Departamento de Arquitectura y Tecnología de Computadores

E.T.S. Ingeniería Informática y de Telecomunicación

Universidad de Granada

TESIS DOCTORAL

Sistemas de Detección de Intrusos con Mapas Autorganizativos Probabilísticos y Optimización Multiobjetivo

Autor
Emiro De la Hoz Franco

Directores
Dr. Andrés Ortiz García
Dr. Julio Ortega Lopera

Editor: Universidad de Granada. Tesis Doctorales
Autor: Emiro de la Hoz Franco
ISBN: 978-84-9125-824-7
URI: http://hdl.handle.net/10481/43551

El doctorando **Emiro De la Hoz Franco** y los directores de la tesis **Dr. Andrés Ortiz García y Dr. Julio Ortega Lopera** Garantizamos, al firmar esta tesis doctoral, que el trabajo ha sido realizado por el doctorando bajo la dirección de los directores de la tesis y hasta donde nuestro conocimiento alcanza, en la realización del trabajo, se han respetado los derechos de otros autores a ser citados, cuando se han utilizado sus resultados o publicaciones.

Granada, Noviembre 13 de 2015

Dr. Andrés Ortiz García
Director de la Tesis

Fdo.: Andrés Ortiz García

Dr. Julio Ortega Lopera
Director de la Tesis

Fdo.:

Emiro De la Hoz Franco
Doctorando

Fdo.:

Agradecimientos

Este trabajo pudo realizarse gracias al apoyo que he encontrado en diferentes personas e instituciones, a las cuales les extiendo mi más afectuoso agradecimiento.

A la Universidad de la Costa – CUC, por haberme apoyado económicamente en la consecución de tan anhelado reto. En particular a sus fundadores, directivos institucionales y directivos del programa al cual me encuentro adscrito.

A la Universidad de Granada – UGR y muy especialmente al director del doctorado en Tecnologías de la Información y Comunicación – TIC, PhD Hector Pomares, a la planta docente de dicho doctorado y del master en Ingeniería de Computadores y Redes.

Al Ministerio de Economía y Competitividad de España y de los Fondos Feder, que han financiado el proyecto TIN2012-32039, en el cual se enmarca este trabajo.

A mis directores de tesis: PhD Andrés Ortiz García y PhD Julio Ortega Lopera, a quienes considero más que directores, modelos de vida a seguir, por las excepcionales calidades académicas y humanas que les identifican. Ambos siempre estuvieron prestos a apoyar mi desarrollo investigativo y brindaron todas las facilidades administrativas para que este proyecto llegase a feliz puerto.

A mis padres por darme la vida y motivarme siempre a alcanzar mis metas.

A mi preciosa esposa, María Rosario Gil Suárez, por su permanente apoyo y una indeclinable actitud de motivación hacia mí. De igual forma a mis amados hijos, Juan David De la Hoz y Alejandro De la Hoz, por su comprensión y paciencia en todos esos años que estuve físicamente lejos de ellos, para poder dedicarme a alcanzar este propósito.

A todos muchísimas gracias.

A Charo, Juancho y Alejo.

Indice

Lista de Tablas

Lista de Figuras

PRESENTACIÓN

Disponer de procedimientos eficientes de clasificación de patrones es esencial en muchas aplicaciones de gran interés socio-económico. Una de ellas es el diseño de Sistemas de Detección de Intrusos en sistemas de cómputo. La eficiencia de los procesos de clasificación esta ligada a diversos factores, uno de ellos es la ejecución previa de procesos de selección de características sobre el *dataset*. Lo cual no sólo mejora la precisión de la clasificación, también mejora la capacidad de generalización en el caso de clasificadores supervisados, o contrarresta el sesgo hacia los números más bajos o más altos de características que presentan algunos de los métodos utilizados para validar la agrupación/clasificación en el caso de los clasificadores no supervisados.

