *pop* de cromosomas

  $$C = [c_1,\, c_2, \ldots,\, c_{pop}]^T$$

  donde cada cromosoma (c_i) representa un HMMCM completo.

- Cada HMMCM tiene las siguiente propiedades: contiene N estados, posee una matriz de transiciones A la cual tiene las siguiente características

  $$\sum_{j=1}^{N} a_{ij} = 1;\ con\ 1 \leq i \leq N,\ a_{ij} \geq 0.$$

  También tiene un vector π con la siguiente propiedad $\sum_{i=1}^{N} \pi_i = 1$, una matriz de medias μ y una matriz de covarianzas Σ.

- Después de generar la población de cromosomas, se procede a calificar cada cromosoma con la función de aptitud

  $$P(O_d \mid C) = \sum_{i=1}^{N} \alpha_T(i),$$

  donde $\alpha_T(i)$ representa la función *Forward* [Rabiner89] que nos permite calcular la probabilidad de que la secuencia de observación sea producida por el modelo, terminando en el estado i. La función consiste en calificar cada modelo c_k, obteniendo el valor de probabilidad total de c_k con la serie de tiempo O_d –de dimensión d–. Donde $O_d = [\mathrm{o}_{1d},\, \mathrm{o}_{2d},\, \ldots,\, \mathrm{o}_{Td}]$ con una longitud T. Para calcular de manera particular la probabilidad de una observación $O_{t+1,\,d}$ de la secuencia y terminar en un estado S_j en un tiempo *t+1*, se utiliza el proceso inductivo de la función α (Forward), dado por

  $$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(j)\ a_{ij}\right]\ b_j(O_{t+1,\,d});\ con\ 1 \leq j \leq N,\ 1\ \leq t \leq T-1$$

donde

$$b_j(O_{t+1,\,d}) = \Theta[O_d,\ \mu_i,\ \Sigma_j].$$

Tenemos que Θ está dado por la Ecuación (3.4); en este proceso es donde utilizamos la función de distribución gaussiana multivariable para calcular la probabilidad de observar a $O_{t+1,\,d}$ en el estado j en *t+1*.

- Otro paso en la evolución consiste en seleccionar un subconjunto de cromosomas C_x de C, y aplicar Ω a C_x para mutar los cromosomas de C_x. En donde $\Omega = \{$Copy, MutParams, AddTransition, DelTransition, MutTransition, AddState, DelState$\}$ son los operadores de mutación que pueden ser aplicados a un cromosoma. Estos operadores funcionan de la siguiente manera: *Copy* copia un cromosoma completo, *MutParams* se encarga de mutar los valores del vector μ y de la matriz de covarianza Σ de un estado del cromosoma, *AddTransition* agrega una nueva transición a un estado del cromosoma, *DelTransition* elimina una transición de un estado, *MutTransition* muta el valor de una transición de un estado, *AddState* agrega un nuevo estado al cromosoma, y *DelState* elimina un estado del cromosoma.

- De manera posterior, se procede a evaluar cada nuevo cromosoma mutado, y se seleccionan los cromosomas mejor evolucionados de toda la población: tanto cromosomas mutados como cromosomas originales.

- A cada generación se aplica el operador Ω al subconjunto C_x de la población. Los operadores de mutación en Ω cuentan con cierto valor de probabilidad definido para interactuar con los cromosomas. Donde la distribución de probabilidad entre los operadores debe ser la unidad,

$$\sum_{k=1} P(\Omega_k) = 1$$

Ω se utilizará en la población hasta llegar al número de generaciones solicitadas.

Al término de las generaciones, el sistema proporciona el cromosoma o HMMCM mejor evolucionado, listo para utilizarse en la detección de anomalías en las series de tiempo subsecuentes. El proceso evolutivo se encarga de crear y optimizar los parámetros del HMMCM de manera automática. El tiempo de procesamiento invertido en encontrar el modelo

óptimo, es directamente proporcional al número de generaciones a crear, la longitud de la secuencia de observación, la dimensión de las observaciones, el tamaño de la población y el número de estados que contengan los cromosomas.

3.6.1. Detección de anomalías usando ventana de tiempo deslizante

El proceso de detección de anomalías se lleva a cabo analizando subsecuencias de la observación de prueba. Estas subsecuencias están formadas por intervalos de observación temporalmente adyacentes y de longitud constante; a estas subsecuencias las llamaremos ventanas deslizantes de datos, o ventanas de tiempo. Las ventanas de tiempo contienen un subconjunto de datos contiguos de tamaño constante; el tamaño de la ventana lo podemos definir en cada análisis de la secuencia de prueba.

A continuación se describe el proceso que se emplea para usar un HMM en la detección de anomalías en series de tiempo de prueba.

- El primer paso consiste en contar con la secuencia de observación o serie de tiempo a analizar, así como los tamaños de ventana de datos a utilizar. Al sistema se le indica que tamaño de ventana w va a analizar, cual es el HMM λ a utilizar y también se le proporciona la serie de tiempo de prueba A.

- El sistema comienza a analizar las ventanas de datos deslizantes, generando las probabilidades para cada ventana en cada marca de tiempo. La ventana se desliza una unidad de tiempo a la vez, hasta llegar al último dato en la serie de tiempo de prueba. La probabilidad de cada ventana está dada por

$$windowProb_t = P_t^w(A_t \mid \lambda)$$

  donde A_t es la ventana de tiempo que comienza en t, con $A_t = [O_t, O_{t+1}, \ldots, O_{t+w-1}]^T$.

- Para detectar si existe algún comportamiento anómalo en los datos de la ventana actual A_t, simplemente se comparan los valores de probabilidad de las ventanas con el umbral de referencia. De esta forma si $windowProb_t \geq Threshold$ se considera que los datos observados en esa ventana son normales; en caso contrario el sistema señala una anomalía en los datos de la ventana.

- Esto se aplica para todas las ventanas que se definan para el análisis de datos.

- Una vez definido el modelo λ, se debe especificar el valor del umbral (*Threshold*). El parámetro Threshold se fija de forma arbitraria de acuerdo a un análisis del rango de probabilidades de las secuencia de datos con los que se entrenó el HMM. Teniendo cuidado en fijar el umbral en un valor adecuado, ya que si fijamos a Threshold = 1, todo sería Falso Positivo –las subsecuencias serían consideradas anomalías–; y si Threshold = 0, entonces todas las probabilidades darían Falso Negativo –todas las subsecuencias serían consideradas normales, presentaran o no anomalías–. Este es el único parámetro que requiere del conocimiento o experiencia previa del usuario, pero todo depende del tipo de sistema modelado.

Esta es la forma como el sistema lleva a cabo el análisis de los datos de prueba, que ayuda a determinar probabilísticamente si alguna secuencia de datos presenta anomalía o no. Este proceso es general, tanto para secuencias de observaciones univariadas como multivariadas.

Se observa que los HMMs pueden ser herramientas de gran ayuda en la detección de anomalías; el sistema evolutivo propuesto libera al usuario de la construcción y entrenamiento de los HMMs. El proceso de análisis de las ventanas de tiempo no representa gran complejidad, sólo requiere de un conocimiento previo del tipo de sistema que se modele y de la serie de tiempo de entrenamiento, para determinar el umbral y poder usarse el HMM en la detección de comportamientos anómalos.

Capítulo 4

Evolución de modelos ocultos de Markov

En este capítulo se describe el marco de trabajo que se utilizó para evolucionar los parámetros de un HMM Multivariado. También se describe la forma de utilizar el HMM evolucionado en la detección de anomalías en series de tiempo o secuencia de observaciones.

4.1. Evolución de HMMs

En esta sección se da a conocer el aspecto interno del sistema de detección de anomalías. El sistema está implementado por dos módulos separados, uno se encarga de la evolución de los HMMs, entrenados con la secuencia de observación proporcionada para el entrenamiento. El otro módulo, una vez generado el HMM con EP, tiene la función de utilizar al HMM en el análisis de las secuencias de observación de prueba. El planteamiento de creación, evolución y uso de los HMMs en la detección de anomalías proporciona un esquema completamente nuevo, a lo que existe en la actualidad en la literatura.

Existen trabajos de otros autores que han aplicado la computación evolutiva en el diseño y optimización de sistemas coneccionistas. Por ejemplo Christian Jacob [Jacob01] presenta un esquema para evolucionar autómatas de estado finitos. Sin embargo, a lo que el autor ha leído, no se han encontrado hasta el momento trabajos que propongan la evolución de HMMs, por consiguiente considero que esta es una de las contribuciones de este trabajo de tesis.

Para poder efectuar un proceso evolutivo, el sistema comienza creando una población

de individuos, representados por cromosomas, de manera aleatoria. El número de cromosomas se define como un parámetro al momento de ejecutar el sistema.

Posteriormente, cada cromosoma HMM de la población inicial generada aleatoriamente, se tiene que evaluar con respecto a la *serie de tiempo* dada y la función objetivo. Con estos dos elementos se pueden calificar los HMMs, o en otras palabras, determinar su valor de aptitud.

Una vez que la población inicial ha sido evaluada, se toman los cromosomas mejor calificados y se les aplican los operadores de mutación evolutivos, como son: *agregar o quitar algún estado*, *agregar o quitar una transición entre estados*, *mutar el valor de transición entre estados*, *mutar el valor del vector de medias y de la matriz de varianzas*.

Una vez generados los cromosomas hijos, cromosomas mutados por los operadores evolutivos mencionados, se agregan a la población original, y son evaluados. Después, nuevamente se seleccionan los mejores cromosomas que pasan a la siguiente generación, donde se vuelve a repetir el ciclo del proceso evolutivo y selectivo de los mejores cromosomas de cada generación, hasta llegar al número de generación deseada.

4.1.1. Arquitectura de un HMM a evolucionar

Como se ha hecho mención, cada cromosoma de una población generada inicialmente, es en realidad una estructura completa de un HMM. Cada cromosoma se encuentra constituido por los siguientes genes: valor de *Aptitud, Tamaño, Matriz de Transiciones entre estados, Matriz de Medias, Matriz de Covarianzas, y el vector Pi*. La Tabla 4.1 muestra el cromosoma completo compuesto de los genes mencionados, y a continuación se describen las características particulares de cada gen.

Tabla 4.1: Arquitectura de un HMM o Cromosoma.

Aptitud	Tamaño	Matriz de Transiciones	Matriz de Medias	Matriz de Covarianzas	Vector Pi

Aptitud. Representa el valor de aptitud del individuo, después de haber sido evaluado por la función objetivo (ver Ecuación 4.1).

Tamaño. Representa el número de estados que contiene el HMM.

Matriz de Transiciones. Contiene la matriz de probabilidad de transiciones entre los estados que constituyen al HMM. Se muestra un esquema en la Tabla 4.2, donde $a_{i,j}$

$= \mathrm{P}(\mathrm{q}_{t+1} = \mathrm{S}_j \mid \mathrm{q}_t = \mathrm{S}_i)$; con $1 \leq i,\ j \leq\ N$. Esto es, cada celda $\mathrm{a}_{i,j}$ representa la probabilidad de transitar a S_j en un tiempo $t + 1$, dado que en el tiempo t se está en S_i. Con la propiedad de $\sum_{j=1}^{N} a_{i,j} = 1;\ \forall i$.

Tabla 4.2: Gen que representa la Matriz de Transiciones.

Estados	S_1	S_2	$\cdots$	S_N
S_1	$\mathrm{a}_{1,1}$	$\mathrm{a}_{1,2}$	$\cdots$	$\mathrm{a}_{1,N}$
S_2	$\mathrm{a}_{2,1}$	$\mathrm{a}_{2,2}$	$\cdots$	$\mathrm{a}_{2,N}$
$\vdots$	$\vdots$	$\vdots$	$\cdots$	$\vdots$
S_N	$\mathrm{a}_{N,1}$	$\mathrm{a}_{N,2}$	$\cdots$	$\mathrm{a}_{N,N}$

Matriz de Medias. Esta matriz se encuentra constituida por los vectores de los valores de las medias de los estados. Cada media representa el valor promedio aritmético de las secuencias observadas de cada variable, con las cuales se entrenan los modelos. Es decir, esta matriz contiene la media de la distribución de la matriz B de cada estado. La matriz B en cada estado representa la probabilidad de que se emita un símbolo en cada estado. Se puede observar este esquema en la Tabla 4.3.

Tabla 4.3: Gen que guarda la Matriz de Medias del Cromosoma.

Estado	Medias		
	X_1	$\cdots$	$\mathrm{X}d$
S_1	$\mu_{1,1}$	$\cdots$	$\mu_{1,d}$
S_2	$\mu_{2,1}$	$\cdots$	$\mu_{2,d}$
$\vdots$	$\vdots$	$\cdots$	$\vdots$
S_N	$\mu_{N,1}$	$\cdots$	$\mu_{N,d}$

Matriz de Covarianzas. Los valores de la matriz son utilizados, al igual que la matriz de medias, como parámetros de entrada en la función de densidad de probabilidad (*pdf*). La función pdf que se utiliza es la distribución multigaussiana, que se encuentra definida en cada estado que constituyen al HMM. Las matrices de medias y covarianzas componen la matriz (B), la cual permite calcular la probabilidad de observar un símbolo en un estado determinado. Se puede observar la estructura de la matriz en la Tabla 4.4. En esta parte de la implementación cabe resaltar que por el momento, sólo se llena la diagonal principal de la matriz con los valores de las varianzas de las variables participantes. Los valores de las correlaciones de las variables no están contempladas aún en la evolución de los cromosomas.

Tabla 4.4: Gen que mantiene la Matriz de Covarianzas para cada Estado.

Variables Aleatorias	X_1	X_2	$\cdots$	X_d
X_1	$\sigma^2_{1,1}$			
X_2		$\sigma^2_{2,2}$		
$\vdots$			$\ddots$	
X_d				$\sigma^2_{d,d}$

Vector Pi. Este último gen (π_i), ver Tabla 4.5, representa la probabilidad de que la primera observación haya sido generada en el estado S_i. Donde $1 \leq i \leq N$.

Tabla 4.5: Gen con el Vector Pi del Cromosoma.

S_1	S_2	...	S_N

4.1.2. Función objetivo

Para poder determinar el grado de aptitud que posee cada HMM, o cromosoma de la población, con respecto a la serie de tiempo dada, utilizamos la función objetivo descrita en la Ecuación (4.1).

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i), \tag{4.1}$$

donde $\alpha_T(i)$ representa a la variable de avance α (*Forward*) [Rabiner89], la cual calcula la probabilidad total de la secuencia observada dado un HMM. Es decir, $P(O|\lambda)$ significa la probabilidad que tiene el HMM de haber sido el modelo que generó la secuencia de observación. En otras palabras, significa qué tan bueno es el modelo de HMM con respecto a la secuencia dada.

A continuación se describen los algoritmos usados, así como los parámetros que necesitan para iniciar el sistema de programación evolutiva.

4.2. Descripción de algoritmos empleados

En esta sección se describirán los algoritmos que son empleados para implementar los operadores evolutivos, que ayudan en la mutación de la estructura de un HMM. En cada

generación se aplican los operadores evolutivos a los individuos de la población, dependiendo de la distribución de probabilidad establecido para cada operador al momento de ejecutar el sistema evolutivo (ver Figura 4.1).

Se comenzará por describir el algoritmo que inicia la evolución de los parámetros de un HMM.

4.2.1. Algoritmo "Evolution"

Como se podrá observar en el **Algoritmo 1** Evolution, se reciben como parámetros de entrada: número de cromosomas *Parents* o tamaño de población a crear, número de descendencia o hijos, *Children*, y el número de generaciones *Generations*, que el sistema tendrá que alcanzar con las iteraciones. No existe por el moemnto una función de paro, el sistema llega hasta la última generación. Una descripción más detallada de estos valores se menciona a continuación:

Parents. Se fija el número de cromosomas padres que tendrá, al inicio y durante su ejecución, en el sistema durante el proceso de iteraciones.

Children. Es la cantidad de cromosomas hijos generados en cada iteración. Estos cromosomas se generan de la población con mejor calificación y después se integran a la población para ser calificadas, sumándose a los cromosomas padres existentes. Siempre manteniendo fijo el tamaño de la población, determinado por el valor de la variable de *Parents*.

Generations. Determina el número de iteraciones que realizará el sistema en la búsqueda del cromosoma óptimo. Durante el paso entre las generaciones se seleccionan los mejores cromosomas entre padres e hijos, los cuales se utilizarán en las generaciones subsecuentes.

En la línea 1 simplemente se inicia una variable (g) que llevará el conteo de la generaciones creadas en el sistema.

En la línea 2 se toman otros parámetros enviados a la función, como son: los Operadores de Mutación (*MutParam, Copy, MutTrans, DelTrans, AddTrans, AddState, DelState*) enviados en una lista. También contiene los siguientes valores: *Min, Max, pm, sSizeCov, sSizeMean, stepSize, EvaluationFunction* y un valor *Epsilon*. Estos valores son

pasados al sistema a través de la interfaz de interacción con el usuario. En la Figura 4.1 se puede observar la forma de declarar y enviar los valores de los parámetros al sistema.

A continuación se describe el significado y uso de estos parámetros utilizados por el sistema.

ProbMutParam. Determina la probabilidad de mutación sobre los valores de la matriz de medias y sobre la matriz de covarianza de cada estado de los HMMs.

ProbCopy. Este valor establece la probabilidad de copiar un cromosoma tomado de la población.

ProbMutTrans. Establece la probabilidad de cambiar valores de la matriz de transiciones entre estados de un HMM.

ProbDelTrans. Fija la probabilidad de eliminar transiciones existentes entre los estados de un cromosoma.

ProbAddTrans. Establece la probabilidad de poder agregar transiciones entre estados existentes de un cromosoma.

ProbAddState. Determina la probabilidad de agregar nodos o estados a los cromosomas.

ProbDelState. Establece la probabilidad de poder eliminar nodos o estados de los cromosomas.

Esta lista de operadores evolutivos, así como su valor probable de participación, se puede observar en la Figura 4.1, en la variable denominada *Operators*.

En la Figura 4.1 también se observan otros parámetros como el valor de *epsilon*, *Parents*, *Children*, y otros valores que ya han sido mencionados en el documento.

Los siguientes parámetros y significado usados en el **Algoritmo 1** Evolution, son:

MinStates. Establece el número mínimo de nodos o estados (*Min*), que deben contener los cromosomas.

MaxStates. Este parámetro determina el número máximo de estados (*Max*) que puede contener un cromosoma. En la Figura 4.1 también se pueden observar los valores definidos para *Min* y *Max*.

```
Epsilon = 0.7;
Parents = 300;
Children = 150;
Generations = 300;

GrafTime =
  Timing[Evolution[EPHMM[Parents, PLUS, Children, Generations,
     Operators → {{MutParameters, 0.1}, {Copy, 0.1}, {MutTransitions, 0.2},
       {DeleteTransition, 0.1}, {AddTransition, 0.2}, {AddState, 0.1},
       {DeleteState, 0.2}},
     Min → 2, Max → 16, pm → 0.2, sSizeMean → 0.8, sSizeCov → 0.8,
     stepSize → 0.5, EvaluationFunction → (ForwardHMM[#] &)]]
  ];
Print["Tiempo: ", GrafTime[[1]]];
Aptitud = GrafTime[[2, 1, Table[i, {i, Length[GrafTime[[2, 1]]]}], 1, 1, 1]];
ListLinePlot[Aptitud]
```

Figura 4.1: Definición de parámetros en el proceso evolutivo.

Algoritmo 1 Evolution($Parents, Children, Generations$)

1: $g \leftarrow 0$
2: $Parameters_{start} \leftarrow \{EvalFunction, Operators\}$
3: $MeanOperators \leftarrow Parameters_{start}$
4: $Pop_{initial} \leftarrow GenInitPop(Parents, minStates, maxStates)$
5: $Pop_{evaluated} \leftarrow EvalFunction(Pop_{initial})$
6: **while** $g < Generations$ **do**
7: $\quad Children \leftarrow BEST(Pop_{evaluated}) + MeanOperators(Pop_{evaluated})$
8: $\quad Pop_{evaluated} \leftarrow EvalFunction(Children)$
9: $\quad g \leftarrow g + 1$
10: **end while**
11: **return** $BEST(Pop_{evaluated})$

pm. Este valor está asociado con el parámetro *ProbMutParam*, *pm* es el valor de probabilidad de mutar, tanto valores del vector de medias como de la matriz de covarianzas, de determinados estados de un cromosoma.

sSizeMean. Es el valor del incremento en los valores de la matriz de medias.

sSizeCov. Es el valor del incremento, step size, en los valores de la matriz de varianza.

stepSize. Es el valor que se usará como incremento en la mutación de las transiciones entre estados de los cromosomas.

EvaluationFunction. Determina el nombre de la función encargada de evaluar a los cromosomas dentro de la población dada, y determinar su valor de aptitud. Para este caso, nuestra función de evaluación se llama *ForwardHMM*.

Epsilon. Se maneja este valor como intervalo para poder calcular el valor de probabilidad de los valores de la serie observada, mediante una función de densidad de probabilidad.

Estos últimos valores mencionados, también pueden observarse en la Figura 4.1 como parte de la interfaz del usuario.