Para desarrollar el objeto de estudio de esta investigación (clasificación mediante redes neuronales y optimización multiobjetivo para la detección de intrusos), previamente ha sido necesario resolver un problema de selección de características. Con el fin de indentificar la elección de un subconjunto adecuado de características, que permita la reducción de la dimensionalidad en el dataset, lo que contribuye a disminuir la complejidad computacional de la clasificación, mejorando el rendimiento del clasificador y evitando características redundantes o irrelevantes. Aunque la selección de características se puede definir formalmente como un problema de optimización con un solo objetivo, es decir, la precisión (*accuracy*) de la clasificación, obtenida usando el subconjunto de característica seleccionada, en los últimos años, se han propuesto algunos enfoques multi-objetivo para este problema. Aquí se rigue esta línea. Así, la principal contribución de este trabajo es un enfoque multi-objetivo para la selección de características y su aplicación a un procedimiento de agrupamiento no supervisado basado en Mapas Auto-Organizados Jerárquicos Crecientes (GHSOMs) que incluye un nuevo método para el etiquetado de unidades y la determinación eficiente de las unidades ganadoras. En el problema de la detección de anomalías de red aquí considerado, este enfoque multi-objetivo hace posible no sólo diferenciar entre el tráfico normal y anómalo (biclase), también lo hace entre las diferentes anomalías (multiclase). La eficiencia de la propuesta se ha evaluado mediante el uso del dataset DARPA/NSL-KDD que contiene características extraídas y ataques etiquetados de alrededor de 2 millones de conexiones. Los conjuntos de características seleccionadas que se han calculado en los experimentos proporcionan tasas de detección de hasta un 99,8% con el tráfico normal y hasta un 99,6% con tráfico anómalo, así como valores de precisión (*accuracy*) de hasta el 99,12%.

Este trabajo se ha realizado en el marco del proyecto TIN2012-32039, financiado por el Ministerio de Economía y Competitividad de España y de los Fondos Feder. Parte del trabajo realizado en esta tesis ha derivado en las publicaciones que se relacionan a continuación.

Principales publicaciones

Feature selection by multi-objective optimisation: Application to network anomaly detection by hierarchical self-organising maps. KBS - Knowledge-Based Systems (Q1). Nederland (Holanda). Volumen 71., 2014, pp 322 - 338. (Autores: Emiro De la Hoz Franco, Eduardo De la Hoz Correa, Andrés Ortiz García, Julio Ortega Lopera y Antonio Martinez Alvarez). ISSN 0950-7051. http://www.sciencedirect.com/science/article/pii/S0950705114002950

Implementation of an Intrusion Detection System based on self organizing map. JATIT - Journal of Theoretical and Applied Information Technology (Q4). Volumen 71, fasículo 3, pp 324-334. (Autores: Emiro De la Hoz Franco, Eduardo De la Hoz Correa, Andrés Ortiz García, Julio Ortega Lopera, Fabio Mendoza Palechor). ISSN 1992-8645.
http://www.jatit.org/volumes/Vol71No3/2Vol71No3.pdf

Lecture Notes in Computer Science: "Network Anomaly Detection with Bayesian Self-Organizing Maps". Book title: Advances in Computational Intelligence. Book subtitle: 12th International Work-Conference on Artificial Neural Networks, IWANN 2013, Puerto de la Cruz, Tenerife, Spain, June 12-14, 2013, Proceedings, Part I. Springer Berlin Heidelberg. Volume 7902, 2013, pp 530-537. (Autores: Emiro De la Hoz Franco, Eduardo De la Hoz Correa, Andrés Ortiz García, Julio Ortega Lopera y Alberto Prieto Espinosa). ISBN 978-3-642-38678-7.
http://link.springer.com/chapter/10.1007%2F978-3-642-38679-4_53#page-1

Modelo de detección de intrusiones en Sistemas de Red, realizando selección de características con FDR y entrenamiento y clasificación con SOM. Revista de la Facultad de Ingenierías. Universidad de la Costa – CUC. Barranquilla-Colombia. Educosta – Editorial Universitaria de la Costa. Volumen 8. Páginas 85-116. (Autores: Emiro De la Hoz Franco, Eduardo De la Hoz Correa, Andrés Ortiz García y Julio Ortega Lopera). 2012. ISSN 0122-6517.

Participación en otras publicaciones relacionadas

PCA filtering and Probabilistic SOM for Network Intrusion Detection. Neurocomputing. (Q2). 2015. (Autores: Eduardo De la Hoz Correa, Emiro De la Hoz Franco, Andrés Ortiz García, Julio Ortega Lopera, Beatriz Prieto). ISSN 0925-2312.
http://www.sciencedirect.com/science/article/pii/S0925231215002982

Lecture Notes in Computer Science: "Network Anomaly Classification by Support Vector Classifiers Ensemble and Non-linear Projection Techniques". Book title: Hybrid Artificial Intelligent Systems. Book subtitle: 8th International Conference, HAIS 2013, Salamanca, Spain, September 11-13, 2013. Proceedings. Springer Berlin Heidelberg. Volume 8073, 2013, pp 103-111. (Autores: Eduardo De la Hoz Correa, Emiro De la Hoz Franco, Andrés Ortiz García y Julio Ortega Lopera). ISBN 978-3-642-40845-8.
http://link.springer.com/chapter/10.1007%2F978-3-642-40846-5_11

Implementación de GHSOM (Growing Hierarchical Organizing Map) en sistemas de detección de intrusos. Revista de la Facultad de Ingenierías. Universidad de la Costa – CUC. Educosta – Editorial Universitaria de la Costa. Volumen 8. Páginas 117-148. (Autores: Eduardo De la Hoz Correa, Emiro De la Hoz Franco, Andrés Ortiz García y Julio Ortega Lopera). 2012. ISSN 0122-6517.

Feature selection, learning metrics and dimension reduction in training and classification processes in intrusion detection systems. JATIT - Journal of Theoretical And Applied Information Technology (Q4). Pakistán. Volumen 71, fasículo 3, pp 324-334. (Autores: Fabio Mendoza, Emiro De la Hoz Franco, Alexis De la Hoz, Paola Ariza). ISSN 1992-8645. Aceptado – Aún no publicado.

Organización de la memoria

Este documento se ha organizado como se indica a continuación:

En el Capítulo 1. (Introducción y fundamentos relativos a los sistemas de detección de intrusiones) Se recopilar la base documental que fundamenta la presente investigación, por ello en primera instancia se aborda una introducción relativa a los ataques informáticos, su categorización y la identificación de los mecanismos de defensa tradicionales contra tales ataques informáticos (sección 1.1). En segunda instancia se analizan las diferentes propuestas de estandarización de los Sistemas de Detección de Intrusos – IDS, sus componentes constitutivos, la clasificación de los IDS de acuerdo a diferentes criterios y las métricas de desempeño utilizadas para evaluar los IDS (sección 1.2). Para una mayor comprensión de lo anterior, se hace una descripción de los registros de conexiones de red, se describen los atributos o características que constituyen dichas conexiones (sección 1.3), se documentan los diferentes datasets empleados como soporte en IDS (sección 1.4), también se documentaron las diferentes fases del proceso de simulación en detección de intrusiones (sección 1.5), identificando los fundamentos relativos a las técnicas de extracción y selección de características (sección 1.6), en las que se basan diferentes propuestas de investigación que se han tomado como referentes. El capítulo culmina con unas conclusiones.

En el Capítulo 2. (Mapas Auto-organizativos Jerárquicos y Algoritmos Evolutivos Multiobjetivo) Se abordan los siguientes temas: en primera instancia los fundamentos de las redes neuronales artificiales - ANN (sección 2.2). A continuación se fundamenta el SOM (sección 2.3), en cuanto a su algoritmo de aprendizaje, campos de aplicación, arquitecturas basadas en SOM y sus inconvenientes. Posteriormente, se fundamenta el GHSOM (sección 2.4), en cuanto a sus principios funcionales, su algoritmo de entrenamiento y la forma como éste soluciona los problemas que presenta SOM. Consecuentemente se fundamenta la optimización como estrategia para definir problemas, mediante modelos y para resolverlos mediante algoritmos (sección 2.5). Se aborda además el tema de los algoritmos de optimización multiobjetivo basados en computación evolutiva (MOEA), utilizados para encontrar múltiples soluciones óptimas de Pareto en una única ejecución de la simulación (sección 2.6). Sin embargo, los MOEA pese a su utilidad han evidenciado una serie de inconvenientes, posteriormente descritos y que son subsanados por el algoritmo NSGA-II (Non-dominated Sorting Genetic Algorithm-II) el cual se describe en (sección 2.7). El capítulo culmina con las correspondientes conclusiones.

En el Capítulo 3. (Clasificación de ataques con GHSOM y Optimización Multiobjetivo) Se hace una descripción del coeficiente de Jaccard como medida de la asimetría de la información, el cual se integra en esta propuesta, con el algoritmo NSGA-II aplicado al proceso de selección de características (sección 3.2), luego se presenta la propuesta de optimización multiobjetivo basada en NSGA-II (sección 3.3), a continuación se detalla la propuesta de clasificación basada en GHSOM, la cual integra una novedosa técnica de re-etiquetado probabilístico (sección 3.4). El capítulo culmina con las correspondientes conclusiones.