Continuando con las instrucciones del **Algoritmo 1**, tenemos en la línea 3 a la variable *MeanOperators* que almacena los nombres de los operadores de mutación que intervendrán en la evolución de los cromosomas, así como el valor del porcentaje de participación dado a cada operador de mutación proporcionado por el usuario.

En la instrucción 4 se almacena en la variable $Pop_{initial}$ la población de cromosomas creados inicialmente de manera aleatoria. Esta creación de la población inicial se realiza a través de la llamada a la función *GenInitPop*, la cual se describirá más adelante.

En la línea 5, se evalúa toda la población inicial, con la llamada a nuestra función de evaluación de aptitud (función Forward, que se describirá en páginas siguientes), esta población calificada se almacena en la variable $Pop_{evaluated}$.

En la línea 7 del algoritmo, dentro del ciclo **while**, se generan los hijos o descendientes de la población (*Children*). Este vector se encontrará formado por los mejores cromosomas padres más los cromosomas mutados por los operadores de mutación, dados al sistema.

En la línea 8, la nueva población *Children*, que representa a los mejores cromosomas padres y cromosomas mutados por los operadores evolutivos, se evalúa con la función de aptitud, para calificar a todos los cromosomas.

Estas dos últimas instrucciones se realizan hasta llegar al término de las iteraciones, que se encuentra marcado por el valor de *Generations*. Al final se regresa la última población evaluada, con el cromosoma mejor calificado al inicio de la lista.

4.2.2. Algoritmo GenInitPop

En el **Algoritmo 2** GenInitPop se genera la población inicial; GenInitPop recibe como parámetros: *Parents, Min* y *Max*. Donde *Parents* especifica el número de cromosomas a crear o tamaño de la población; los parámetros *Min* y *Max*, indica el número mínimo y máximo de estados que podrían tener los cromosomas al momento de crearse.

Algoritmo 2 GenInitPop($Parents, Min, Max$)

```
1: for i = 1 to Parents do
2:     Pop[i] ← InitChrome(Min, Max, timeSeries)
3: end for
4: return Pop
```

En la línea número 2, después de la instrucción **for** del algoritmo, se establece la llamada a la función *InitChrome*, que es la encargada de crear cada cromosomas. La línea 4 del código del **Algoritmo 2**, regresa la lista (*Pop*) de cromosomas creados.

4.2.3. Algoritmo InitChrome

El **Algoritmo 3** InitChrome realiza la creación de un cromosoma. Inicia con los parámetros *Min* y *Max*, los que indican el rango de valores, entre el mínimo y máximo, que podría tomar un cromosoma al momento de generar sus estados. El parámetro *timeSeries* contiene el conjunto de las series de tiempo con las cuales se creará y entrenará cada HMM.

En la línea 1 se selecciona aleatoriamente un número entre *Min* y *Max*, que se almacena en la variable *nStates*, la cual se encargará de indicar el número de estados que contendrá el HMM.

En la línea 2 se hace una llamada a la función *FindClusters* de Mathematica, que nos proporciona un conjunto o número de clusters de acuerdo a los datos más similares. Se le pasa como parámetro la serie de tiempo y el número de clusters que se desean generar; el número de cluster representa el número de estados presentes en el cromosoma. La función nos regresa una lista con los conjuntos de datos similares. Esta función se utiliza para obtener clusters de los datos similares dados en la serie de tiempo, y una vez agrupados en clusters, se determina la media (μ) y la matriz de varianzas (Σ) para cada estado.

En la línea 3, es donde prácticamente se inicia la creación del HMM. Sus valores se generan a partir de la línea 4, en esta línea se visualiza el parámetro llamado *qualifica-*

Algoritmo 3 InitChrome($Min, Max, timeSeries$)

```
1: nStates ← RandomInteger(Min, Max)
2: nClusters ← FindClusters(timeSeries, nStates)
3: HMM ← [
4:       qualification[unDef],
5:       size ← nStates,
6:       transitions ← List(Transitions(nStates)),
7:       means ← FindMeanCluster(nClusters),
8:       covariance ← FindCovariance(nClusters),
9:       pi ← List(1/nStates)
10: ]
11: return HMM
```

tion[unDef], que es donde se almacenará su valor de aptitud, una vez que el cromosoma sea evaluado por la función correspondiente. Por el momento, en el instante de su creación, este valor se encuentra sin definir.

En la línea 5 se asigna al parámetro *size*, el número de estados que contiene el HMM.

En la línea 6 se generan de manera aleatoria las transiciones entre los (*nStates*) estados, con la función *Transitions*. Manteniendo la propiedad de que la suma total de probabilidad de las transiciones de salida de cada estado es *1*, en toda fila de la matriz de transiciones entre estados.

En la línea 7 se calcula la Matriz de Medias, mediante la función *FindMeanCluster*, la cual recibe la lista de clusters de valores en *nClusters* y determina la media para cada estado del cromosoma.

La línea 8 es parecida a la anterior, sólo que aquí se calcula la Matriz de Varianzas con la función *FindCovariance*.

En la línea 9 se establece de manera proporcional la probabilidad de iniciar un estado en un momento de $t = 1$, al comienzo de la ejecución del HMM.

En la línea 10 se puede ver el cromosoma o HMM que regresa el algoritmo, después de ser creado y configurado.

4.2.4. Algoritmo Initialization

En las líneas del **Algoritmo 4** Initialization, se describe el cálculo de la probabilidad del primer vector de datos de la serie de tiempo, dado el modelo HMM. A esto se le conoce como el *método de inicialización* del algoritmo Alfa (*Forward*) [Rabiner89]. El algoritmo retorna la lista de probabilidad de la primera observación del vector de datos, de iniciar en todos los estados del HMM, en el tiempo $t = 1$.

Algoritmo 4 Initialization($HMM, Data, Epsilon$)

1: $alpha \leftarrow \{\}$
2: $nStates \leftarrow getSizeHmm(HMM)$
3: **for** $i = 1$ to $nStates$ **do**
4: $\quad probab \leftarrow FuncProbMult(getMeansState(HMM, i),$
$\qquad\qquad getCovarianceState(HMM, i), Data, Epsilon)$
5: $\quad alpha[i] \leftarrow getPiState(HMM, i) * probab$
6: **end for**
7: **return** $alpha$

Las instrucciones más importante de este algoritmo se encuentra en las líneas 4 y 5. En la 4 se llama a la función *FuncProbMult*, que se encarga de calcular la probabilidad de un vector de datos, dado un estado S_i del HMM en particular.

La función toma como parámetros la matriz de medias, de la cual toma únicamente el vector de medias del estado S_i, y la matriz de covarianzas del estado S_i, el vector de datos, y el valor de *Epsilon* establecido.

En la línea 5 se calcula la probabilidad de que inicie un estado S_i, multiplicado por la probabilidad de que el estado S_i observe el vector de datos en el tiempo $t = 1$. Es decir, se determina la probabilidad que al iniciar el sistema se inicie en un estado y se observe el vector de datos en ese instante.

4.2.5. Algoritmo Induction

En el **Algoritmo 5** Induction, que se utiliza a continuación del algoritmo Initialization (ver **Algoritmo 4**), se calcula la probabilidad de los vectores de datos restantes de la serie de tiempo. Este algoritmo calcula la probabilidad de cada uno de los datos, a

partir del segundo dato en la serie, contra un modelo HMM. Este proceso, denominado *inducción*, necesita el vector de probabilidades *alpha*, generado con el algoritmo Initialization que procesó el primer dato de la secuencia, para poder calcular las probabilidades de los datos restantes en la secuencia de observación.

Al **Algoritmo 5** se le conoce como el *método de inducción* del algoritmo Alfa (*Forward*). El algoritmo regresa la lista o vector de probabilidad de la serie de tiempo, verificada contra todos los estados del HMM en un tiempo de *t + 1*.

El algoritmo necesita como parámetros de entrada: el *HMM*, el vector de datos (*Data*), el vector de probabilidad inicial (*alpha*), y el valor del *Epsilon*.

En la línea 3 del algoritmo, se toman todas las transiciones existentes entre los estados del modelo, y se asignan a la variable $matrixA_{i,j}$.

En la línea 5 se calcula la probabilidad del vector de datos, de ser observado en un estado S_j en un tiempo *t + 1*.

Algoritmo 5 Induction($HMM, Data, alpha, Epsilon$)

1: $ListAlpha \leftarrow \{\}$
2: $nStates \leftarrow getSizeHmm(HMM)$
3: $matrixA_{i,j} \leftarrow getAllTransitions(HMM)$
4: **for** $j = 1$ to $nStates$ **do**
5: $probab \leftarrow FuncProbMult(getMeansState(HMM, j),$
$getCovarianceState(HMM, j), Data, Epsilon)$
6: $sumAlpha \leftarrow alpha[j] * matrixA_{i,j}[All, j]$
7: $ListAlpha[j] \leftarrow sumAlfa * probab$
8: **end for**
9: **return** $ListAlpha$

En la línea número 6 se calcula la probabilidad de haber iniciado en el estado S_i en el tiempo t (representado por *alpha*), por la probabilidad de transitar del estado S_i al estado S_j en el tiempo *t + 1* (almacenado en la variable $matrixA_{i,j}$).

En la línea 7 se almacena en la variable *ListAlpha* la probabilidad total de haber partido del estado S_i, en un tiempo t, de haber transitado del estado S_i al estado S_j, y de observar el vector de datos en el estado S_j, en un tiempo *t + 1*.

La línea 9 muestra el regreso de la lista o vector *ListAlpha* con la probabilidad del

dato $t + 1$, siguiente dato en la serie de tiempo, dado por el HMM.

4.2.6. Algoritmo Forward

El **Algoritmo 6** Forward calcula la suma de la probabilidad total de la serie contra el HMM en cuestión. Este es el algoritmo que empleamos como *Función Objetivo*, con el cual evaluamos la aptitud de cada HMM contra la serie de tiempo dada o lista de vector de datos.

Después de evaluados todos los HMMs se ordenan y seleccionan los HMMs mejor calificados.

El algoritmo retorna el valor de probabilidad total de la serie de tiempo generada con el HMM, en la marca de tiempo $t = T$, que representa el último dato de la serie.

Algoritmo 6 Forward($HMM, timeSeries, Epsilon$)

1: $alpha \leftarrow \{\}$
2: $len \leftarrow Length(timeSeries) - 1$
3: $alpha[1] \leftarrow Initialization(HMM, timeSeries[1], Epsilon)$
4: **for** $t = 1$ to len **do**
5: $\quad alpha[t+1] \leftarrow Induction(HMM, timeSeries[t+1], alpha[t], Epsilon)$
6: **end for**
7: **return** $ProbabTotal[alpha[len+1]]$

En la línea 3 del algoritmo se inicia el proceso de cálculo de la probabilidad, con la función *Initialization*, tomando el modelo HMM, el primer vector de datos de la serie de tiempo y el valor de epsilon.

En la instrucción, de la línea 5, se envían los vectores de datos subsecuentes a la función *Induction*, para el cálculo de las probabilidades restantes hasta $t = T$, donde T es la longitud total de nuestra serie de tiempo o longitud de la secuencia de datos dada.

Como *alpha* es una matriz, la suma de la probabilidad total del último vector contendrá la probabilidad total de la serie de tiempo, generada con el HMM. Por eso se retorna ese valor que constituye la calificación final de aptitud al HMM.

4.2.7. Algoritmo FuncProbMult

En el **Algoritmo 7** FuncProbMult se procesan los parámetros recibidos: el vector de medias *vMean*, la matriz de varianzas *matCovar*, el vector de datos (*Data*) con los valores de las variables, y el valor de *Epsilon* ϵ.

El algoritmo se encarga de calcular y retornar la probabilidad del vector de datos, utilizando para esto la función que integra el resultado de la función de densidad de probabilidad, que en este caso es la *función de distribución normal multivariable*. La *función de distribución normal multivariable* utiliza el vector de medias y la matriz de varianzas del estado S_i, y por último, el valor de *Epsilon* ϵ. Es decir, que la función calcula la probabilidad de observar el vector de datos en el estado S_i en una marca de tiempo *t*. El valor de *Epsilon* ϵ determina el intervalo a considerar para determinar la probabilidad del vector de datos dado, en el estado S_i.

Algoritmo 7 FuncProbMult($vMean, matCovar, Data, \epsilon$)

1: $probab \leftarrow$
$$\int_{X_D^*-\epsilon}^{X_D^*+\epsilon} \dots \int_{X_2^*-\epsilon}^{X_2^*+\epsilon} \int_{X_1^*-\epsilon}^{X_1^*+\epsilon} f(\mu, \Sigma, O) dx_1 dx_2, \dots, dx_D$$

2: **return** *probab*

Se conoce que la probabilidad asociada a un valor específico para una variable aleatoria continua x_0, es cero. Es por esto, que se necesita especificar un intervalo (x_0 - ϵ, $x_0 + \epsilon$) para poder calcular la probabilidad de que la variable aleatoria continua pueda caer en ese intervalo.

Ahora bien, si se tiene una observación D-dimensional de las variables

$$\mathbf{X} = \{X_1, X_2, \dots, X_D\}$$

y se desea calcular la probabilidad de que $\mathbf{X}$ está en la cercanía de

$$\{X_1^*, X_2^*, \dots, X_D^*\}$$

se necesita integrar D veces como se muestra en la ecuación:

$$\int_{X_D^*-Ep}^{X_D^*+Ep} \dots \int_{X_2^*-Ep}^{X_2^*+Ep} \int_{X_1^*-Ep}^{X_1^*+Ep} f(\mu, \Sigma, O) dx_1 dx_2, \dots, dx_D$$

donde $f(\mu, \Sigma, O)$ es la función de multidistribución gaussiana.

4.2.8. Algoritmo AddTransition

El **Algoritmo 8** AddTransition, es el operador de mutación que se encarga de agregar una nueva transición a un estado. Recibe los siguiente valores como parámetros de entrada: el *HMM* a mutar, el número de estado *nState* a modificar seleccionado de manera aleatoria, y el número de espacios o transiciones *availabTrans*, donde puede insertarse un nuevo valor de transición.

En la línea 1 del algoritmo se obtiene la lista de las transiciones existentes del estado.

En la línea 2, se determina el valor del incremento *Increm* para operar con las transiciones existentes, considerando únicamente las transiciones que tienen un valor distinto de cero, más el incremento en una unidad, que corresponde a la nueva transición.

En la línea 4 determinamos de manera aleatoria, en caso de tener más de un espacio disponible, el número de espacio o estado donde se va a insertar la nueva transición.

Algoritmo 8 AddTransition($HMM, nState, availabTrans$)

1: $transState \leftarrow getTransState(HMM, nState)$
2: $Increm \leftarrow 1/(transState \neq 0) + 1$
3: **if** $availabTrans > 1$ **then**
4: $availabTrans \leftarrow RandomInteger(Length(availabTrans))$
5: **end if**
6: **for** $i = 1$ to $Length(transState)$ **do**
7: **if** $transState[i] \neq 0$ **then**
8: $transState[i] \leftarrow transState[i] - (transState[i] * Increm/(1 - Increm))$
9: **end if**
10: **end for**
11: $transState[availabTrans] \leftarrow Increm + (Increm * Increm/(1 - Increm))$
12: **return** transState

En la línea 8, dentro del ciclo **for**, se realiza una operación aritmética para determinar el nuevo valor que contendrán las transiciones del estado, tomando en cuenta su valor actual más el valor del incremento. La forma de mantener de manera correcta la distribución de la unidad entre las transiciones es de la manera siguiente: al número de transiciones que tiene actualmente el estado, se le agrega una más, y se divide la unidad entre este

número ($inc = [1/(\text{numTrans} + 1)]$). *inc* es el incremento que se agregará a las transiciones actuales de la forma $Trans_i = Trans_i - [Trans_i * inc/(1 - inc)]$; *con* $i < N$ porque existen transiciones con valor de 0 en el estado.

En la línea 11 se inserta el nuevo valor de la transición, dependiendo del valor proporcional que le corresponda generado de la operación aritmética $Trans_{new} = inc + [inc * inc/(1 - inc)]$.

Y por último, se retorna la lista de transiciones actualizada para ese estado. Que de manera posterior se incorparará al HMM mutado.

4.2.9. Algoritmo DeleteTransition

En el **Algoritmo 9** DeleteTransition, describe al operador de mutación que elimina una transición de un estado. Este algoritmo recibe como único parámetro de entrada el *HMM* sobre el cual va a operar para mutar.

En la línea 1 se selecciona aleatoriamente un número de estado al cual se le eliminará la transición, el número de estado se almacena en la variable *delTrans*.

En la instrucción 2 del algoritmo se guarda la matriz de transiciones del HMM en la variable *allTrans*.

En la siguiente instrucción se elige aleatoriamente el número preciso de la transición a eliminar *delete*, del estado seleccionado previamente.

En la línea 4 se actualiza con un cero, lo que significa que se elimina la transición, existente en el estado seleccionado.

En la instrucción 5 se normalizan los valores restantes del vector de transiciones, de tal forma que sumen la unidad. Esto es, de manera aleatoria se distribuyen nuevos valores a las transiciones activas, manteniendo el valor de 1 en la distribución.

En la línea 6 se actualiza la matriz de transiciones del modelo HMM, y finalmente se retorna el HMM con los valores mutados.

4.2.10. Algoritmo MutParameters

El **Algoritmo 10** MutParameters, se encarga de operar sobre los valores del vector de medias y los de la matriz de covarianzas. Incrementando o decrementado los valores guardados en el vector de medias y en la matriz de covarianzas de manera aleatoria, de acuerdo a un valor de regresado por una función de distribución normal. El algoritmo

Algoritmo 9 DeleteTransition(HMM)

1: $delTrans \leftarrow RandomInteger(getSizeHmm(HMM))$
2: $allTrans \leftarrow getAllTrans(HMM)$
3: $delete \leftarrow RandomInteger(getSizeHmm(HMM))$
4: $allTrans(delTrans, delete) \leftarrow 0$
5: $allTrans[delTrans] \leftarrow NormalizeTransition(allTrans(delTrans))$
6: $HMM \leftarrow allTrans$
7: **return** HMM

recibe como único parámetro el *HMM* a mutar.

Como se podrá observar dentro del primer ciclo **for**, en las líneas 2 y 3 se obtienen el vector de medias y la matriz de varianzas del estado en cuestión del HMM.

En la instrucción de la línea 4 se genera un valor de tipo real con la función *RandomReal*, el cual se compara contra el parámetro *pm* (*probabilidad de mutación*), establecido y descrito previamente. La comparación determina si se procede o no a modificar los valores del vector de medias y la matriz de covarianzas del estado.

Algoritmo 10 MutParameters(HMM)

1: **for** $j = 1$ to $getSizeHmm(HMM)$ **do**
2: $vMean \leftarrow HMM$
3: $matCovar \leftarrow HMM$
4: **if** $RandomReal() < pm$ **then**
5: **for** $k = 1$ to $numVars$ **do**
6: $vMean[j,k] \leftarrow vMean[j,k] + NormalDistribution(sSizeMean)$
7: $matCovar[k,k] \leftarrow matCovar[k,k] + NormalDistribution(sSizeCov)$
8: **end for**
9: **end if**
10: **end for**
11: $HMM \leftarrow vMean$
12: $HMM \leftarrow matCovar$
13: **return** HMM

En las líneas 6 y 7, se procede a modificar los valores del vector de medias y la matriz de varianzas, al sumarle los valores generados de manera aleatoria por una distribu-

ción normal. A la función de distribución normal se le pasan los parámetros *sSizeMean* y *sSizeCov*, como se pueden ver en las líneas 6 y 7. Los valores de *sSizeMean* y *sSizeCov* fueron establecidos por el usuario al iniciar el sistema evolutivo y representan pequeños desplazamientos del valor de las medias y de la matriz de varianzas. Los incrementos en los valores de desplazamiento se realizan de acuerdo a los fijado previamente como parámetros de inicio del sistema.

Una vez realizada esta operación iterativa para cada estado del HMM, en las instrucciones de las líneas 11 y 12 se transfiere el vector de medias y matriz de varianza al HMM. En la línea 13, el algoritmo regresa el HMM con sus valores mutados.

4.2.11. Algoritmo MutTransitions

El **Algoritmo 11** MutTransitions, se encarga de llevar a cabo la mutación de los valores de las transiciones de un estado en particular. La forma de operar es como sigue: en la línea 1 se determina el número de estados que tiene el HMM.

Algoritmo 11 MutTransitions(HMM)

1: $nStates \leftarrow getSizeHmm(HMM)$
2: $mutState \leftarrow RandomInteger(nStates)$
3: $transState \leftarrow getTransState(HMM, mutState)$
4: **for** $i = 1$ to $nStates$ **do**
5: $\quad transState[i] \leftarrow transState[i] + NormalDistribution(stepSize)$
6: **end for**
7: $HMM \leftarrow NormalizeTransition(transState)$
8: **return** HMM

En la línea 2 se selecciona de manera aleatoria el número de estado sobre el cual operar (*mutState*). En la siguiente instrucción se obtienen las transiciones de ese estado, y se almacenan en el vector *transState*.

En la línea 5, dentro del ciclo iterativo *for*, se modifican todos los valores de las transiciones que tiene el estado, almacenado en el vector *transState*. El valor que se asigna es el que se crea de manera aleatoria con una función de distribución normal (*NormalDistribution*). La variable *stepSize* es un parámetro definido y descrito de manera previa, que es un valor declarado por el usuario al inicio del programa como parámetro.

En la línea 7, una vez modificados los valores de las transiciones del estado, se normalizan los valores, de tal manera que la suma del vector de transiciones sea igual a la unidad. En la última línea se regresa el HMM modificado o evolucionado aleatoriamente.

4.2.12. Algoritmo DeleteState

El **Algoritmo 12** DeleteState realiza la operación de mutación que se encarga de eliminar un estado de un HMM dado.

Algoritmo 12 DeleteState(HMM)

1: $nStates \leftarrow getSizeHmm(HMM)$
2: $randState \leftarrow RandomInteger(nStates)$
3: $nStates \leftarrow nStates - 1$
4: $allTrans \leftarrow Delete(getAllTrans(HMM), randState)$
5: **for** $i = 1$ to $nStates$ **do**
6: $\quad allTrans[i] \leftarrow Delete(allTrans[i], randState)$
7: **end for**
8: $HMM \leftarrow NormalizeTransition(allTrans)$
9: $HMM \leftarrow nStates$
10: $nClusters \leftarrow FindCluster(timeSeries, nStates)$
11: $HMM \leftarrow FindMeanCluster(nClusters)$
12: $HMM \leftarrow FindCovariance(nClusters)$
13: $HMM \leftarrow NormalizeTransition(Delete(getPiHmm(HMM), randState))$
14: **return** HMM

En las líneas 1 y 2, se obtiene el número de estados con que cuenta el HMM y se selecciona aleatoriamente un número de estado a eliminar.

En la instrucción número 4 se procede a eliminar la lista o vector de transiciones del estado seleccionado (*randState*), de la matriz de transiciones del modelo HMM representado por *getAllTrans(HMM)*. Y con esta instrucción se elimina el vector de transiciones del estado de la matriz.

En la instrucción 6, dentro del ciclo iterativo del **for**, se procede a eliminar toda transición existente hacia el estado eliminado, de todos los vectores de los estados restantes que aún se encuentran en la matriz de transiciones.

En la línea 8 se procede a normalizar el valor de todas las transiciones de los estados restantes. Es decir, que la suma de los valores de cada vector de transiciones sea igual a 1.

En la línea 9 se actualiza el número de estados que ahora tiene el HMM. Y en la línea 10 se determina el número de clusters en base a la serie de tiempo y el nuevo número de estados del HMM.

En las líneas 11 y 12 del algoritmo se determina el nuevo vector de medias y las nuevas matrices de varianza para cada estado del HMM.

En la línea 13 se procede a eliminar el valor de Pi del estado borrado, y se normalizan los valores de Pi de los estados restantes. Por último se regresa el HMM modificado.

4.2.13. Algoritmo AddState

En el **Algoritmo 13** AddState, se realiza la operación de mutación que consiste en agregar un nuevo estado a un cromosoma o HMM.

Algoritmo 13 AddState(HMM)

1: $nStates \leftarrow getSizeHmm(HMM) + 1$
2: $nClusters \leftarrow FindClusters(timeSeries, nStates)$
3: $HMM \leftarrow nStates$
4: $HMM \leftarrow newTransState(getAllTrans(HMM))$
5: $HMM \leftarrow FindMeanCluster(nClusters)$
6: $HMM \leftarrow FindCovariance(nClusters)$
7: $HMM \leftarrow Pi(1/nStates)$
8: **return** HMM

En la línea 1 se obtiene el número de estados actual, y se incrementa el valor en uno. Lo cual indica que el HMM aumentará su tamaño en un estado adicional.

Con esta información, se determina el número de clusters en la línea 2. Tomando como datos principales la serie de tiempo (*timeSeries*) y el nuevo número de estados (*nStates*).

En la instrucción 3, se actualiza el nuevo número de estados del HMM. Y en la línea 4 se realiza la creación del vector de transiciones, que contendrá el nuevo estado creado, hacia los estados existentes. Estas conexiones son creadas de manera aleatoria, y sus valores son normalizados. En los estados existentes, no se establece la conexión hacia el nuevo

estado de manera inmediata, sino que se deja que el mismo sistema siga evolucionando, y que en una generación futura, sea probable que se se aplique el operador de agregar una nueva transición a ese estado.

En las líneas 5 y 6 se vuelve a determinar el vector de medias y la matriz de varianzas para todos los estados, debido a la inserción del nuevo estado.

En la línea 7 se recalcula la probabilidad de iniciar algún estado en un tiempo de $t = 1$. Es decir, la probabilidad de que el sistema tome al iniciar, cualquier estado como punto de arranque. Y por último se regresa el *HMM* mutado.

4.3. HMM usado en la detección de anomalías.

Al efectuar la inicialización de la población de cromosomas y de comenzar el ciclo evolutivo con éstos, aplicando los operadores evolutivos, se obtienen cromosomas mutados que son una forma codificada de posible solución al problema. Las generaciones y operaciones evolutivas, sobre un subconjunto de individuos de la población, se repiten hasta que se alcanza el número máximo de generaciones.

Una vez que el sistema evolutivo ha finalizado con las generaciones y ha evolucionado a los HMMs, se toma el mejor modelo de la última generación y se usa para determinar si una secuencia de observación dada presenta o no anomalías. Con el modelo estadístico evolucionado se pueden examinar nuevas secuencias de observaciones y determinar si una serie de tiempo de datos pertenece al modelo. Esto es, si la secuencia de observación tiene probabilidad de haber sido generada por el modelo. Si la secuencia de observación tiene una baja probabilidad de pertenecer al modelo, entonces se considera como una secuencia con anomalías.

En la Figura 4.2 se presenta un HMM evolucionado con el sistema. Este modelo fue creado de manera automática basado únicamente en la secuencia de observación dada por la lectura del consumo del ancho de banda de la red del posgrado de Ingeniería Eléctrica de la UMSNH.

En la siguiente sección se describirán los algoritmos que se utilizaron para efectuar el análisis de los datos de secuencias de observación y proporcionar información que ayude a determinar, con ayuda del modelo, la probabilidad de ser una secuencia normal o anómala.

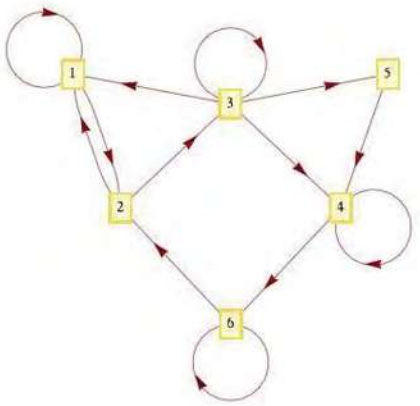

Figura 4.2: Ejemplo de un HMM evolucionado con EP.

4.4. Algoritmos empleados

En esta sección se describirán los algoritmos que se emplearon para poder determinar la probabilidad de que una serie de tiempo, posea o no anomalías. Estos algoritmos toman como base para el análisis de las secuencias de datos, el modelo HMM evolucionado y un valor de umbral.

El valor del umbral (*Threshold*) define el valor de probabilidad mínima que deben generar las secuencias de datos analizadas (Ecuacion 4.2). La especificación del valor del umbral está dada por el análisis del comportamiento o probabilidad de las ventanas del mismo tamaño, generadas con los datos de entrenamiento, $Threshold = P_t^w(O \mid \lambda)$. Por lo tanto, las probabilidades iguales o mayores a este umbral, generadas con las secuencias de pruebas, serán consideradas normales, y anómalas en caso contrario.

$$P_t^w(A \mid \lambda) \geq Threshold \tag{4.2}$$

4.4.1. Algoritmo iniProcChkWin

En el **Algoritmo 14** iniProcChkWin se inicia el proceso de verificación de las probabilidades de las ventanas de datos. La función recibe como parámetros el modelo *HMM* evolucionado, el tamaño inicial *winSizeIni* y el tamaño final de la ventana *winSizeEnd* a

analizar. La serie de tiempo *timeSerie* y el valor de *Epsilon*.

Los tamaños de ventana que el sistema analizó en los casos de prueba, fueron de 3, 4, 5 y 6 datos por ventana de tiempo. Esta función hace un llamado a la función *WindowChk*, que se encarga de efectuar el análisis de los datos en el tamaño de ventana de datos correspondiente.

Algoritmo 14 iniProcChkWin($HMM, winSizeIni, winSizeEnd, timeSerie, Epsilon$)

1: **for** $winSize = winSizeIni$ to $winSizeEnd$ **do**
2: $WindowChk(HMM, timeSerie, winSize, Epsilon)$
3: **end for**

4.4.2. Algoritmo WindowChk

El **Algoritmo 15** se encarga de calcular la probabilidad de la subsecuencia o ventana de datos, en toda la serie de tiempo dada para la prueba. La ventana se desplaza cada marca de tiempo, en una unidad y realiza el cálculo de probabilidad para esa ventana en esa marca de tiempo.

Algoritmo 15 WindowChk($HMM, timeSerie, winSize, Epsilon$)

1: $nStates \leftarrow getSizeHmm(HMM)$
2: $window \leftarrow Take(timeSerie, winSize)$
3: $alpha \leftarrow Initialization(HMM, window[1], Epsilon))$
4: $timeSerie \leftarrow Take(timeSerie - winSize)$
5: **while** $Length(timeSerie) > 0$ **do**
6: **for** $i = 1$ to $nStates$ **do**
7: $probAlpha[i] \leftarrow ChkTimeSerie(HMM, winSize, i, alpha, window, Epsilon)$
8: **end for**
9: $Prob \leftarrow SumTotal(probAlpha)$
10: **print** $(WindowProbability : Prob)$
11: $window \leftarrow Delete(window, 1)$
12: $window \leftarrow Append(window, First(timeSerie))$
13: $timeSerie \leftarrow Drop(timeSerie, 1)$
14: $alpha \leftarrow Initialization(HMM, window[1], Epsilon))$
15: **end while**

En la línea 1 del algoritmo se obtiene el número de estados que contiene el HMM. En la línea 2 se toma el primer subconjunto de datos, de tamaño especificado para la ventana *winSize*, de la serie de tiempo de prueba *timeSerie*.

La línea 3 especifica la llamada al algoritmo Initialization, para generar la probabilidad del primer dato de la ventana y almacenarlo en el vector *alpha*.

En la línea 4 se elimina de la serie de tiempo de prueba *timeSerie*, el número de datos tomados por la ventana de tiempo.

En la línea 5 se comienza el ciclo de recorrido de la ventana deslizante, eliminando los datos analizados de la serie de tiempo de prueba.

En la línea 7 dentro del ciclo iterativo **for** se llama al algoritmo *ChkTimeSerie*, con el cual se calcula la probabilidad de observar la secuencia de los datos de la ventana, en los estados del modelo. Es decir, se calcula la probabilidad que la secuencia de datos, que contiene la ventana de tiempo, haya sido generada por el modelo.

En la línea 9 se obtiene la probabilidad total que acumuló el vector *probAlpha* de la ventana de datos. Este valor bien se puede imprimir o guardar en archivo.

En la línea 11 se elimina el primer dato de la ventana, y en la línea 12 se agrega al final de la ventana el primer dato que se encuentra en la serie de tiempo de prueba. La ventana hace el recorrido hacia el siguiente dato, y así continuará hasta llegar al último dato de la serie de tiempo de prueba.

En la línea 13 se elimina el primer elemento de la serie de tiempo de los datos de prueba, que fue el dato que se agregó a la ventana de datos.

En la línea 14 se hace el llamado al algoritmo *Initialization*, para iniciar el cálculo de probabilidad *alpha* para el primer elemento de la ventana de datos, y regresa al ciclo **while**, para calcular la probabilidad total de esta nueva ventana. Así hasta terminar con la última ventana y con los datos de la serie de tiempo de prueba.

4.4.3. Algoritmo ChkTimeSerie

El **Algoritmo 16** ChkTimeSerie se encarga de calcular la probabilidad para el dato siguiente en la ventana de tiempo, tomando en cuenta la probabilidad de pasar a un estado *edoJ* (S_j), en un tiempo $t + 1$.

En la línea 3 se hace una llamada al algoritmo *Induction* que se encarga de calcular la probabilidad del dato siguiente ($t + 1$) de la ventana de datos, a partir del vector de

Algoritmo 16 ChkTimeSerie($HMM, winSize, edoJ, alpha, timeSerie, Epsilon$)

1: $time \leftarrow 1$
2: **while** $time \leq winSize$ **do**
3: $alpha \leftarrow Induction(HMM, alpha, timeSerie[t+1], Epsilon)$
4: $time \leftarrow time + 1$
5: **end while**
6: **return** *alpha*

probabilidad *alpha*. Este proceso se realiza hasta alcanzar el tamaño de la ventana *winSize*.

En la línea 6 se regresa el vector de probabilidad *alpha* de la secuencia de datos de la ventana de tiempo analizada.

En resumen podemos mencionar lo siguiente:

- El modelo HMM que se utiliza es el creado y entrenado con programación evolutiva, y la serie de tiempo utilizada, es la serie de datos de prueba. Que son datos que genera el sistema modelado en su proceso de actividad cotidiana.
- El HMM es una herramienta de gran ayuda en la detección estadística de comportamiento anómalo en series de tiempo; su función es la de detectar probabilísticamente un comportamiento anómalo y, ayudar a discriminarlo de los normales.
- El uso que se les dio a los HMMs se enfoca en la técnica de la detección de anomalías basado en métodos probabilísticos estadísticos. Construimos un perfil o modelo de comportamiento normal de un sistema, basado en métodos estadísticos y empleamos pruebas probabilísticas para determinar si una actividad observada se desvía significativamente del modelo normal.

Capítulo 5

Resultados

En este capítulo se muestran las pruebas y resultados de los experimentos realizados con los HMMs Continuos Univariados (HMMCUs) y Multivariados (HMMCMs). Estos modelos fueron entrenados con Programación Evolutiva (EP, Evolutive Programming). Se incluye también un modelo HMM optimizado con el Algoritmo de Baum-Welch (HMM-BW). Todos estos modelos fueron empleados en la detección de anomalías en series de tiempo.

5.1. Introducción

Antes de comenzar con la presentación de la información, se describirán las características de la plataforma, sobre la cual se desarrollaron y se probaron todos los HMMs evolucionados. La plataforma se encuentra constituida por una *iMac*, con el sistema operativo OS *X* versión 10.5.8. Con un procesador de arquitectura Intel Core 2 Duo, a 2.16 Ghz, y con 1 Gb en memoria RAM; el software utilizado en el desarrollo de los algoritmos y en el cálculo de las distribuciones de probabilidad, fue Mathematica ver. 7.0.

Una forma de concebir a la *Programación Evolutiva* (EP) es en un nivel de abstracción mayor que los *Algoritmos Genéticos* (GAs). Aunque la EP se encuentra basada en AGs, trata en general, con sistemas de cómputo específicos y las funciones de sus operadores son particulares con el sistema en cuestión. Por ejemplo, tenemos a las máquinas de estados finitos, donde las funciones de sus operadores evolutivos consistirán en agregar-eliminar nodos, agregar-eliminar aristas, generar un individuo de tipo de máquina de estados, etc.

En la generación de los HMMs, el sistema hace evolucionar estos modelos usan-

do EP y un conjunto de operadores evolutivos particulares (como Agregar-Eliminar nodo, Agregar-Eliminar Transición, etc). Se probó la evolución de los modelos con diferentes números de generaciones y diferentes tamaños de población.

La Figura 5.1 muestra los resultados parciales de cómo la EP evoluciona un HMM en el sistema desarrollado en este trabajo de tesis. Cada subfigura representa un paso en el proceso de creación y optimización de los HMMs, presentando al mejor HMM en cada generación.

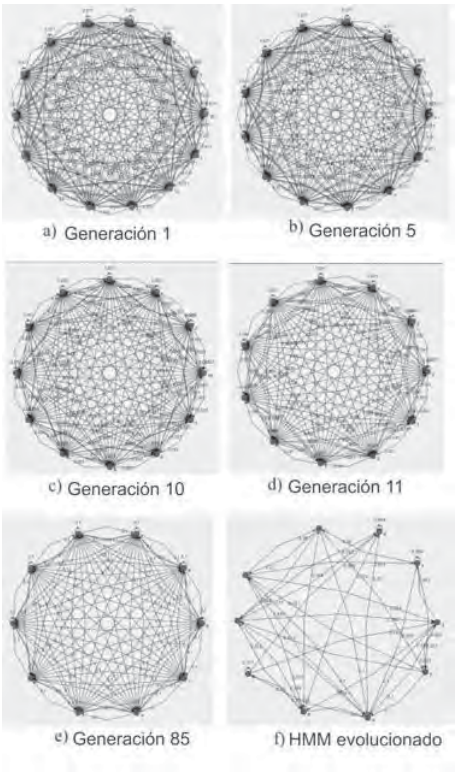

Figura 5.1: Evolución de un HMM con EP.

En la subfigura a) de la Figura 5.1, se presenta la generación 1 del proceso evolutivo, donde se observa un HMM formado por 14 estados en su proceso evolutivo. La subfigura b) muestra al HMM evolucionado con 13 estados hasta la generación 5. En la subfigura c) se observa al HMM con 12 estados en la generación 10. En la subfigura d) se muestra el HMM que ha evolucionado a sólo 11 estados en la generación 11. En la subfigura e) se presenta al HMM con 10 estados, evolucionado hasta la generación 85. En la última

subfigura, f) se presenta al HMM con 10 estados, y el proceso continua hasta llegar al número de generación especificado. En este experimento se estableció en 120 generaciones. En la última generación, la EP ha alcanzado un número óptimo de estados y se concentran en refinar las probabilidades de transición.

5.2. Secuencias de observación de prueba

En esta subsección se darán a conocer las secuencias de observación de prueba que se utilizaron en los experimentos de análisis de detección de anomalías. Cada modelo utilizó diferentes series de tiempo de diferente longitud. En algunas comparaciones, que se identificarán más adelante, se utiliza la misma secuencia de observación de prueba pero con diferentes modelos.

5.2.1. Secuencias de observación univariadas

En los experimentos con HMMCUs, HMM Continuos Univariados, se usó la serie de tiempo producida por la variable aleatoria continua, que representa el consumo de ancho de banda (Figura 5.2) de la subred por la División de Estudios de Posgrado de la Facultad en Ingeniería Eléctrica de nuestra universidad. La serie de observaciones fueron proporcionados por el centro de cómputo y representan el consumo de ancho de banda de red normal, de los días miércoles y jueves. Con esta serie de tiempo fueron creados y entrenados los HMMCUs evolucionados.

En los experimentos de detección de anomalías, a la serie de tiempo del consumo de ancho de banda de red normal, se le modificó un subconjunto de datos para simular variaciones en la dinámica del comportamiento del tráfico de la red (variación en el consumo del ancho de banda de red). Una variación en el consumo de ancho de banda representa un comportamiento anómalo, pudiendo ser generado por un ataque informático, barrido de las direcciones de red, denegación de servicio, etc.

A la secuencia de observación generada por el consumo de ancho de banda de red, la denominaremos (O_{eu}), secuencia de observación para el entrenamiento de HMMs univariados. Los HMMCUs se crearon y entrenaron usando esta serie de tiempo de 48 valores, que corresponde al consumo del ancho de banda en kbit/segundo monitoreada cada hora, en dos días consecutivos. Los valores de la serie fluctúan entre 14.625 y 5964.81 kbit/segundo.

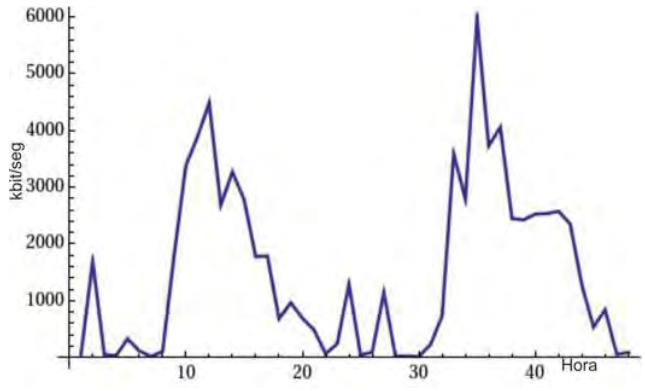

Figura 5.2: Consumo de ancho de banda en Kbit/seg en 48 hrs.

A la secuencia original de entrenamiento O_{eu}, se le insertaron datos sintéticos creados aleatoriamente para simular una anomalía en la secuencia de observación, y experimentar si el sistema detector de anomalías sería capaz de detectar esta situación anómala. A esta secuencia de prueba con datos sintéticos la denominamos O_{pu}, secuencia de observación de prueba univariada, para poder hacer referencia a esta serie de tiempo en los experimentos.

$$O_{pu} = [o_{pu1},\ o_{pu2}, \ldots,\ o_{pu48}]^T$$

La Figura 5.3 muestra las gráficas que representan a la secuencia de observación normal O_{eu} (*comportamiento normal*) y la secuencia de observación anómala O_{pu} (*comportamiento anómalo*).