En el Capítulo 4. (Estudio experimental aplicado a la detección de intrusos) Se presentan los resultados experimentales obtenidos mediante la aplicación del clasificador GHSOM, descrito en capítulos anteriores, para el dataset NSL-KDD [110]. Inicialmente se hace un análisis de la razón por la cual se selecciona el dataset NSL_KDD (sección 4.2) y la configuración inicial de los escenarios de experimentación (sección 4.3), la importancia del pre-procesamiento (sección 4.4)

y normalización (sección 4.5) de los datos para asegurar la homogeneidad en la contribución de las características a la medida de la distancia. A continuación se detallan los resultados experimentales entrenando el GHSOM tanto con la totalidad del conjunto de características (sección 4.6) como con una reducción previa del conjunto de características (sección 4.7). A partir de los resultados obtenidos con los escenarios de experimentación, se realiza una evaluación de las prestaciones del NSGA-II considerando la convergencia del algoritmo y el uso del índice de Jaccard (sección 4.8). Posteriormente se analiza la efectividad computacional de la propuesta, en relación al nivel de discriminación de las características para todas las clases para una cantidad aceptable de tiempo de cómputo (sección 4.9). De forma complementaria, y dado que los resultados de la clasificación pueden variar entre las diferentes corridas, se efectuaron pruebas de significación estadística, con el fin de analizar si los valores medios obtenidos mediante la aplicación de varios experimentos, son diferentes. Así a partir de los resultados obtenidos se elaboró un ANOVA (sección 4.10). El capítulo termina con las conclusiones correspondientes.

En el Capítulo 5. (Conclusiones y principales aportaciones) Se describen las conclusiones y aportaciones a las cuales se llega producto del análisis de los resultados de los experimentos descritos en el capítulo anterior. El capítulo finaliza con la identificación de algunos trabajos futuros, coherentes con la propuesta aquí enunciada.

La memoria concluye con el listado de referencias consultadas.

CAPÍTULO 1. INTRODUCCIÓN Y FUNDAMENTOS RELATIVOS A LOS SISTEMAS DE DETECCIÓN DE INTRUSIONES

1.1 INTRODUCCIÓN

Desde el posicionamiento de Internet como medio de interconexión global, se han venido incrementando los incidentes que intentan vulnerar la seguridad de los sistemas computacionales. Entendiendo por incidentes o amenazas: los errores u omisiones (referidos a las actuaciones que realizan los usuarios al utilizar los sistemas y que puedan comprometer la integridad de la información que maneja la organización), las intrusiones o ataques (entendidas como intentos de acceso con intenciones claramente fraudulentas o con el único fin de que el intruso pruebe sus habilidades), los accidentes y desastres (eventualidades cotidianas físicas o lógicas que puedan tener efectos negativos sobre los sistemas de información) y las amenazas a la privacidad de las personas (referida a la recolección de información personal con fines ilegales en detrimento del buen nombre de las personas). La presente investigación centra su estudio en la detección de incidentes concernientes a intrusiones o ataques informáticos.

En este capítulo se pretende recopilar la base documental que fundamenta la presente investigación, por ello en primera instancia se aborda una introducción relativa a los ataques informáticos, su categorización y la identificación de los mecanismos de defensa tradicionales contra tales ataques informáticos (sección 1.1). En segunda instancia se analizan las diferentes propuestas de estandarización de los Sistemas de Detección de Intrusos – IDS, sus componentes constitutivos, la clasificación de los IDS de acuerdo a diferentes criterios y las métricas de desempeño utilizadas para evaluar los IDS (sección 1.2). Para una mayor comprensión de lo anterior, se hace una descripción de los registros de conexiones de red, se describen los atributos o características que constituyen dichas conexiones (sección 1.3), se documentan los diferentes *datasets* empleados como soporte en IDS (sección 1.4), también se documentaron las diferentes fases del proceso de simulación en detección de intrusiones (sección 1.5), identificando los fundamentos relativos a las técnicas de extracción y selección de características (sección 1.6), en las que se basan diferentes propuestas de investigación que se han tomado como referentes. El capítulo culmina con unas conclusiones.

1.1.1. Ataques informáticos

El propósito del ataque informático es aprovechar las vulnerabilidades (debilidades o fallas inherentes a los sistemas operativos, aplicaciones, protocolos de comunicación, hardware o personas que interactúan con el sistema) del sistema informático, con la intención de causar un efecto negativo en su seguridad y consecuentemente obtener un beneficio o reconocimiento, usualmente económico. Un ataque informático está constituido por cinco fases, definidas en [1], las cuales se detallan a continuación y se esquematiza en la Figura 1.1.

Reconocimiento (*Reconnaissance*): se obtiene información de la víctima potencial (una persona u organización), recurriendo técnicas, como la Ingeniería Social, el *Dumpster Diving* y el *sniffing*.

Exploración (*Scanning*): se utiliza la información obtenida del reconocimiento para sondear o explorar el sistema informático, y obtener información, relativa a direcciones IP, nombres de host, datos de autenticación, entre otros. Se utilizan herramientas de exploración, como el *network mappers*, los *port mappers*, los *network scanners*, los *port scanners* y los *vulnerability scanners*.