5.2.2. Secuencias de observación multivariadas

La Tabla 5.1 muestra los campos de tipo de datos que son monitoreados en el centro de cómputo de la Universidad, con la herramienta llamada *PRTG Traffic Grapher v6.2.2.984*. El monitoreo se lleva a cabo de manera constante, sin interrupciones, observando todas las subredes de todos los campus, registrando los datos cada hora durante todo el año, así que la tabla puede llegar a tener miles de registros.

Tabla 5.1: Datos del monitoreo del consumo de ancho de banda de red.

Fecha	Tráfico IN		Tráfico OUT		Suma		Covertura
	kbyte	kbit/sec	kbyte	kbit/sec	kbyte	kbit/sec	%

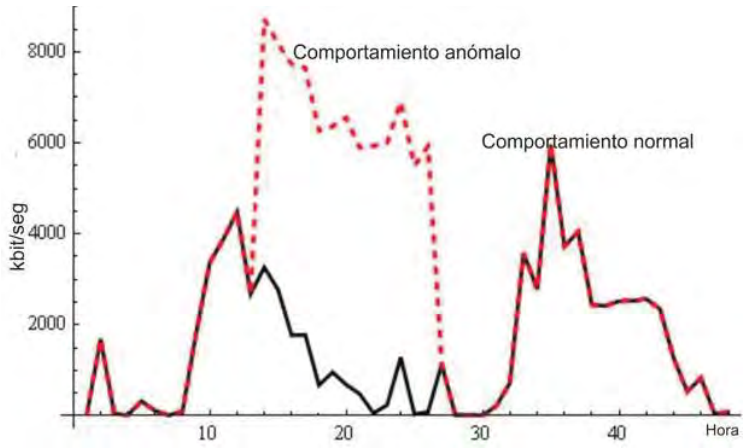

Figura 5.3: Datos de uso de ancho de banda con comportamiento normal y anormal.

El significado de Tráfico IN es el tráfico de red de Entrada, y Tráfico OUT es el tráfico de red de Salida.

En los experimentos con HMMs Continuos Multivariados (HMMCMs) se utilizaron secuencias de observación que contenían el consumo de ancho de banda de entrada y el de salida. La Figura 5.4 muestra la gráfica del comportamiento del consumo de ancho de banda de red basado en los valores de las variables de entrada y salida.

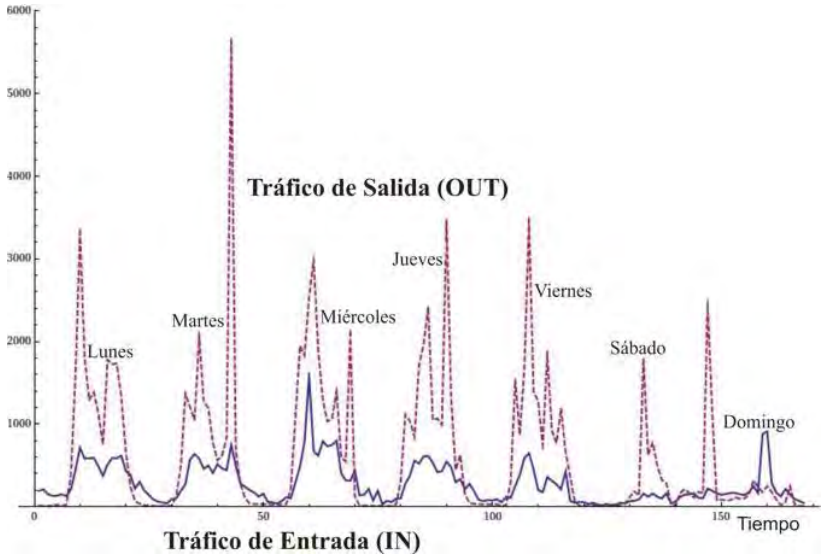

Figura 5.4: Comportamiento del consumo de ancho de banda de red de entrada y salida.

Las variables de consumo de ancho de banda de red de entrada y salida son las que se utilizarán principalmente en el modelado de los HMMs Continuos Multivariados. Existe un experimento en el que se agrega el tiempo a estas dos variables, para formar una secuencia de 3 datos por observación; este experimento de describe en subsecciones más adelante.

En la definición de la secuencia de observación (O_{em}) se emplearon como variables al formato fecha.hora y el consumo de ancho de banda. Con esta secuencia de observación se crearon y entrenaron los HMMCMs con $d = 2$. La secuencia O_{em} contiene los datos del tráfico de red de la semana completa empezando por el día lunes a la 1 a. m. hasta las 12 p. m. del día domingo. En la Figura 5.4 se puede observar la gráfica del comportamiento normal del consumo de ancho de banda de la red en el área de Posgrado en Ingeniería Eléctrica. La gráfica representa el consumo de una semana completa con un registro de cada hora. Los datos de la semana comprenden del Lunes 29 de Noviembre al Domingo 5 de Diciembre de 2010, contabilizando un total de 168 datos.

De la secuencia normal O_{em}, se intercambiaron únicamente los datos del consumo de ancho de banda del día lunes por el sábado y los del sábado por los del día lunes. Generando una secuencia con comportamiento anómalo O_{pm}. Esta secuencia de observación la definimos como secuencia de prueba multivariada, la secuencia tiene la forma:

$$O_{pm} \ = \ [o_{pm1}, o_{pm2}, \ldots, o_{pm168}]^T$$

donde cada dato es una pareja; por ejemplo $o_{pmk} = (1.10, 243.13)$. En estas dos secuencias el primer valor corresponde al día de la semana y la hora. Por ejemplo, el valor 1.10 corresponde al día lunes a las 10 a. m. El segundo valor 243.13 correspondería al valor del ancho de banda de la red utilizado. Otra forma de representar el formato de día-hora podría ser por la expresión en decimal de la hora, por ejemplo el valor $10/24 = 0.42$ representaría las 10 a. m.

5.3. Proceso de detección de anomalías

El proceso de detección de anomalías utilizando HMMs, se lleva a cabo analizando subsecuencias de observación de la serie de tiempo de prueba. Estas subsecuencias están formadas por intervalos de observación temporalmente adyacentes y de longitud constante. A estas subsecuencias las llamaremos *ventanas deslizantes de datos*, o *ventanas de tiempo*.

Las ventanas de tiempo contienen un subconjunto de datos contiguos de tamaño constante; el tamaño se puede especificar en cada análisis de la secuencia de prueba.

El proceso a detalle se describe a continuación:

- El primer paso consiste en contar con la secuencia de observación de prueba, (O_{pu} y O_{pm}), y con el tamaño de ventana de datos a utilizar. Al sistema se le indica el tamaño de ventana w, el modelo HMM (λ) a utilizar y la serie de tiempo de prueba A que va a utilizar en el análisis.

- El sistema comienza a analizar las ventanas de datos deslizantes, generando las probabilidades para cada ventana en cada marca de tiempo. La ventana se desliza una unidad de tiempo a la vez, hasta llegar al último dato en la serie de tiempo de prueba. La probabilidad de cada ventana está dada por

$$windowProb_t = P_t^w(A_t \mid \lambda)$$

  donde A_t es la ventana de tiempo que comienza en t, con $A_t = [O_t,\ O_{t+1}, \ldots,\ O_{t+w-1}]^T$.

- Para detectar si existe alguna anomalía en los datos de la ventana actual A_t, simplemente se comparan los valores de probabilidad de las ventanas con el umbral de referencia. De esta forma si $windowProb_t \geq Threshold$ se considera que los datos observados en esa ventana son normales; en caso contrario el sistema señala una anomalía en los datos de la ventana.

- Esto se aplica para todas las ventanas que se definan para el análisis de datos.

Esta es la forma como el sistema lleva a cabo el análisis de los datos de prueba, que ayuda a determinar probabilísticamente, si alguna secuencia de datos presenta anomalía o no. Este proceso es general, tanto para secuencias de observaciones univariadas como multivariadas.

Los tamaños de las ventanas que se probaron fueron de 3, 4, 5 y 6. Todas las probabilidades generadas por el análisis del HMM y la ventana de tiempo deslizante, fueron comparadas con nuestro umbral de probabilidad (ver Figura 5.5) para determinar si existía una anomalía.

El establecimiento del valor del umbral (*Threshold*) es arbitrario, de acuerdo al análisis del rango de probabilidades de las secuencia de datos con los que se entrenan a

los HMMs. Al definir el valor del umbral se debe tener cuidado en fijarlo en un valor adecuado. Si fijamos Threshold = 1, todos los valores de probabilidad de las secuencias de los datos serían Falsos Positivos –las subsecuencias serían consideradas anomalías–; si el valor de Threshold = 0, entonces todas las probabilidades darían Falso Negativo –todas las subsecuencias serían consideradas normales, presentaran o no anomalías–. Este es el único parámetro que requiere del conocimiento o experiencia previa del usuario, pero todo depende del tipo de sistema que se esté modelando.

Un ejemplo de definición de un valor de Threshold, se observa en la Figura 5.5. Este umbral, definido en 1.0×10^{-9}, es el valor contra el cual se comparan todas las probabilidades obtenidas de la aplicación del HMM a la ventana deslizante de datos, y poder establecer o diferenciar entre el comportamiento normal y anómalo de los datos en la ventana de tiempo. Todo valor de probabilidad generado por la ventana de tiempo ($windowProb_t$) se compara contra el threshold. Si $windowProb_t \geq Threshold$, los datos exhibidos en esa ventana se consideran con comportamiento normal, y anómalo en caso contrario.

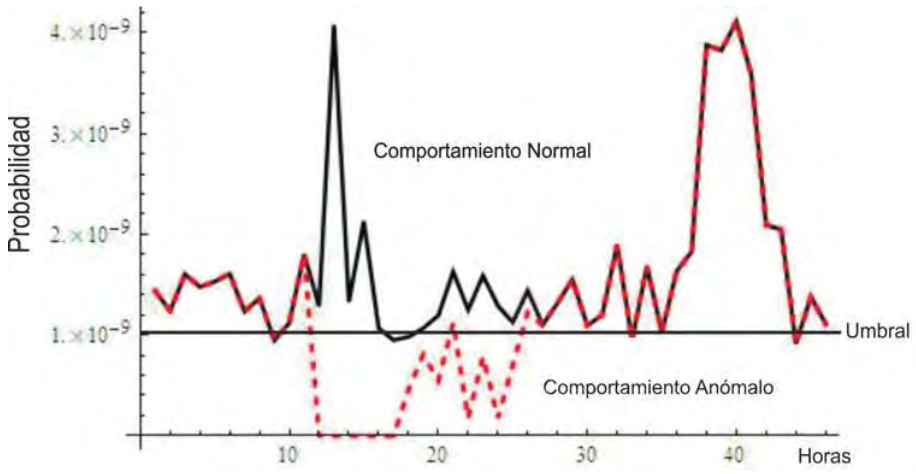

Figura 5.5: Ejemplo de la definición del threshold.

5.4. Resultados de detección usando HMMs

En esta sección se describirán los resultados de aplicar los HMM a la detección de anomalías en series de observación que sirvieron como pruebas. Se utilizará la secuencia de observación O_{pu}, como serie de prueba con presencia de anomalías en sus datos. Los modelos

de HMMs analizarán esta secuencia tratando de encontrar las anomalías. O_{pu} representa el consumo de ancho de banda de red, con 48 datos y con datos sintéticos insertados en la secuencia, que hacen la función de presencia de datos anómalos. El comportamiento de los HMMs, como herramientas de apoyo en la detección de comportamiento anómalo, fue la esperada.

En esta sección se demuestra que la evolución de HMMs es un proceso de optimización de parámetros de los modelos, que no requiere de conocimiento previo para lograr encontrar un modelo óptimo. En cambio, en la optimización de los parámetros de un HMM con el algoritmo de Baum-Welch, depende en gran medida de sus parámetros iniciales. Es decir, requiere de conocimiento previo del análisis de los datos y la estructura a diseñar de los modelos.

5.4.1. HMMCU basado en Baum-Welch

En esta subsección se presentan en resumen las detecciones que pudo realizar el HMM entrenado con Baum-Welch. El proceso de análisis es el mismo que se describió en la sección llamada *Proceso de detección de anomalías*.

La Tabla 5.2 describe la detección de comportamiento anómalo presente en O_{pu}, utilizando HMM-BW. Esta tabla presenta el porcentaje de Aciertos, Falsas Alarmas o Falsos Positivos, y las anomalías no reconocidas (Falsos Negativos).

Tabla 5.2: Resultado de la detección usando un HMM basado en Baum-Welch.

Tamaño de Ventana	%		
	Aciertos	Falso Positivo	Falso Negativo
3	57	43	0
4	60	40	0
5	59	41	0
6	65	35	0

En los datos que presenta la Tabla 5.2 se puede observar que la ventana de datos que presentó mejor desempeño en la detección, fue la ventana de 6 datos. Con un 65 % de aciertos en la detección de anomalías, 35 % de falsos positivos y cero por ciento de falsos negativos. La peor ventana de datos fue la de tamaño de 3 datos, con 57 % de aciertos en la detección de anomalías, 43 % de falsos positivos y cero por ciento de falsos negativos.

5.4.2. HMMCU basado en EP

Con el HMM obtenido de la evolución, del experimento número 3, se analiza, con la ayuda de ventanas de tiempo, los datos de la secuencia de observación O_{pu}; para determinar la probabilidad de que una subsecuencia de datos presente o no anomalías. El proceso de análisis de detección es el mismo que se ha descrito anteriormente en la sección llamada *Proceso de detección de anomalías*.

Los tamaños de ventana que se probaron fueron de 3, 4, 5 y 6 datos; de donde todas las probabilidades que se generaron con la ventana de tiempo deslizante, se compararon con el valor del umbral (threshold) para determinar si existía o no alguna anomalía.

La Tabla 5.3 presenta el porcentaje de Aciertos, Falsas Alarmas o Falsos Positivos, y las anomalías no reconocidas (Falsos Negativos).

Tabla 5.3: Detección de anomalías usando un HMMCU basado en EP.

Tamaño de Ventana	%		
	Aciertos	Falso Positivo	Falso Negativo
3	89	7	4
4	93	2	5
5	89	4	7
6	84	5	11

El experimento reporta el mejor resultado con un tamaño de ventana de 4 datos, usando un HMM basado en EP, con un 93 % de detecciones, 2 % de falsos positivos, y un 5 % de falsos negativos. El peor resultado fue producido por un tamaño de ventana de 6 datos, con 84 % de aciertos en la detección de anomalías, 5 % de falsos positivos, y un 11 % de falsos negativos.

En resumen se puede concluir que los HMMs generados con programación evolutiva tuvieron mucho mayor eficiencia que los entrenados con el algoritmo de Baum-Welch. El algoritmo de Baum-Welch sólo refina u optimiza los valores iniciales del HMM, en cambio los HMMs generados con programación evolutiva no requieren de conocimiento previo y proporcionan mejores resultados.

5.5. Comparación de modelo evolutivo y modelo desarrollado por un experto

Se planeó realizar un experimento para comparar el funcionamiento y desempeño de los HMM evolucionados con EP, contra otro HMM diseñado en base al análisis del comportamiento de los datos y a la experiencia del investigador. El HMM diseñado con conocimiento a priori, se utilizó en la detecció de secuencias anómalas en la carga de un procesador de un servidor de correos electrónicos [Navarrete04].

El modelo contra el cual se comparó fue el HMM creado con 6 estados y desarrollado en el trabajo de tesis titulado "Detección de Anomalías en la Carga de un Procesador, utilizando Modelos Ocultos de Markov" [Navarrete04]. El HMM que se presenta en el trabajo de tesis de Navarrete, fue desarrollado y entrenado con valores de datos que corresponden a la carga promedio del procesador de un servidor de correo instalado en FIRA; las mediciones fueron obtenidas en intervalos de 15 minutos, dando una secuencia de observación de 96 datos en total.

El HMM evolucionado en este trabajo de tesis, sólo requiere de la secuencia de observación que captura el comportamiento o dinámica del sistema. Se tomaron los mismos datos de la secuencia de observación de la carga del procesador del servidor de correos [Navarrete04], y con EP se evolucionó el HMM.

El HMM evolucionado obtuvo mejores resultados de rendimiento que el generado por Navarrete, en la detección de anomalías en la secuencia de carga del procesador. El desempeño del modelo evolucionado fue el esperado en el reconocimiento del comportamiento anómalo en los datos de prueba, que se utilizaron.

La Figura 5.6 presenta los tipos de ataques que se simularon, y que fueron utilizados por los HMMs para conocer su capacidad de detección.

La figura representa la modificación a la secuencia original de los datos con los que se entrenó al HMM [Navarrete04], para insertar valores aleatorios (datos sintéticos) que representaran un aumento en la carga del procesador del servidor de correo. La secuencia original se encuentra representada por la línea continua, y los datos sintéticos por la línea punteada.

En la subfigura a) se representa el Ataque tipo A. En este tipo de ataque se proporcionan incrementos no superiores al 30 % durante la fase de menor carga del servidor. Este ataque, puede ser uno que intenta esconderse durante el tiempo en el que el servidor

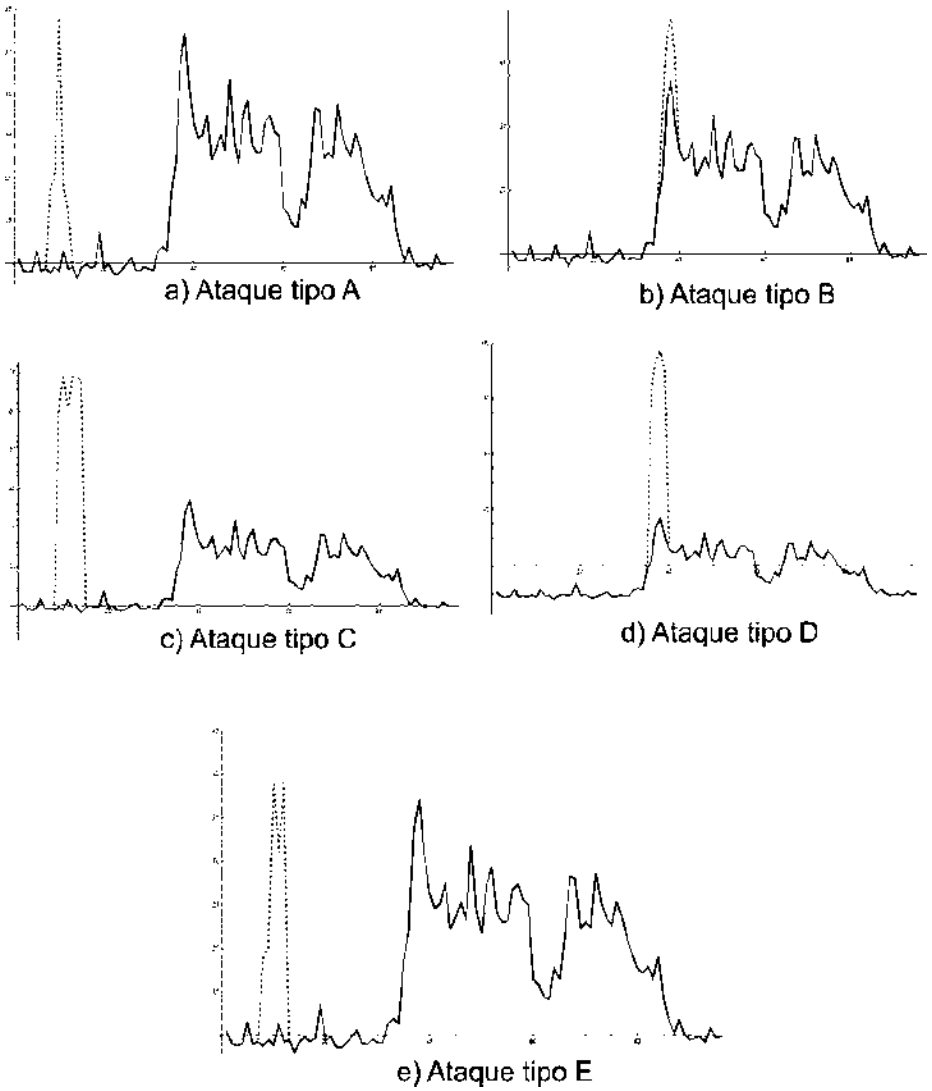

Figura 5.6: Ejemplos de ataques informáticos.

tiene la menor carga.

La subfigura b) representa el Ataque tipo B, que representa un aumento del 10 % durante la fase de mayor carga del servidor. Este ataque puede ser uno que intenta esconderse durante el tiempo en el que el servidor tiene la mayor carga. El posible ataque, pretende pasar desapercibido no aumentando mucho la carga, en momentos donde el servidor ya tiene una carga de trabajo alta.

Subfigura c), Ataque tipo C es un aumentos entre 50 % y 60 % durante la fase de menor carga del servidor. El posible ataque, no intenta esconderse, pues aumenta de manera muy notable la carga del procesador, pero intenta aprovechar que la fase de menor carga corresponde con el tiempo en el que se monitorea menos el servidor (dado que la fase puede coincidir con pausas de trabajo u horarios no laborables).

En la subfigura d) se representa un Ataque tipo D que representa un aumento del 60 % en la carga del servidor durante la fase de su mayor carga. El posible ataque, no intenta esconderse, pues aumenta de manera muy notable la carga del procesador. Este ataque podría tener por objetivo, disminuir (o hasta impedir) el servicio del servidor (ataque tipo Denegación del Servicio) ejecutándolo durante los ciclos de mayor trabajo.

La subfigura e) representa el Ataque tipo E, que consiste entre un 10 % y 30 % de la carga del servidor durante la fase de menor trabajo. Esta posible amenaza puede representar un ataque al servidor, que trata de pasar desapercibido dado que no aumenta de manera notable la carga de trabajo y se ejecuta durante un ciclo de trabajo donde no hay actividad (día no laborable).

En los resultados del HMM utilizado por Navarrete en la detección de anomalías, reporta que para los tipos de ataques C y D el modelo HMM no tuvo problema en detectar la variación en el comportamiento de carga del procesador. El modelo HMM evolucionado por nuestro sistema, también detectó sin problemas estas variaciones de probabilidad en las secuencias dadas por los tipos de ataques C y D.

Ahora bien, el mismo HMM creado por Navarrete en la detección de los ataques A, B y E reporta en sus resultados, que el modelo no fue capaz de detectar las variaciones de probabilidad en las secuencias de datos. En cambio, el HMM obtenido del proceso evolutivo, tuvo los siguiente resultados: ataque tipo A). Reconoció la diferencia de probabilidad con un umbral = 1.4×10^{-04}, usando la ventana de 3 datos, 2 ventanas anómalas de 7. En la ventana con 4 datos con umbral = 1.0×10^{-05}, detectó 3 ventanas con anomalías de 8 en total. En la ventana de 5 datos con umbral = 1.0×10^{-06} detectó 5 ventanas anómalas de 9. Usando la ventana de 6 datos con umbral = 1.0×10^{-08}, detectó 6 ventanas anómales de 10.

En el ataque tipo B), usando ventanas de 3 datos con umbral = 1.0×10^{-04}, el HMM evolucionado detectó 6 ventanas anómalas de 6. En la ventana con 4 datos con umbral = 1.0×10^{-05}, detectó 7 ventanas con anomalías de 7 en total. En la ventana de 5 datos con umbral = 1.0×10^{-06} detectó 7 ventanas anómalas de 8. Usando la ventana de 6 datos

con umbral = 1.0×10^{-07}, detectó 7 ventanas anómales de 9.

El ataque tipo E), usando ventanas de 3 datos con umbral = 1.0×10^{-04}, el HMM evolucionado detectó 3 ventanas anómalas de 7. En la ventana con 4 datos con umbral = 1.0×10^{-06}, detectó 2 ventanas con anomalías de 8 en total. En la ventana de 5 datos con umbral = 1.0×10^{-07} detectó 4 ventanas anómalas de 9. Usando la ventana de 6 datos con umbral = 1.0×10^{-09}, detectó 3 ventanas anómales de 10.

La conclusión del desempeño del HMM evolucionado en los experimento, fue mejor debido que el sistema de detección de anomalías utiliza ventanas deslizantes de 3 a 6 datos, que son subseries de la observación original. Otro factor importante es el umbral que participa en la detección de variaciones de las probabilidades de las ventanas de datos. En la Figura 5.7 se muestra un ejemplo del valor del umbral que se estableció para tomarlo como referencia en los valores de probabilidad generados por la ventana de 3 datos. En la subfigura a) de la Figura 5.7 se observan las probabilidades generadas por las ventanas de datos, con la secuencia original normal y con la secuencia de observación en presencia de anomalías. En la subfigura b) se presenta la misma gráfica de probabilidades de ambas secuencias (normal y anormal), pero ampliada, donde se observa el valor del umbral que se estableció. El valor del umbral fue establecido en 1.4×10^{-04}. El umbral se definió en ese valor para poder abarcar las probabilidades de las ventanas de los datos con anomalías. Si el umbral se desplaza a un valor superior, detectará más anomalías, pero también se incrementarán los falsos positivos.

La técnica empleada verifica las probabilidades de las ventanas de datos contral el umbral, y es más efectivo verificar subsecuencias de datos y encontrar anomalías, que verificar la probabilidad de toda la secuencia de observación. Con estos dos elementos, ventanas de datos y umbral, más el HMM, que participan en la detección de las anomalías, fue posible detectar las variaciones de probabilidad que el modelo creado por Navarrete no pudo detectar. En todas las ventanas de datos se notaron variaciones en la probabilidad. En todos los ataques fue posible detectar la variación de la probabilidade de las ventanas de datos. El ataque más difícil de detectar es el ataque tipo A, debido a que el porcentaje de uso del procesador cae dentro del rango normal de uso. Con las ventanas deslizantes fue posible detectar la variación de la probabilidad en esa ventana, y al compararla contra el umbral, fue detectada la anomalía. Los ataque restantes fueron más sencillos de detectarlos porque incrementan el porcentaje de uso del procesador del servidor, reduciendo la probabilidad que ese valor de dato haya sido dado en el entrenamiento al HMM.

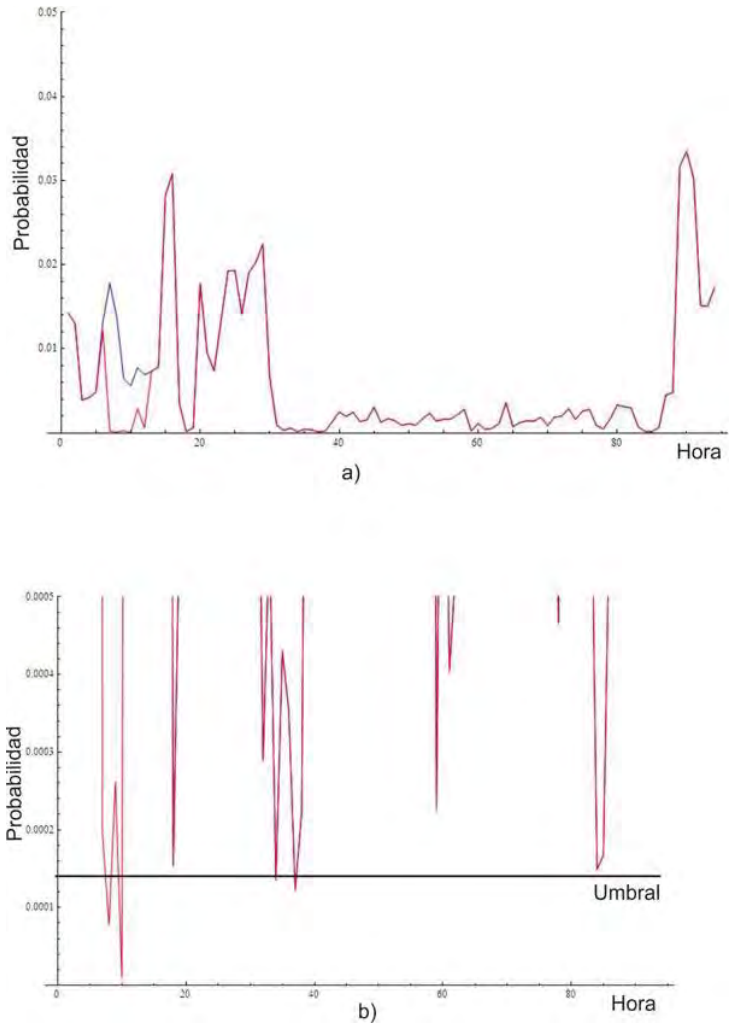

Figura 5.7: Probabilidades generadas por la ventana de datos.

La conclusión del experimento para el caso del trabajo de Navarrete es que toman para el análisis de la detección de anomalías toda la secuencia de observación. Logrando con esta técnica una baja efectividad. Si existe un ataque oculto (que se comporta como un uso normal de carga del procesador) el HMM es incapaz de detectarlo como anómalo, debido a que la probabilidad de la secuencia completa de ser generada por el HMM se mantiene sin variación.

La diferencia que presenta el sistema de detección que se desarrolló en [Navarrete04], es que utilizan toda la secuencia de observación para detectar la anomalía. En cambio en

nuestro trabajo se toman subsecuencias de datos o ventanas, y se analizan en busca de datos anómalos, por lo que aumenta la probabilidad de encontrar los datos que presenten anomalías.

5.6. Comparación de modelos entrenados con EP y Baum-Welch

En esta sección se compararán los modelos de HMMs evolucionados con programación evolutiva contra otro modelo entrenado con el algoritmo de Baum-Welch. Para el caso del HMM entrenado con Baum-Welch, se toma el HMM creado de manera aleatoria. Es decir, sus parámetros como transición entre estados, valor de su media y varianza se asignan con valores completamente aleatorios. Simulando que la creación del modelo se hace por una persona sin experiencia en la construcción de los HMMs.

5.6.1. HMMCUs generados con EP

A continuación se presentan algunos de los HMMs resultantes de la evolución usando EP. Cabe resaltar que durante el entrenamiento de los modelos HMMCUs, sólo se utilizó la variable de consumo de ancho de banda de la red (O_{eu}), y con esta secuencia de observación se crearon y optimizaron estos modelos.

Tabla 5.4: HMMCUs generados con EP.

Núm. de Experimento	Población	Núm. de Generaciones	Núm. de Edos.	$P(O \mid \lambda)$	Tiempo min.
1	500	500	5	7.70688×10^{-103}	226.6
2	500	250	5	7.70688×10^{-105}	114.73
3	250	250	6	2.52091×10^{-75}	67.44
4	200	50	6	7.19032×10^{-98}	13.15
5	1000	500	8	1.19399×10^{-88}	321.93

La descripción de la tabla es la siguiente: la primera columna de la Tabla 5.4 corresponde al número de experimento realizado, la segunda columna muestra el número de la *Población* de individuos involucrados en la evolución; la tercer columna nos muestra el *Número de Generaciones* utilizadas para la obtención del mejor modelo. La cuarta columna representa el *Número de Estados* del HMM generado de la población y de las generaciones. La quinta columna, $P(O \mid \lambda)$, nos representa la probabilidad que tiene el modelo de haber

generado la secuencia de observación O_{eu}, dada como comportamiento normal. La última columna representa el tiempo máquina que se requirió en la creación y entrenamiento de los modelos, expresado en minutos.

Para nuestro análisis de detección se usó el modelo producido por el experimento 3, el cual tiene 6 estados y una probabilidad de observar la secuencia O_{eu}, de 2.52091×10^{-75}. La gráfica de este modelo se presenta en la Figura 5.8.

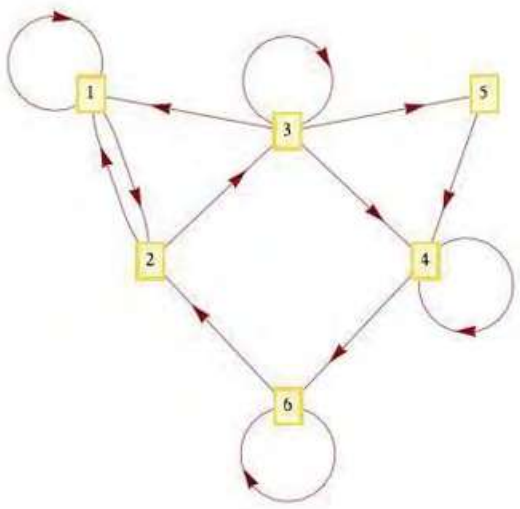

Figura 5.8: HMMCU creado con EP.

5.6.2. HMMCUs entrenados con Baum-Welch

En esta sección se presentan los HMMs creados de manera aleatoria y entrenados con el algoritmo de Baum-Welch (BW). El modelo que presente la probabilidad más alta para la secuencia de entrenamiento O_{eu}, será utilizado en la detección de anomalías. En la Figura 5.9 se muestra la estructura de un HMMCU creado de manera aleatoria. La aleatoriedad se encuentra en los valores que forman la estructura del HMM, es decir, los valores de transición entre estados fueron dados de manera aleatoria, cumpliendo la condición que la suma de las transiciones de salida de cada estado suman 1. Los valores de la media y varianza de cada estado también fueron proporcionados de manera aleatoria.

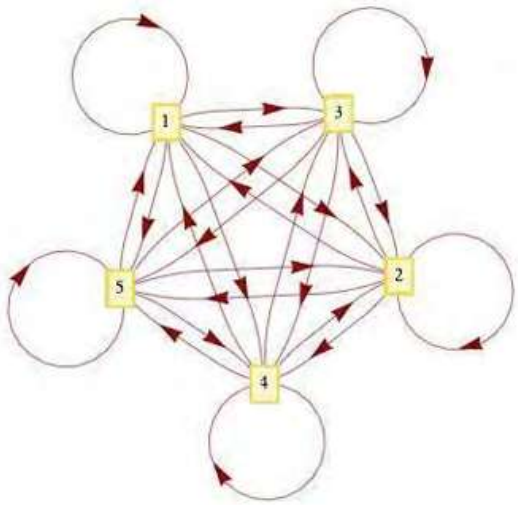

Figura 5.9: HMMCU creado con valores aleatorios sin "entrenar".

El proceso de entrenamiento consiste en seleccionar cada modelo, por ejemplo, se toma el HMM de la Figura 5.9 y se "entrena" con el algoritmo de *BW*, obteniendo de este proceso un HMMCU reestimado, el cual se puede observar en la Figura 5.10. El HMMCU fue entrenado con la secuencia de observación de datos O_{eu}, dada por la serie de tiempo del consumo de ancho de banda usado en la red. El HMMCU que se obtuvo de la reestimación con BW, fue a través de un proceso de 50 iteraciones de refinamiento de los parámetros.

En la Tabla 5.5 se describen los modelos creados y ajustados con al algoritmo BW. Una vez definido un HMM de manera aleatoria, se le aplicó la reestimación de sus parámetros usando el algoritmo de BW, tratando de obtener las mejores estimaciones para los modelos. Se comenzó con HMMCUs de 3 estados, como se puede observar en la segunda columna de la Tabla 5.5. La tercera columna, etiquetada $P(O \mid \lambda)$, muestra la probabilidad que presenta cada modelo con respecto a la serie de tiempo O_{eu}.

A todos los modelos creados se les aplicó el algoritmo de BW con 50 iteraciones de reestimación. Todos los modelos fueron creados con parámetros aleatorios; estos modelos presentan un valor de probabilidad baja para la secuencia de observación O_{eu}. El valor de probabilidad de observar O_{eu} generada por los modelos es baja, debido a que están tratando

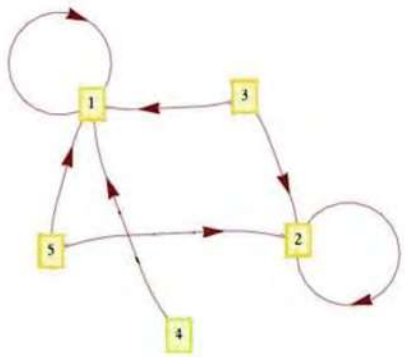

Figura 5.10: HMMCU con parámetros reestimados con BW.

Tabla 5.5: HMMCUs "entrenados" con el algoritmo de Baum-Welch.

Núm. de Experimento	Núm. de Edos.	$P(O \mid \lambda)$
1	3	1.8920×10^{-123}
2	4	6.44442×10^{-175}
3	5	3.43836×10^{-137}
4	6	1.07894×10^{-127}
5	7	5.90112×10^{-125}
6	8	1.09739×10^{-119}
7	9	1.22676×10^{-119}
8	10	6.55885×10^{-172}

de representar la probabilidad total de la secuencia de observación O_{eu}, y porque los valores de los parámetros de inicio de los modelos fueron generados de manera aleatoria.

Para las pruebas efectuadas de detección de anomalías usando un HMM-BW, se seleccionó el modelo de 9 estados, del experimento número 7, con una probabilidad de observar O_{eu} de 1.22676×10^{-119}. Este modelo es el que mejor califica para detectar situaciones anómalas en la secuencia de datos.

5.7. HMMCMs generados con EP

En secciones anteriores se mencionaron a los HMM Continuos Univariados y su forma de evolucionarlos, utilizando Programación Evolutiva. La característica principal de estos modelos es que para su entrenamiento, sólo necesitan la secuencia de observación de una sola variable.

5.7.1. Modelo HMMCU

En esta subsección se describen los datos de observación que se utilizaron para hacer evolucionar un modelo HMMCU y utilizarlo en la detección de anomalías. En la evolución del HMMCU sólo se tomó como referencia, el valor de la variable del consumo de ancho de banda de entrada de red, Figura 5.13. Esto es, el vector O_{pm} transfiere a la serie de tiempo de una variable O_{pu2}, los valores del consumo de ancho de banda de red.

Al sistema evolutivo se le que proporcionaron las características que se describen en la Tabla 5.6, generando al modelo HMMCU, que se muestra en la Figura 5.11.

Tabla 5.6: HMMCU generado con EP.

Num. de Datos	Población	Num. de Generaciones	Num. de Edos.	$P(O \mid \lambda)$	Tiempo
168	200	150	5	2.56×10^{-441}	41.14

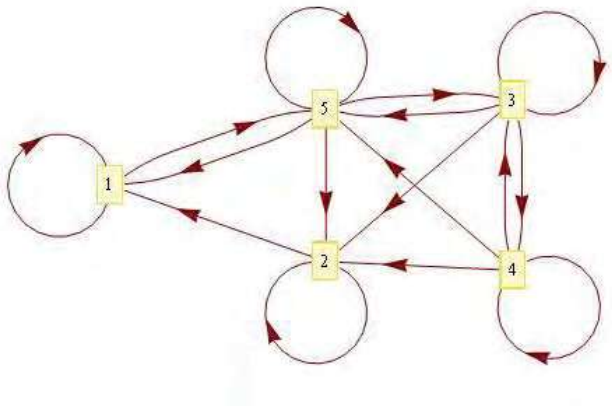

Figura 5.11: HMMCU evolucionado del ancho de banda con 168 datos.

5.7.2. Modelo HMMCM

En esta subsección se describen los HMM Continuos Multivariados (HMMCMs) los cuales para modelar el comportamiento o dinámica del sistema necesitan una secuencia de observación de dimensión d, es decir, cada observación es un vector de d datos; la secuencia de observación es de longitud T. Esto de manera lógica complica el análisis y desarrollo del sistema de detección de anomalías, y sobre todo la evolución de los HMMs con EP.

En la Tabla 5.7 se observan algunos ejemplos de HMMCMs, que se generaron con la EP y con la secuencia de observación O_{em}.

Tabla 5.7: HMMCMs generados con EP.

Experimento	Núm. de Datos	Población	Núm. de Generaciones	Núm. de Edos.	$P(O \mid \lambda)$	Tiempo min.
1	31	100	100	3	1.17×10^{-94}	26.84
2	31	150	100	4	8.52×10^{-3}	34.14
3	31	200	150	6	9.88×10^{-1}	19.31
4	31	20	50	14	1.71×10^{-60}	17.65
5	31	50	50	16	4.79×10^{-59}	52.54
6	168	30	10	5	7.44×10^{-558}	8.61
7	168	200	150	7	7.49×10^{-529}	847.98
8	168	200	150	15	6.22×10^{-1076}	2021.95

La descripción de la evolución de los HMMCMs se menciona en el capítulo 4; en esta parte se describirá la forma de utilizarlos en la detección de anomalías.

Los datos que se utilizaron para crear y entrenar a los HMMCMs, de los experimentos del 1 al número 7, fueron los datos de fecha, con formato de *dia.hora*, y el valor de uso de ancho de banda de entrada. Estos dos valores se utilizaron como secuencia de observación con $d = 2$. El sistema se encargó de hacer evolucionar y obtener el mejor HMMCM; un ejemplo de un HMMCM evolucionado se observa en la Figura 5.12.

El modelo que se utilizó para la detección de anomalías en O_{pm}, fue el del experimento número 7, el cual tiene 7 estados. El modelo fue evolucionado en 150 generaciones, con una población de 200 individuos, presentando una probabilidad, para el vector de observaciones, de 7.492×10^{-529}, como se puede observar en la Tabla 5.7. Los resultados de detección de anomalías de este modelo se presentan en la subsección siguiente, donde se compara contra un HMMCU.

En la evolución hecha en el experimento número 8, se emplearon los datos de fecha,

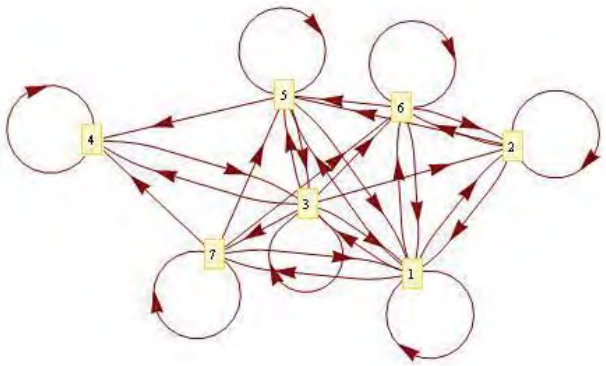

Figura 5.12: HMMCM evolucionado con dos variables continuas.

con formato de *dia.hora*, el valor de uso de ancho de banda de entrada (IN) y el uso de ancho de banda de salida (OUT) (ver Tabla 5.1), la representación gráfica de estos valores se observa en la Figura 5.13.

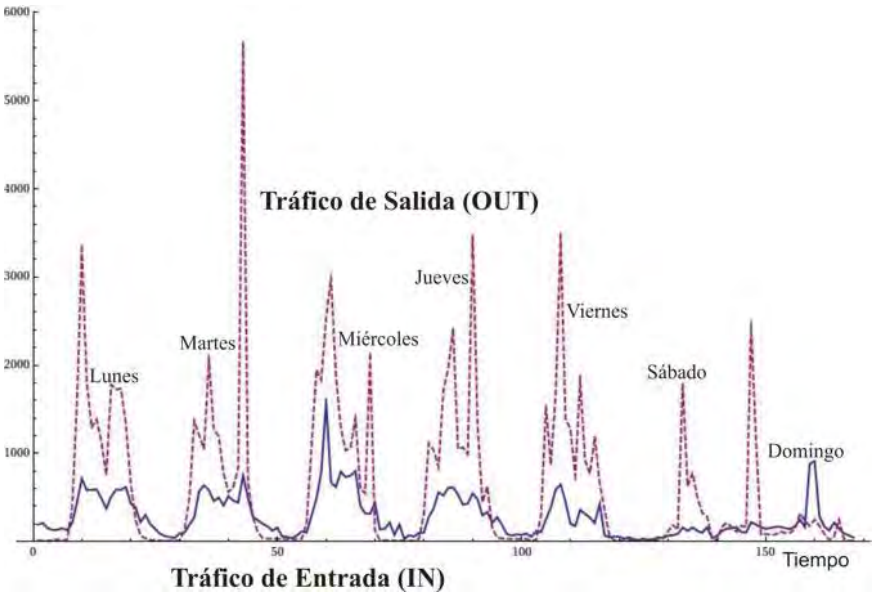

Figura 5.13: Tráfico de consumo de ancho de banda de entrada y salida.

Estos tres valores se utilizaron como secuencia de observación, para $d = 3$. El sistema se encargó de hacer evolucionar y obtener el mejor HMMCM de 15 estados, como

se observa en la Figura 5.14. Este mismo modelo se utilizó en la detección de anomalías en O_{pm}, más el tráfico de red de salida. El modelo fue evolucionado en 150 generaciones, con una población de 200 individuos, presentando una probabilidad, para el vector de observaciones de 3 datos, de 6.22×10^{-1076}, como se puede observar en la Tabla 5.7. Los resultados de detección de anomalías de este modelo se presentan en la subsección siguiente, donde se compara su uso y grado de certeza contra otros HMMCs.

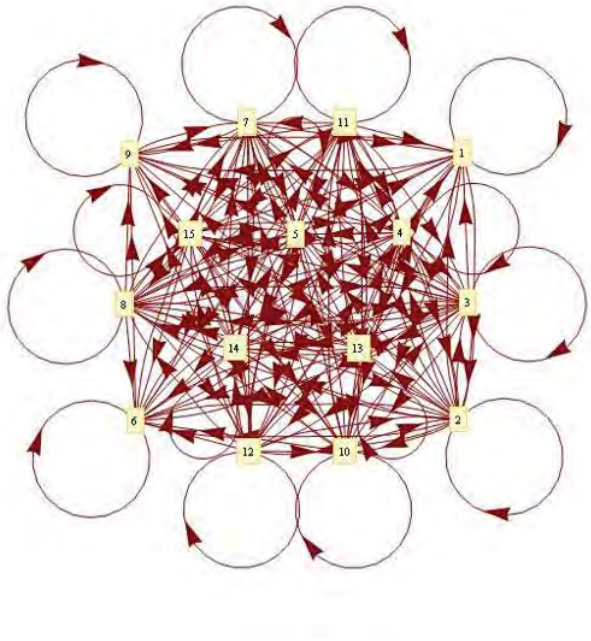

Figura 5.14: HMMCM evolucionado correspondiente a un sistema con tres variables continuas.

5.7.3. Comparación de HMMCU y HMMCM en la detección

En esta sección se evalúa la efectividad de la detección de anomalías para datos multivariados contra la detección de datos univariados. El HMMCM se basará en el análisis de dos variable como son: fecha, con el formato de *dia.hora* y el dato de consumo de ancho de banda de red. El HMMCU sólo toma como referencia los datos de la variable de consumo de ancho de banda de red.

La prueba consiste en determinar que modelo es mejor en la detección de datos con

comportamiento anómalo en la secuencia dada, en este caso O_{pm}. La serie de tiempo O_{pm} tiene los datos de consumo de ancho de banda del día sábado, ubicados en el día lunes, y los del día lunes en los datos del sábado (Figura 5.15). Es decir, los datos fueron intercambiados y se procede al análisis por parte de los modelos para determinar cual tiene mayor grado certeza en la detección.

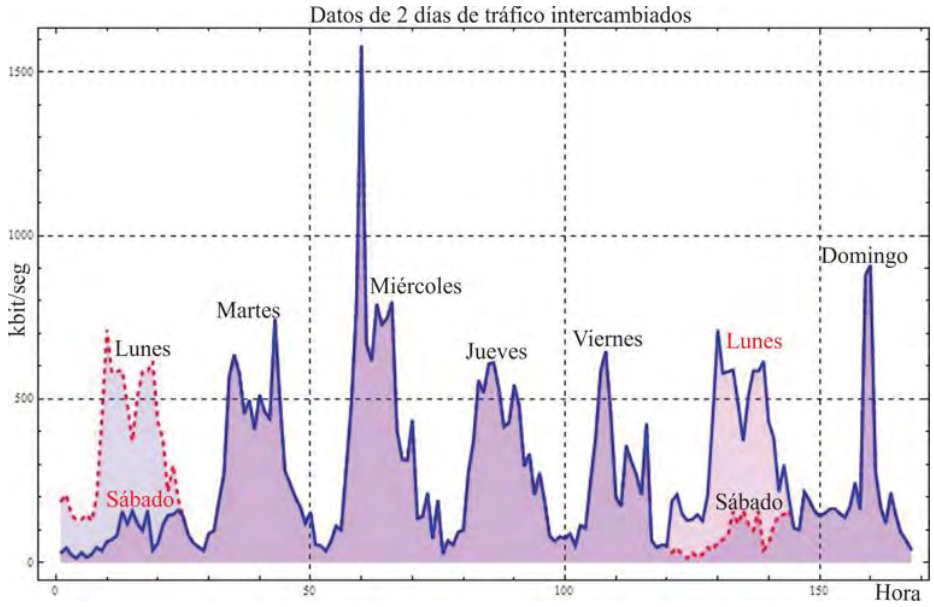

Figura 5.15: Consumo del ancho de banda con días intercambiados.

Para darle al modelo la noción de tiempo y sea capaz de reconocer que el tráfico considerado normal presente en un día entre semana representa una anomalía si se presenta en un fin de semana o día festivo, se ha incluido la variable tiempo en los modelos.

Era de esperarse que la variable tiempo no siga una distribución normal, lo cual fue corroborado aplicando la prueba Anderson-Darling [Daniel83]. Sin embargo, apoyados en el teorema del límite central, si tenemos una muestra de más de 30 datos, el error incurrido al usar la distribución normal es mínimo [Daniel83]. Esta suposición fue corroborada por un buen comportamiento en la detección de anomalías al introducir un tráfico que sería normal entre semana a un día no hábil.

El modelo del HMMCU que se utilizó en esta comparación, fue el evolucionado con el valor de la variable del consumo de ancho de banda de red de entrada, y la gráfica del modelo HMMCU se encuentra en la Figura 5.11.

Las ventanas de subsecuencias de datos que se analizaron con el HMMCU fueron de tamaño 3, 4, 5 y 6 datos. Todas las probabilidades que se generaron con la ventana de tiempo deslizante y el HMMCU, se compararon con el valor del umbral para determinar si existía o no alguna anomalía. Finalmente se calculó el porcentaje de aciertos del proceso de detección de la serie O_{pu2}.

La Tabla 5.8 presenta los porcentajes de Aciertos, Falsas Alarmas o Falsos Positivos, y las anomalías no reconocidas (Falsos Negativos), que son los resultados obtenidos de usar el HMMCU, obtenido del sistema evolutivo, en la detección de anomalías.

Tabla 5.8: Resultados de detección usando un HMMCU.

Tamaño de Ventana	%		
	Aciertos	Falso Positivo	Falso Negativo
3	0	0	100
4	12	0	88
5	8	0	92
6	25	0	75

Observando la Tabla 5.8 se nota que el HMMCU, entrenado únicamente con la variable del consumo del ancho de banda de red, no fue capaz de detectar el cambio de datos de los días lunes y sábado de la secuencia O_{pu2} (tomados de O_{pu2}), en gran medida. La ventana de datos que mejor se comportó en la detección, fue la de tamaño 6, con un 25 % de reconocimiento y un 75 % de falla en la detección. La ventana de datos que menos reconocimiento logró fue la de tamaño 3, con 0 % de detección y con un 100 % de fallas. El intercambio de los datos de los días lunes y sábado de consumo de ancho de banda para el modelo HMMCU, casi pasó desapercibido.

Para el caso de prueba del modelo HMMCM con $d = 2$, las ventanas de subsecuencias de datos que se probaron fueron de tamaño 3, 4, 5 y 6 datos. Todas las probabilidades que se generaron con la ventana de tiempo deslizante y el HMMCM, se compararon con el valor del umbral (threshold) para determinar si existía o no alguna anomalía. Finalmente se calculó el porcentaje de aciertos del proceso de detección de la serie O_{pm}. El modelo gráfico del HMMCM se observa en la Figura 5.12.

La Tabla 5.9 presenta los porcentajes de Aciertos, Falsas Alarmas o Falsos Positivos, y las anomalías no reconocidas (Falsos Negativos), que son los resultados obtenidos de usar el HMMCM en la detección de anomalías.

Observando la Tabla 5.9 se nota que el sistema fue capaz de detectar el cambio

Tabla 5.9: Resultados de detección usando un HMMCM con $d = 2$.

Tamaño de Ventana	%		
	Aciertos	Falso Positivo	Falso Negativo
3	52	0	48
4	58	0	42
5	63	0	37
6	71	0	29

de datos de los días lunes y sábado de la secuencia O_{pm}. La ventana de datos que mejor se comportó en la detección, fue la de tamaño 6, con un 71 % de reconocimiento y un 29 % de falla en la detección. La ventana de datos que menos reconocimiento logró fue la de tamaño 3, con 52 % de detección y con un 48 % de falla. Pero en todas las ventanas se logró identificar un patrón o comportamiento anómalo, aunque en diferente grado de certeza.

Por último, para el caso de prueba del modelo HMMCM con $d = 3$, las ventanas de subsecuencias de datos que se probaron fueron de tamaño 3, 4, 5 y 6. Todas las probabilidades que se generaron con la ventana de tiempo deslizante y el HMMCM se compararon con el valor del umbral para determinar si existía o no alguna anomalía. Finalmente se calculó el porcentaje de aciertos del proceso de detección de la serie O_{pm}, más el tráfico de red de salida. El modelo gráfico del HMMCM se observa en la Figura 5.14.

La Tabla 5.10 muestra los resultados obtenidos de usar el HMMCM con $d = 3$, obtenido del sistema evolutivo, en la detección de anomalías. En la Tabla 5.10 se presentan los porcentajes de Aciertos, Falsos Positivos, y las anomalías no reconocidas (Falsos Negativos).

Tabla 5.10: Resultados de detección usando un HMMCM con $d = 3$.

Tamaño de Ventana	%		
	Aciertos	Falso Positivo	Falso Negativo
3	63	0	37
4	67	0	33
5	71	0	29
6	71	0	29

Los valores presentados en la Tabla 5.10 muestran que el modelo de HMMCM fue capaz de detectar anomalías en los datos que se le proporcionaron como secuencia de prueba. La secuencia de observación multivariada de 3 datos estaba formada por el formato de *dia.hora*, por el consumo de tráfico de entrada y el consumo de tráfico de salida de red,

en un período de 7 días. A esta secuencia de observación únicamente se le intercambiaron los datos del día lunes por los del día sábado, y los del día sábado por los del lunes.

Observando la Tabla 5.10 se nota que el sistema también fue capaz de detectar el cambio de datos de los días lunes y sábado de la secuencia O_{pm}, más la variable de tráfico de red de salida. Las ventanas de datos que mejor se comportaron en la detección, fueron las de tamaño 5 y 6, con un 71 % de reconocimiento y un 29 % de falla en la detección. La ventana de datos que menos reconocimiento logró fue la de tamaño 3, con 63 % de detección y con un 37 % de falla. En todas las ventanas se logró identificar un patrón o comportamiento anómalo, y en comparación con el modelo HMMCM de 2 datos, se incrementó el porcentaje de detección de anomalías en las ventanas de datos de 3, 4 y 5, aunque en la ventana 6 no hubo cambios.

5.7.4. Detección de una anomalía real

Una última prueba consistió en evolucionar un HMM con los datos de la secuencia de observación del tráfico de red de entrada y salida de la subred. Los datos corresponden a 7 días de la semana con lectura del tráfico de red cada hora, dando como resultado una secuencia de 168 datos en total, como se observa en la Figura 5.16.

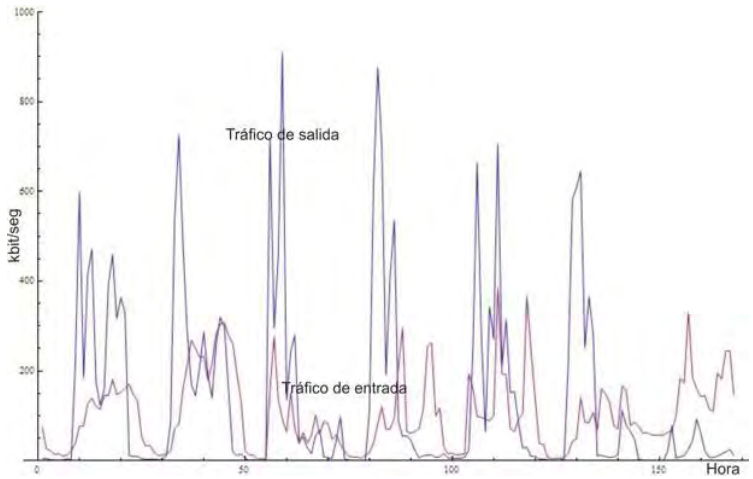

Figura 5.16: Representación gráfica del tráfico de red de entrada y salida.

El HMM multivariado (HMMCM) evolucionado que se obtuvó con la observación multivariada de la dinámica del sistema, presenta 21 estados y se muestra en la Figura 5.17.

El HMMCM fue creado y entrenado con los valores de los datos de la dinámica del sistema, representados por la secuencia de observación normal (Figura 5.16), y tomando los valores de esta secuencia como probabilísticamente normales.

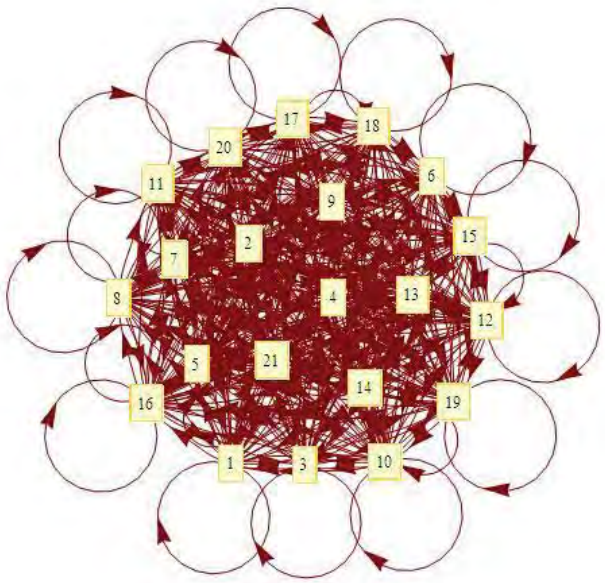

Figura 5.17: HMMCM evolucionado con el ancho de banda de entrada y salida.

El modelo HMMCM se utilizó para detectar las anomalías presentes en la secuencia de observación de los datos del consumo de ancho de banda real de la subred. El comportamiento del consumo de ancho de banda de red con comportamiento anómalo se grafica en la Figura 5.18; estos valores en los datos ocasionan un comportamiento distinto a la dinámica normal del sistema

En el experimento hecho con el sistema detector de anomalías, basado en HMMCM, el umbral y la secuencia de observación anómala, (Figura 5.18), detectó que cierto conjunto de valores de datos caían fuera del umbral o comportamiento normal de la dinámica del sistema, considerando a estos datos como probabilísticamente anómalos. Es decir, que sus valores de probabilidad presentaban una magnitud muy baja del comportamiento habitual del consumo de ancho de banda de red.

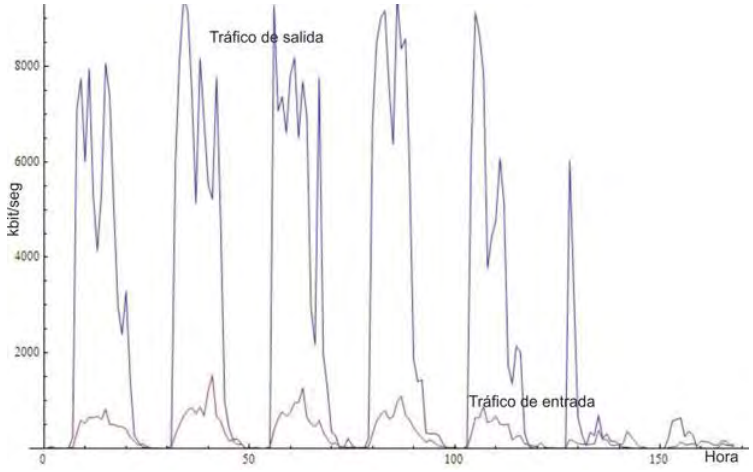

Figura 5.18: Consumo real anómalo del ancho de banda de entrada y salida.

En las pruebas hechas en el experimento se obtuvieron los siguiente resultados: con una ventana de 3 datos y un umbral de 1.0×10^{-17} se detectaron 82 ventanas con datos anómalos de 166 ventanas en total. En la ventana de 4 datos y un umbral de 1.0×10^{-22} se detectaron 88 ventanas con anomalías de 165 ventanas en total. En las ventanas de 5 datos con un umbral de 1.0×10^{-24} se detectaron 106 ventanas con anomalías de 164. En la ventana de 6 datos con un umbral de 1.0×10^{-33} se detectaron 100 ventanas que presentaban anomalías de un total de 163 ventanas. El resultado proporciona la conclusión de que entre más datos contenga la ventana, el valor de la probabilidad de la ventana se remarca, tanto como para caer dentro del umbral o fuera de éste. En este experimento la ventana de 5 datos presentó mejor rendimiento como consecuencia del tamaño de la ventana y por la especificación del umbral.

El sistema de detección de anomalías fue capaz de detectar los datos anómalos sin dificultad, gracias a la diferencia en el comportamiento de la dinámica del sistema modelado. Es decir, los valores de los datos con los cuales el HMM fue entrenado, no correspondían con los valores observados y analizados por el HMM en las ventanas de datos.

El esquema de detección de anomalías, basado en HMMCMs, detectó que el tráfico de red en la subred presentaba desviaciones considerables en su comportamiento normal.

Al aplicar el modelo HMM evolucionado a las subsecuencias de datos o ventanas, los resultados de las probabilidades fueron muy bajas en las ventanas que presentaban datos anómalos.

5.8. Curvas ROC

En la teoría de detección de señales una curva ROC (acrónimo de Receiver Operating Characteristic, o Característica Operativa del Receptor) es una representación gráfica de la sensibilidad frente a (1 - especificidad) para un sistema clasificador binario según se varía el umbral de discriminación.

Otra interpretación de este gráfico es la representación de la razón de verdaderos positivos frente a la razón de falsos positivos según se varía el umbral de discriminación (valor a partir del cual decidimos que un caso es un positivo).

Se aplicaron las curvas ROC a los modelos generados en este trabajo de tesis, para comprobar su comportamiento con la definición de los valores del umbral.

La Figura 5.19 muestra el dominio para la definición correcta del umbral en la detección de anomalías. En este experimento se realizó una iteración de 95 ciclos, desplazando el umbral de 0 a 1, con el empleo de un HMM evolucionado, una ventana de 3 datos, y con una secuencia de observación que presentaba anomalías. En los datos de la secuencia existen anómalías que se encuentran claramente identificadas, por lo que el modelo detecta la variación de las probabilidades de los datos dentro de los umbrales definidos, y es por eso que la curva está pegada casi en su totalidad al eje de la sensibilidad. La secuencia de observación con datos anómalos utilizada en este experimento, se puede ver en la gráfica de la Figura 5.3.

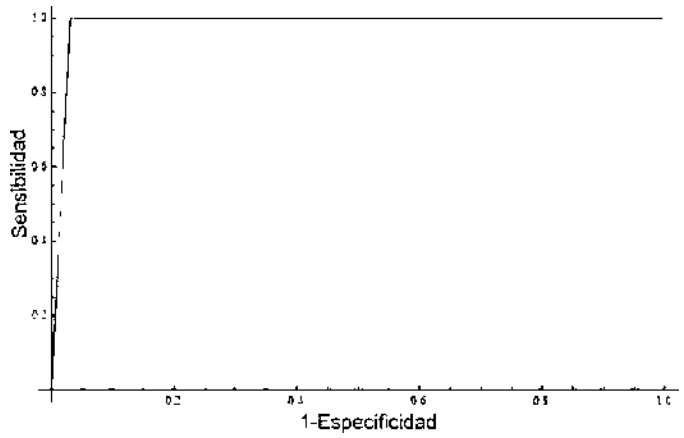

Figura 5.19: ROC de un HMM en la detección de anomalías.

Al encontrarse pegada la gráfica de la curva al eje de sensibilidad, significa que el

modelo fue capaz de detectar las anomalías con el manejo del umbral. Es decir, que tiene propiedad de poder discriminar correctamente entre datos anómalos y normales.

En la Figura 5.20 se muestra otro ejemplo de la curva ROC que corresponde a la detección de anomalías con un HMM evolucionado y una secuencia de observación con datos anómalos, llamado en [Navarrete04] Ataque tipo B (ver Figura 5.6). Se manejaron los valores de umbral dentro del mismo ciclo que los anteriores ejemplos. La secuencia de observación presenta algunos valores de datos que no se le dieron al HMM en el momento de su entrenamiento. Es decir, estos datos anómalos no reflejan la dinámica del comportamiento normal del sistema y se salen de los valores de los datos considerados como normales. Así, el HMM junto con el umbral definido fue capaz de detectar las anomalías, generando que la curva de sensibilidad esté casi sobre el eje Y. Es decir, que los valores que presentan anomalías fueron claramente identificados.

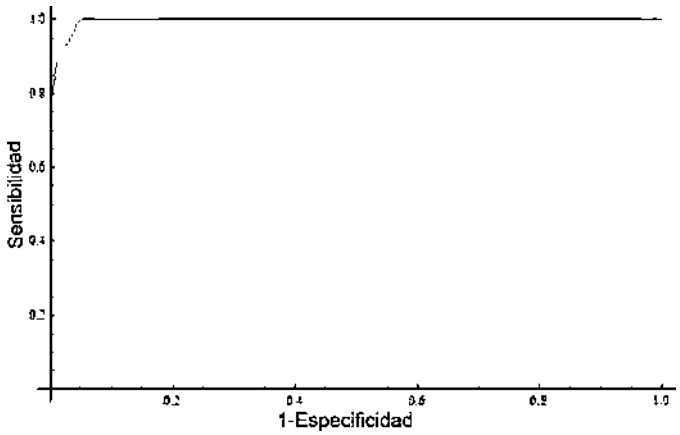

Figura 5.20: Generación de la curva ROC en el manejo del umbral.

La Figura 5.21 muestra el dominio para la definición correcta del umbral, en una iteración de 95 ciclos, el umbral comprende valores entre 0 a 1. En este experimento se empleó un HMM entrenado con Baum-Welch y con una ventana de 6 datos. La secuencia de observación que utilizó presenta anomalías que se encuentran claramente identificadas, Figura 5.3. El HMM, presentó dificultades para discriminar correctamente entre valores normales y anómalos. El HMM fue entrenado con Baum-Welch, pero los parámetros inciales del modelo fueron dados de manera aleatoria, tanto las transiciones como los valores de su media y varianza. Lo que indica que el HMM sólo es capaz de detectar ciertas anomalías

con certeza. Al incrementar o disminuir el umbral repercutirá en falsas alarmas.

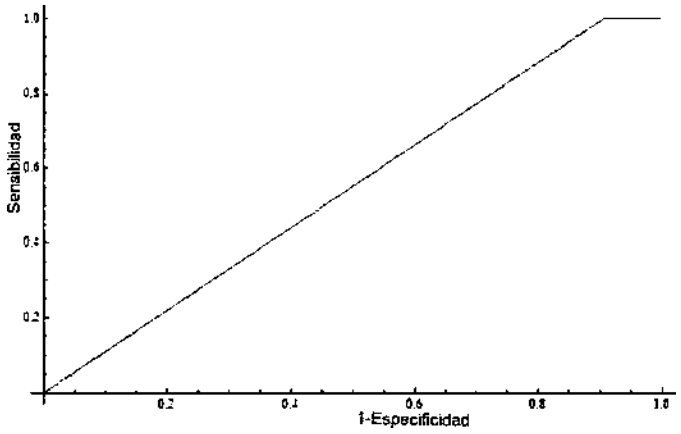

Figura 5.21: Curva ROC para un HMM entrenado con BW.

En la Figura 5.22 se presenta la gráfica de la curva ROC, para un modelo de HMM evolucionado y utilizado en la detección de anomlías. El HMM se utiliza con diversos valores de umbral y con una ventana de datos con secuencias anómalas. La curva de la Figura 5.22 reporta que tan eficiente se comporta el modelo en la detección de anomalías.

En la generación de la curva se hizo una iteración de 95 ciclos incrementado el valor del umbral en cada iteración. Los valores de umbral que se manejaron fue dentro del rango de valores de 0 a 1. el umbral se tomó para la comparación y análisis de las probabilidades generadas por las ventanas de datos. En este experimento se utilizó la ventana de 3 datos. El experimento corresponde al análisis de una secuencia de observación sintética que presenta datos anómalos, pero que son difíciles de detectar, ya que estos datos se encuentran ocultos dentro de la actividad normal del sistema. La secuencia fue tomada del trabajo de [Navarrete04], llamada secuencia de Ataque tipo A, ver Figura 5.6.

La curva de la Figura 5.22 muestra, para el HMM y la secuencia de observación dada, que los valores del umbral se encuentran en un punto medio entre los valores de sensibilidad (verdaderos positivos) y (1-especificidad) –verdaderos negativos–. Lo que indica que el HMM con un umbral definido sólo es capaz de detectar ciertas anomalías con certeza. Al incrementar o disminuir este valor repercutirá en falsas alarmas.

Las curvas ROC son una gran herramienta para diagnosticar el comportamiento de los modelos generados con programación evolutiva, en la detección de anomalías. Las curvas ROC proporcionan una manera de ver de manera gráfica, el comportamiento del

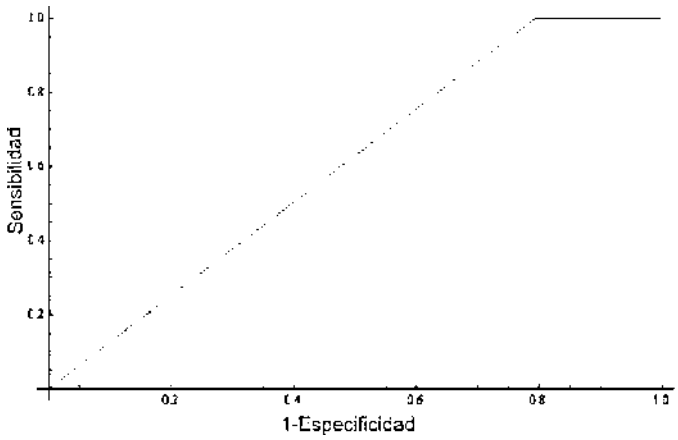

Figura 5.22: Curva ROC para un HMM evolucionado.

umbral y el HMM en su desempeño en el análisis de las secuencias de observación.

5.9. Comparación de los HMMs utilizados

De los experimentos realizados y observados se puede concluir que los HMM optimizados con el algoritmo de Baum-Welch, necesitan para su correcta operación lo siguiente:

- Un HMM con un número inicial de estados.
- Un punto de partida. Es decir, una estimación de probabilidad de qué estado y qué transiciones hay entre estos.
- Una función de distribución de probabilidad para cada estado.

Otra observación es que los modelos que se crearon y entrenaron con el algoritmo de BW, no funcionaron adecuadamente. Se llegó a esta conclusión dado que los modelos no se desempeñaron tan bien como los modelos de HMMCUs creados con EP. La probabilidad del mejor modelo creado con EP, fue de 2.5209×10^{-75}, con 6 estados. El mejor modelo optimizado por BW, con 9 estados, tiene una una probabilidad de 1.22676×10^{-119}. Se nota claramente la ventaja que presenta la EP en la optimización de los parámetros de los HMMs.

Con respecto a los HMMCU evolucionados con EP, no requirieron intervención humana durante el proceso de diseño y entrenamiento. Los modelos se ajustaron al com-

portamiento del sistema analizado, a través de la secuencia de observaciones. Al contrario del algoritmo de Baum-Welch, que necesita la experiencia del usuario para crear un HMM general y deducir, en base a su conocimiento, los parámetros iniciales del modelo.

Podemos agregar, de acuerdo a nuestras pruebas, los siguientes puntos:

- La EP, junto con los GAs, realmente no necesitan que el usuario posea conocimientos previos de probabilidad y estadística para hacer evolucionar los HMMs. La EP construye prácticamente todo, crea y optimiza un HMMCU basado en los datos de las series de tiempo y en una función probabilística, con la cual se califican las observaciones de la secuencia.

- La EP necesita únicamente los datos de la serie de tiempo, una función de aptitud, y tiempo de procesamiento para el aprendizaje y ajuste de los parámetros.

- La EP construye el HMMCU con sus parámetros correctos.

- El algoritmo de *Baum-Welch* (BW) [Rabiner89] es un buen método para el proceso de aprendizaje, usando datos de la serie de tiempo, pero su problema consiste en que converge en un óptimo local.

- Otro problema que presenta BW, es que los usuarios necesitan un conocimiento previo acerca de la teoría y construcción de los HMMs en general.

- El rendimiento del algoritmo de BW depende en gran medida de los adecuados parámetros iniciales, para el proceso de reestimación de los mismos. Es decir, que si iniciamos con parámetros no adecuados, el método puede diverger.

Un último punto acerca de los HMMCU, es importante resaltar que los HMMCUs no pueden detectar anomalías de variaciones pequeñas en la secuencia de observación. Por ejemplo, si se variara el comportamiento del consumo de ancho de banda de red, de un día por otro; el modelo no sería capaz de detectar esta variación. Esto se debe a que para el sistema, la magnitud de ese consumo de ancho de banda de red, cae dentro de un comportamiento normal. Se necesita agregar al modelo otra variable para que pueda detectar la variación en un rango de valor diferente.

Lo comentado acerca de la debilidad que presentan los HMMCUs con respecto a variaciones en los valores de los datos, sin conocimiento temporal, se resuelve con la inclusión de temporalidad en la serie de datos. Para probar esta teoría se hizo evolucionar

un HMMCM de una secuencia de observación que además de uso de ancho de banda, incluyó la variable tiempo, en formato de día.hora. También se hizo evolucionar un HMMCU con la secuencia de uso de ancho de banda solamente. Al realizarse las pruebas el modelo HMMCM presentó mejores resultados del reconocimiento de anomalías que el HMMCU. Basta decir que la ventana de datos que menor reconocimientos tuvo del HMMCM, fue mayor en reconocimientos que la mejor ventana de datos del HMMCU.

Un último experimento consistió en hacer evolucionar un HMMCM, que además de involucrar al tiempo y el tráfico de red de entrada, se consideró como una tercera variable el tráfico de red de salida. Se observó que el modelo obtenido incrementó el porcentaje de aciertos en la detección de anomalías, pero nada más en las ventanas de tamaño 3, 4 y 5 datos. Es decir, el modelo remarcó la baja probabilidad que todas las ventanas de datos con anomalías, tenían con respecto al umbral fijado, pero no incrementó el porcentaje de aciertos en la ventana de 6 datos.

La ventaja que presenta un HMMCU con respecto a un HMMCM es en relación al tiempo de procesamiento. La evolución de un HMMCU es más rápida que en un HMMCM, dado que sólo procesa una sola variable del sistema. El mayor tiempo de procesamiento que involucra evolucionar un HMMCM, se ve compensado con la mayor certeza de detectar anomalías, con desviaciones mínimas, en el comportamiento del sistema cosa que un HMMCU no puede.

Otra situación que se observó es que entre más variables intervengan en el modelado del comportamiento de la dinámica de un sistema, el valor de la probabilidad de la secuencia observada disminuirá. Esto debido a que el valor de probabilidad de la secuencia, generado por el HMM, es el producto de observar cada vector de datos en los estados que forman al HMM.

Se puede concluir que la utilización de HMMs en la detección de anomalías es posible. Ya sean modelos de HMM univaridados o multivariados, entrenados con Baum-Welch o a través de Evolutivos o GAs, son una herramiento poderosa en la detección de anomalías en series de tiempo.

Capítulo 6

Conclusiones y Trabajo Futuro

En este capítulo se resumen los resultados obtenidos y se dan las conclusiones de los mismos. También se presentan algunas propuestas de actualización y ampliación sobre este trabajo de tesis, que podrían dar inicio a nuevos trabajos de investigación.

6.1. Conclusiones

El motivo que dio origen al presente trabajo de tesis fue la problemática de la seguridad informática, de cómo detectar un posible ataque o intento de intrusión a un sistema de cómputo. La idea era la de proponer una alternativa a los esquemas de detección de anomalías, creando modelos estocásticos probabilísticos, basados en los valores reales que genera un sistema en su desempeño normal.

Un problema central que se encontró fue ¿cómo diseñar y probar esta propuesta de detección?, la solución fue encontrada en los algoritmos genéticos y la programación evolutiva que ayudaron en la resolución de este problema.

Un punto importante en este trabajo fue el proporcionar un solución alternativa al problema número 3 de los HMMs, esto es, ajustar u optimizar los parámetros de un HMM dada la serie de observación. Con esta tesis se diseña y entrena un HMM con sólo la secuencia de observación, encontrando los mejores parámetros para el HMM.

También se puede decir que en este trabajo de Tesis la propuesta de utilizar HMMs, como modelos estocásticos probabilísticos y la Programación Evolutiva para evolucionarlos, y utilizar los modelos evolucionados en la detección de anomalías, fue como se esperaba. Ya que en las pruebas que se han hecho han resultado ser capaces de detectar anomalías en las

series de tiempo de prueba.

El sistema evolutivo es capaz de modelar HMMs con secuencias de valores de variables aleatorias continuas, sin necesidad de discretizar los valores originales. Otras herramientas, como Jahmm (Java Tool para HMMs) [Francois11], sólo trabaja con valores discretos.

Este trabajo a diferencia de muchos otros que utilizan HMMs y los optimizan usando GAs, para el reconocimiento de voz o secuencia de ADNs, este sistema se utiliza con secuencias multivariadas de valores reales, y se aplica para la detección de anomalías presentes en cualquier secuencia de observación.

El esquema que se implementó para optimizar los parámetros de los HMMs fue la de Programación Evolutiva (EP), similar a la propuesta por el Dr. Lawrence J. Fogel, creador de este enfoque.

El sistema de programación evolutiva trabaja con los HMMs de manera sencilla, incluso tiene valores por default para optimizar los HMMs con la secuencia que se le proporcione. Alternativamente, pueden especificarse valores para las probabilidades de los operadores evolutivos, indicar el número de generaciones, tamaño de la población, etc.

El sistema es capaz de modelar la dinámica de un sistema, utilizando HMMs de una variable (Univariado) o de un conjunto de ellas (Multivariado), y modelar el comportamiento probabilístico normal actual del sistema en cuestión. Para la actualización del modelo, sólo es necesario volver a ejecutar nuestro sistema con la nueva serie de observación, para crear y entrenar el modelo con las nuevas series de observación.

El modelo obtenido del sistema, se utiliza en la detección de anomalías en series de tiempo. Para HMMs Univariados, donde sólo observa una variable, es capaz de distinguir una anomalía cuando la serie presenta cambios estadísticamente significativos, de lo contrario tiende a considerar la observación dentro del intervalo o umbral probabilístico aceptable.

El tiempo de análisis y proceso para series de tiempo de una variable, es relativamente corto. No invierte mucho tiempo en el procesamiento y obtención del modelo del sistema. Sin embargo, el grado de certeza de detectar anomalías en una secuencia de observación que no presente desviaciones significativas, no es alta.

El tiempo de procesamiento para un HMM Univariado está relacionado de manera directa con la longitud de la secuencia de observación, con el número de estados que contengan los HMMs evaluados, el tamaño de la población y el número de generaciones a crear.

En una primera comparación entre los HMMs Univariados y un HMM-BW (Baum-Welch), optimizado con el Algoritmo de BW, nos proporcionó un mejor análisis de detecćon de las anomalías el modelo de HMMs generados con Programación Evolutiva (EP) que el optimizado con BW.

También concluimos que en la optimización de los parámetros de un HMM, a través del Algortimo de Baum-Welch, depende mucho de los valores iniciales con los cuales se comience la optimización del modelo. Si son correctos, el modelo se comportará de la manera esperada en la detección de anomalías, y sino el modelo no será capaz de detectar las anomalías.

Otra conclusión que se tiene de este trabajo, es del experimento que se realizó entre un HMM Univariado contra un HMM Multivariado de 2 variables. El modelo HMM multivariado presenta un mejor rendimiento en la detección de anomalías. Estos resultado se deben a que el HMM multivariado toma más referencias en las observaciones, debido a que tomó los valores del comportamiento de dos variables, y pudo detectar variaciones que no eran tan significativas. Al contrario del modelo de HMM Univariado, que al basarse únicamente en la observación de una variable y que al no presentar ésta variaciones significativas, no fue capaz de detectar las anomalías presentes en las secuencias de observaciones que se le dieron a analizar.

Ahora bien, el tiempo de procesamiento para un HMM Multivariado está también relacionado de manera directa con la longitud de la secuencia de observación, con el número de estados que contengan los HMMs evaluados, el tamaño de la población y el número de generaciones a crear. Además hay que considerar la dimensión del vector de observaciones, que a diferencia de los HMMs Univariados que sólo tienen una variable, en los HMM Multivariados cada observación es un conjunto de observaciones o datos.

Podemos concluir que nuestros modelos de HMMs generados con EP son capaces de distinguir entre comportamiento normal y comportamiento anormal. Nuestros experimentos muestran que los HMMs Univariados realizan bien su actividad, en un 75 % en la detección de anomalías.

Los modelos descritos en este trabajo fueron construidos con los datos de las series de tiempo del consumo de ancho de banda, para los Univariados. Para los HMMs Multivariados de 2 variables, además del ancho se banda de entrada, se tomó como referencia el día de la semana y la hora del consumo del ancho de banda de la red de entrada; y para los HMMCMs de 3 variables se tomaron, aparte de los dos variables anteriores, la variable

del consumo de ancho de banda de salida. Consideramos que nuestros modelos podrían emplearse en el centro de cómputo de la universidad, en un futuro cercano, o bien, ser utilizados en otros centros de cómputo.

6.2. Trabajo Futuro

El sistema actual de detección de anomalías basado en HMMs, no cuenta con la optimización de tiempo de procesamiento. Una sugerencia de mejora y actualización al sistema propuesto, consiste en optimizar el tiempo de procesamiento basado en secuencias cortas de observación, o la utilización de otras funciones alternativas evaluadoras de aptitud.

Una mejora sugerida al sistema es poder generar de manera automática matrices de covarianzas, con los valores de las correlaciones de las variables que intervengan. Para poder hacer evolucionar de manera completa los HMMs multivariados, el desempeño se estima que mejorará con la matriz completa de varianzas y correlaciones.

Una propuesta de mejora alternativa, que también pretende reducir el tiempo de procesamiento, es el de combinar nuestro esquema con otro método de optimización de parámetros de los HMMs. Por ejemplo, utilizando el algoritmo de Baum-Welch (u otras técnicas de optimización como swarm, partículas, bees, etc.) en la población inicial generada aleatoriamente, y a partir de ahí comenzar la evolución de los HMMs.

Otra propuesta consiste en utilizar Programación Evolutiva Paralela o Distribuida, donde las poblaciones, generaciones, operaciones evolutivas, así como la calificación de aptitud de los individuos, se ejecute en diferentes procesadores. Los procesadores pueden ubicarse físicamente en un equipo o en varios equipos distribuidos.

Se propone que se realice un análisis más detallado de la secuencia de observación o dotar de inteligencia al sistema detector, para que en base a la tendencia y características de la serie de tiempo u observaciones, el sistema sea capaz de decidir qué función de distribución modela mejor a la serie de tiempo. Logrando con esto que el sistema sea capaz de crear un modelo lo más cercano a la dinámica del sistema. Para sistemas que les interese más el modelo y la probabilidad de la secuencia dada, que podría utilizarse en búsquedas o coincidencia de patrones, en vez de anomalías.

También se propone que el sistema de detección de anomalías sea capaz de tomar decisiones de acuerdo a la serie de tiempo dada y la probabilidad obtenida o secuencia de estados generada. Los Procesos de Decisión de Markov (Markov Decision Process) toman

en cuenta la transición entre estados, y en cada transición se ejecuta una acción o decisión. Esto se podría aplicar a los esquemas de software utilizado en seguridad informática, que dada ciertas observaciones, secuencias de estados, ejecute un plan de refuerzo automático de la seguridad perimetral de una LAN.

Nuestro esquema de detección de anomalías, es sencillo de utilizar y es muy general. Esto es, el sistema es capaz de modelar series de tiempo de cualquier sistema que genere datos. Pudiéndose aplicar a diferentes sistemas de diferentes áreas como: medicina (frecuencias cardiacas, respiratorias, etc.), en sistemas eléctricos (lecturas o monitoreo de frecuencias), pronóstico de clima, tendencias en la bolsa de valores, etc. Todo sistema que genere series de observaciones, es posible modelarlo de manera probabilístico.

En estos días, el tratar de proporcionar una protección completa a un sistema de cómputo de ataques, es una tarea muy difícil; ya que las redes de computadoras por más protegidas que se encuentren, están propensas a ser atacadas y comprometidas alguna vez.

Esperamos que esta propuesta contribuya al desarrollo de modelos más exactos en la detección de anomalías usando HMMs u otros esquemas mixtos utilizando HMMs.

Referencias

[Al-ani07] Al-ani, T. et al. On-line automatic detection of human activity in home using wavelet and hidden markov models scilab toolkits. *IEEE Multi-conference on System and Control*, 16th IEEE International Conference on Control Applications:485–490, oct. 2007.

[Baker07] Baker, W. H. y Wallace, L. Is information security under control? *IEEE Computer Society, Security & Privacy*, 5(1):36–44, feb. 2007.

[Barrera08] Barrera, J. y Flores, J. Análisis de sistemas dinámicos utilizando herramientas de inteligencia artificial. *Posgrado en Eléctrica*, 1(1):1–95, mar. 2008.

[Bengio97] Bengio, Y. Markovian models for sequential markovian models for sequencial data. *Statistical Science*, 1997.

[Bhowmik07] Bhowmik, T. K., Parui, S. K., Kar, M., y Roy, U. *HMM Parameter Estimation with Genetic Algorithm for Handwritten Word Recognition*, tomo 4815/2007 de *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, nov. 2007. ISBN 978-3-540-77045-9. doi:10.1007/978-3-540-77046-6.

[Bhuriyakorn08] Bhuriyakorn, P., Punyabukkana, P., y Suchato, A. A genetic algorithm-aided hidden markov model topology estimation for phoneme recognition of thai continuous speech. *En Ninth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing*, págs. 475–480. IEEE, 2008.

[Brewer06] Brewer, N. et al. Using coupled hidden markov models to model suspect interactions in digital forensic analysis. *Integrating AI and Data Min-*

ing, 2006. AIDM '06. International Workshop, págs. 58–64, 2006. doi: 10.1109/AIDM.2006.16.

[Cappé05] Cappé, O., Moulines, E., y Rydén, T. *Inference in Hidden Markov Models*, tomo 1 de *Springer Series in Statistics*. Springer, 1ª edón., 2005. ISBN 0-387-40264-0.

[Chau97] Chau, C., Kwong, S., Diu, C., et al. Optimization of hmm by a genetic algorithm. *Acoustics, Speech, and Signal Processing, IEEE International Conference on*, 3:1727, 1997. ISSN 0-8186-7919-0. doi:ICASSP.1997.598857.

[Cheshomi10] Cheshomi, S. et al. Hmm training by a hybrid of chaos optimization and baum-welch algorithms for discrete speech recognition. *Digital Content, Multimedia Technology and its Applications (IDC), 2010 6th International Conference*, págs. 337–341, 2010.

[Chien08] Chien, J.-T. y Liao, C.-P. Maximum confidence hidden markov modeling for face recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions*, 30:606–616, 2008. doi:10.1109/TPAMI.2007.70715.

[Chunyue06] Chunyue, Z., Yun, L., y Hongke, Z. A pattern matching based network intrusion detection system. *IEEE Control, Automotion, Robotics & Vision*, ICARCV 2006(9th International Conference):1–4, dic. 2006. ISSN 1-4244-0341-3.

[Dai94] Dai, J. Hybrid approach to speech recognition using hidden markov models and markov chains. *Vision, Image and Signal Processing, IEE Proceedings*, 141:273–279, 1994. doi:10.1049/ip-vis:19941321.

[Dan10] Dan, S. y YanLing, S. Detection of network intrusion based on a hmm model. *Multimedia and Information Technology (MMIT), 2010 Second International Conference*, 1:286–289, 2010. doi:10.1109/MMIT.2010.48.

[Daniel83] Daniel, W. y Terrel, J. *Business Statics*, tomo 1. Houghton Mifflin Company, Boston, USA., 3ª edón., 1983.

[Dasgupta01] Dasgupta, D. y Brian, H. Mobile security agents for network traffic analysis. *IEEE Transactions on Power Systems*, 2(332-340), 2001.

[Deng06] Deng, C. y Zheng, P. A new hidden markov model with application to classification. *Intelligent Control and Automation, 2006. WCICA 2006. The Sixth World Congress*, 2:5882–5886, 2006. doi: 10.1109/WCICA.2006.1714206.

[Ephraim02] Ephraim, Y. y Merhav, N. Hidden markov processes. *IEEE TRANSACTIONS ON INFORMATION THEORY*, 48(6):1518–1568, jun. 2002.

[Esquivel04] Esquivel, R. y Flores, J. J. *Seguridad en Redes UNIX: Un Esquema Multinivel Basado en Software Libre*, tomo 1 de *UMSNH*. UMSNH, Morelia, Mich., 1ª ed$^{\underline{\text{ón}}}$., nov. 2004. ISBN 970-703-291-X.

[Foo04] Foo, S. Y. y Arradondo, M. Mobile agents for computer intrusion detection. *IEEE System Theory, 2004*, 0(0):517–521, sep. 2004. ISSN 0094-2898.

[Francois11] Francois, J.-M. Jahmm tool, Enero 2011.
URL `http://www.run.montefiore.ulg.ac.be/ francois/software/jahmm/`

[Garcia06] Garcia, J. M., Navarrete, T., y Corona Orozco, C. Workload hidden markov model for anomaly detection. *SECRYPT 2006*, Proceedings of the International Conference on Security and Cryptography:56–59, August 2006.

[Ghosh98] Ghosh, A., Wanken, J., y Charron, F. Detecting anomalous and unknown intrusions against programs. *Computer Security Applications Conference*, págs. 259–267, dic. 1998. ISSN 0-8186-8789-4.

[Green07] Green, I., Raz, T., y Zviran, M. Analysis of active intrusion prevention data for predicting hostile activity in computer networks. *Communications of the ACM*, 50(4):63–68, abr. 2007.

[He97] He, J. Performance and manageability design in an enterprise network security system. *Enterprise Networking Mini-Conference. ENM-97*, IEEE, ICC-97:127–134, jun. 1997. ISSN 0-7803-4112-0.

[hyeonLee07] hyeon Lee, D., young Kim, D., y il Jung, J. *Mobile Agent Based Intrusion Detection System Adopting Hidden Markov Model*, tomo 4706/2007

de *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, ago. 2007. ISBN 978-3-540-74475-7. doi:10.1007/978-3-540-74477-1_12.

[Islam05] Islam, M. H. y Jamil, M. Taxonomy of statistical based anomaly detection techniques for intrusion detection. págs. 270–276, 2005.

[Jacob01] Jacob, C. *Illustrating Evolutionary Computation with Mathematica*, tomo 1. Academic Press, 525 B Street, San Diego Ca., 1ª ed$^{\text{ón}}$., 2001. ISBN 1-55860-637-8.

[Jemili07] Jemili, F., Zaghdoud, M., y Ben Ahmed, M. A framework for an adaptive intrusion detection system using bayesian network. *Intelligence and Security Informatics, IEEE*, 2007.

[Jha01] Jha, S., Tan, K., y Maxion, R. A. Markov chains, classifiers, and intrusion detection. *Computer Security Foundations Workshop*, págs. 206–219, 2001.

[Juang85] Juang, B.-H. y Rabiner, L. Mixture autoregressive hidden markov models for speech signals. *Acoustics, Speech and Signal Processing, IEEE Transactions*, 3(6):1404–1413, 1985. doi:10.1109/TASSP.1985.1164727.

[Kannadiga05] Kannadiga, P., Zulkernine, M., y Ahamed, S. I. Towards an intrusion detection system for pervasive computing environments. *IEEE, International Conference on Information Technology*, 2(ITCC'05):227–282, abr. 2005. ISSN 0-7695-2315-3.

[Khanna06] Khanna, R. y Liu, H. System approach to intrusion detection using hidden markov model. *IWCMC'06, ACM*, July 2006.

[Khreich09] Khreich, W., Granger, E., et al. Combining hidden markov models for improved anomaly detection. *Communications, 2009. ICC '09.*, págs. 1–6, jun. 2009. ISSN 1938-1883. doi:10.1109/ICC.2009.5198832.

[Kirshner05] Kirshner, S. *Modeling of Multivariate Time Series Using Hidden Markov Models*. Tesis Doctoral, University of California, Irvine, 2005.

[Korayem07] Korayem, M., Badr, A., y Farag, I. Optimizing hidden markov models using genetic algorithms and artificial immune systems. *Computing and Information Systems*, 11(2), 2007.

[Kotsalis06] Kotsalis, G. et al. A model reduction algorithm for hidden markov models. *Decision and Control, 2006 45th IEEE Conference*, págs. 3424–3429, 2006. doi:10.1109/CDC.2006.377011.

[Kurimo92] Kurimo, M. y Torkkola, K. Training continuous density hidden markov models in association with self-organizing maps and lvq. *Proceedings of the 1992 IEEE-SP Workshop*, 1:174–183, ago. 1992. ISSN 0-7803-0557-4. doi:10.1109/NNSP.1992.253695.

[Lane03] Lane, T. y Brodley, C. E. An empirical study of two approaches to sequence learning for anomaly detection. *Machine Learning*, 51(1):73–107, abr. 2003. ISSN 0885-6125. doi:10.1023/A:1021830128811.

[Lang04] Lang, B., Liu, J., y Zheng, J. The research on automated intrusion response system based on mobile agents. *Computer Supported Cooperative Work in Design*, 1(0):344–347, mayo 2004. ISSN 0-7803-7941-1.

[Li04] Li, C., Li, P., y Song, H.-Z. Process trends analysis via wavelet-domain hidden markov models. *IEEE Machine Learning and Cybernetics*, 1(0):372–377, ago. 2004. ISSN 0-7803-8403-2.

[Li09] Li, Y., Wang, R., Xu, J., Yang, G., y Zhao, B. Intrusion detection method based on fuzzy hidden markov model. *En FSKD '09: Proceedings of the 2009 Sixth International Conference on Fuzzy Systems and Knowledge Discovery*, págs. 470–474. IEEE Computer Society, Washington, DC, USA, 2009. ISBN 978-0-7695-3735-1. doi: http://dx.doi.org/10.1109/FSKD.2009.79.

[Link10] Link, W. A. y Barker, R. J. *Bayesian Inference with Ecological applications*. ELSEVIER. Academic Press, 1ª ed$^{\underline{\text{ón}}}$., 2010. ISBN 978-0-12-374854-6.

[López05] López, M.-E. et al. A navigation system for assistant robots using visually augmented pomdps. *Autonomous Robots*, 1(19):67–87, 2005. ISSN 0929-5593. doi:10.1007/s10514-005-0607-3.

[Martínez09] Martínez, M. La influencia de darwin en el pensamiento científico contemporáneo. *La Ciencia y el Hombre*, XXII(3), dic. 2009. URL `http://www.uv.mx/cienciahombre/revistae/vol22num3/articulos/darwin/index.ht`

[Miners05] Miners, B. y Basir, O. Dynamic facial expression recognition using fuzzy hidden markov models. *Systems, Man and Cybernetics, 2005 IEEE International Conference*, 2:1417–1422, 2005. doi: 10.1109/ICSMC.2005.1571345.

[Missaoui08] Missaoui, O. y Frigui, H. Optimal feature weighting for the continuous hmm. *Pattern Recognition, ICPR 2008. 19th International Conference*, (1):1–4, dic. 2008. ISSN 978-1-4244-2174-9. doi: 10.1109/ICPR.2008.4761076.

[Mukkamala07] Mukkamala, S. et al. Hybrid multi-agent framework for detection of stealthy probes. *Applied Soft Computing*, 7(3):631–641, jun. 2007. ISSN 1568-4946.

[Navarrete04] Navarrete, T. y Garcia, J. M. Detección de anomalías en la carga de un procesador, utilizando modelos ocutlos de markov. *Instituto Tecnológico de Morelia*, 1, 2004.

[Nicholl08] Nicholl, P., Amira, A., Bouchaffra, D., y Perrot, R. H. A statistical multiresolution approach for face recognition using structural hidden markov models. *EURASIP Journal on Advances in Signal Processing, ACM*, 2008:13, 2008.

[Ogawa06] Ogawa, T. y Kobayashi, T. Genetic algorithm based optimization of partly-hidden markov model structure using discriminative criterion. *IEICE TRANS. INF. & SYS.*, E89-D(3), March 2006.

[Oudelha10] Oudelha, M. y Ainon, R. Hmm parameters estimation using hybrid baum-welch genetic algorithm. *Information Technology (ITSim), 2010 International Symposium*, 2:542–545, 2010. doi:10.1109/ITSIM.2010.5561388.

[Pachter01] Pachter, L. et al. Applications of generalized pair hidden markov models

to alignment and gene finding problems. *RECOMB 2001, Research in Computational Molecular Biology*, págs. 241–248, 2001.

[Rabiner86] Rabiner, L. R. y Juang, B. H. An introduction to hidden markov models. *IEEE ASSP Magazine*, 1986.

[Rabiner89] Rabiner, L. R. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2), February 1989.

[Ragsdale00] Ragsdale, D., Carver, C., Humphries, J., y Pooch, U. Adaptation techniques for intrusion detection and intrusion response systems. *En In Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics at*, págs. 2344–2349. Society Press, 2000.

[Seymore99] Seymore, K., Mccallum, A., y Rosenfeld, R. Learning hidden markov model structure for information extraction. *En In AAAI 99 Workshop on Machine Learning for Information Extraction*, págs. 37–42. 1999.

[Shivaraj08] Shivaraj, G., Song, M., y Shetty, S. A hidden markov model based approach to detect rogue access points. *IEEE Military Communications Conference*, págs. 1–7, nov. 2008. ISSN 978-1-4244-2676-8. doi: 10.1109/MILCOM.2008.4753358.

[Shlens09] Shlens, J. A tutorial on principal component analysis. *Systems Neurobiology Laboratory, Salk Insitute for Biological Studies*, págs. 1–12, abr. 2009.

[Shyu07] Shyu, M.-L. et al. Network intrusion detection through adaptive subeigenspace modeling in multiagent systems. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 2(3):9–37, sep. 2007. ISSN 1556-4665.

[Singh07] Singh, S., Donat, W., Pattipati, K., y Willet, P. Anomaly detection via feature-aided tracking and hidden markov model. *Aerospace Conference, IEEE*, págs. 1–18, March 2007.

[Slimane96] Slimane, M., Venturini, G., et al. Optimizing hidden markov models with a genetic algorithm. *En* ACM, ed., *Artificial Evolution*, tomo 1063 de *Lecture Notes in Computer Science*, págs. 384–396. Springer-Verlag, London, UK, 1996. ISBN 3-540-61108-8. doi:10.1007/3-540-61108-8_52.

[Spafford08] Spafford, E. James p. anderson: An information security pioneer. *IEEE Security and Privacy*, 6:9, 2008. ISSN 1540-7993. doi:10.1109/MSP.2008.15.

[Srivastava08] Srivastava, A., Kundu, A., Sural, S., y Majumdar, A. K. Credit card fraud detection using hidden markov model. *IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING*, 5(1):37–48, ene. 2008.

[Sugaya09] Sugaya, M., Ohno, Y., y Nakajima, T. Lightweight anomaly detection system with hmm resource modeling. *International Journal of Security and Its Applications*, 3(3):35–53, jul. 2009.

[Taylor98] Taylor, H. M. y Karlin, S. *An Introduction to Stochastic Modeling*. Academic Press, 3ª edón., 1998. ISBN 0-12-684887-4.

[Toure94] Toure, M. An interdisciplinary approach for adding knowledge to computer security systems. *Security Technology*, IEEE 28th Annual International Carnahan Conference on:158–168, oct. 1994. ISSN 0-7803-1479-4.

[Van-Le99] Van-Le, T. et al. Fuzzy evolutionary programming for hidden markov modelling in speaker identification. *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress*, 2, 1999. doi:10.1109/CEC.1999.782505.

[Vanluyten07] Vanluyten, B., Willems, J., y Moor, B. D. A new approach for the identification of hidden markov models. *Decision and Control, 2007 46th IEEE Conference*, págs. 4901–4905, 2007. doi:10.1109/CDC.2007.4434912.

[Wadron92] Wadron, T. W. *Network Security in the 90's Issues and Solutions for Managers*, tomo 1 de *Wiley Professional Computing*. John Wiley & Sons, Inc., Canada, 1ª edón., 1992. ISBN 0-471-54777-8.

[Wang10] Wang, P. et al. Survey on hmm based anomaly intrusion detection using system calls. *Computer Science and Education (ICCSE), 2010 5th International Conference*, págs. 102–105, 2010. doi:10.1109/ICCSE.2010.5593839.

[Warrender99] Warrender, C., Forrest, S., y Pearlmutter, B. Detecting intrusions using system calls: Alternative data models. *En Security and Privacy, 1999. Proceedings of the 1999 IEEE Symposium on*, págs. 133–145. IEEE, May 1999.

[Welch03] Welch, L. The shannon theory hidden markov models and the baum-welch algorithm. *En IEEE Information Theory Society Newsletter*, tomo 53 de *IEEE*. IEEE, dic. 2003.

[Wolfram09] Wolfram, S. http://www.wolfram.com/products/ mathematica/index.html, Octuber 2009.
URL `http://www.wolfram.com/ products/mathematica/ index.html`

[Won04] Won, K.-J. et al. Training hmm structure with genetic algorithm for biological sequence analysis. *Life Sciences, Bioinformatics*, 20(18):3613–3619, ago. 2004. doi:10.1093/bioinformatics/bth454.

[Won05] Won, K.-J., Hamelryck, T., Prugel-Bennett, A., y Krogh, A. Evolving hidden markov models for protein secondary structure prediction. *Evolutionary Computation, IEEE*, 1:1–18, September 2005.

[Xiuqing10] Xiuqing, C., Yongping, Z., y Jiutao, T. Hmm-based integration of multiple models for intrusion detection. *Advanced Computer Theory and Engineering (ICACTE), 2010 3rd International Conference*, 2:V2–137–V2–140, 2010. doi:10.1109/ICACTE.2010.5579109.

[Yang08a] Yang, F., Zhang, C., y Bai, G. A novel genetic algorithm on tabu search for hmm optimization. *Fourth International Conference on Natural Computation, IEEE*, 2008.

[Yang08b] Yang, F., Zhang, C., y Sun, T. Comparison of particle swarm optimization and genetic algorithm for hmm training. *Pattern Recognition, 2008*, ICPR 2008. 19th International Conference(0):1–4, dic. 2008. ISSN 1051-4651. doi:10.1109/ICPR.2008.4761282.

[Yasami07] Yasami, Y., Farahmand, M., y Zargari, V. An arp-based anomaly detection

algorithm using hidden markov model in enterprise networks. *En ICSNC*, ICSNC 2007, pág. 69. IEEE Computer Society, ago. 2007.

[Ye10] Ye, X., Li, J., y Li, Y. An anomaly detection system based on hide markov model for manet. *Wireless Communications Networking and Mobile Computing (WiCOM), 2010 6th International Conference*, págs. 1–4, 2010.

[Zaki10] Zaki, M. J., Carothers, C. D., y Szymanski, B. K. Vogue: A variable order hidden markov model with duration based on frequent sequence mining. *ACM Trans. Knowl. Discov. Data*, 4(1):1–31, 2010. ISSN 1556-4681. doi: http://doi.acm.org/10.1145/1644873.1644878.

[Zeng09] Zeng, F., Yin, K., Chen, M., y Wang, X. A new anomaly detection method based on rough set reduction and hmm. *Computer and Information Science, ACIS International Conference*, págs. 285–289, 2009. doi: 10.1109/ICIS.2009.140.

[Zhang06] Zhang, D. y Leckie, C. An evaluation technique for network intrusion detection systems. *ACM International Conference Proceeding Series*, INFOSCALE'06(52), mayo 2006. ISSN 1-59593-428-6.

[Zhang07] Zhang, X., Wang, Y., y Zhao, Z. A hybrid speech recognition training method for hmm based on genetic algorithm and baum welch algorithm. *IEEE Innovative Computing, Information and Control*, 0:572–572, 2007. ISSN 0-7695-2882-1. doi: http://doi.ieeecomputersociety.org/10.1109/ICICIC.2007.33.

[Zhao09] Zhao, J., Huang, H., Tian, S., y Zhao, X. Applications of hmm in protocol anomaly detection. *IEEE Computer Society*, cso, 2:347–349, abr. 2009. ISSN 978-0-7695-3605-7.

[Zhecheva07] Zhecheva, V. y Nikolova, E. Decoding efficiency of the map and the max-log map algorithm as a strategy in anomaly-based intrusion detection systems. *Proceedings of the 2007 international conference on Computer systems and technologies*, 1:1–6, 2007. doi:doi.acm.org/10.1145/1330598.1330633.

[Zhicai10] Zhicai, S. y Yongxiang, X. A novel hidden markov model for detecting complicate network attacks. *Wireless Communications, Networking and Information Security (WCNIS), 2010 IEEE International Conference*, págs. 312–315, ago. 2010.

[Zihui10] Zihui, C. y Xueyun, J. An efficient intrusion detection approach based on hidden markov model and rough set. *Machine Vision and Human-Machine Interface (MVHI), 2010 International Conference*, págs. 476–479, 2010. doi:10.1109/MVHI.2010.199.