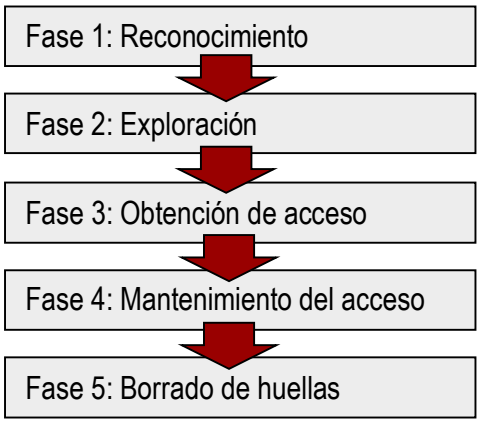

Figura 1.1 Fases de un ataque informático

Obtención de acceso (*Gaining Access*): se inicia el ataque a través de la explotación de las vulnerabilidades del sistema. Las técnicas utilizadas aquí son ataques de *Buffer Overflow*, de *Denial of Service* (DoS), *Distributed Denial of Service* (DDos), *Password filtering* y *Session hijacking*.

Mantenimiento del acceso (*Maintaining Access*). En esta fase se busca la implantación de herramientas que permitan mantener el acceso presente y futuro del atacante, ya sea de forma local o remota. Para ello se utilizan las herramientas *backdoors*, *rootkits* y troyanos.

Borrado de huellas (*Covering Tracks*). Consiste en eliminar los rastros dejados en archivos de registro (*log*) o alarmas del Sistema de Detección de Intrusos, producto del acceso intrusivo, con el objetivo de no ser posteriormente detectado e identificado por el administrador de la red informática.

Según [2] los ataques se clasifican en cuatro categorías principales (ver Figura 1.2): DoS (*Denial of Service*), R2L (*Remote to Local*), U2R (*Usert to Root*) y *Probing* (Vigilancia y otros tipos de sondeo de redes).

El **DoS** (*Denial of Service* - Denegación de Servicio) es un tipo de ataque que conlleva al agotamiento de los recursos en un sistema de cómputo o red informática (memoria, CPU o conectividad, entre otros), debido a la sobrecarga de los recursos computacionales (ancho de banda) de la red víctima. Durante este tipo de ataques se saturan los puertos de comunicación con excesivo flujo de datos, de tal forma que la sobrecarga del sistema haga imposible la correcta prestación del servicio, denegando las diferentes peticiones efectuadas por los clientes que las solicitan. Ver Tabla 1.1.

R2L (*Remote to Local* - Acceso no autorizado a una máquina remota) se produce cuando un atacante que no posee cuenta de usuario en una máquina remota, puede establecer sesión como

root o como usuario con restricciones en tal máquina. Ver Tabla 1.2.

Figura 1.2 Categorías de los ataques informáticos

Tabla 1.1 Clasificación de ataques de Denegación de Servicios

Ataque	Descripción
Back	Ataque contra el servidor web Apache cuando un cliente pide una URL que contiene muchas barras.
Land	Envío de TCP/SYN falso con la dirección de la víctima como origen y destino, causando que se responda a sí mismo continuamente.
Neptune	Inundación por envíos de TCP/SYN en uno o más puertos.
Pod	Ping de la muerte, envía muchos paquetes ICMP muy pesados.
Smurf	El atacante envía un ping, que parece proceder de la víctima, en broadcast a una tercera parte de la red, donde todos los host responderán a la víctima.
Teardrop	Usa el algoritmo de fragmentación de paquetes IP para enviar paquetes corruptos a la víctima.

U2R (*Usert to Root* - Acceso no autorizado a privilegios de super usuario) se produce cuando un atacante que ya dispone de una cuenta en un sistema informático, obtiene mayores privilegios de los inicialmente establecidos para él. Esto se da debido a vulneraciones de los sistemas operativos o de las aplicaciones, a la falta de concienciación de los usuarios o a la previa instalación de programas espías que posibiliten el acceso intrusivo. Ver Tabla 1.3.

Probing (Vigilancia o sondeo de redes), el propósito del *probing* es escanear redes de datos con el objeto de identificar direcciones IP válidas y recopilar información acerca de ellas. La información más relevante se refiere a detectar qué servicios ofrecen y los sistemas operativos que utilizan. A partir de esto, el atacante identifica una lista de vulnerabilidades potenciales para lanzar ataques tanto a los servicios como a las máquinas sobre las que éstos se ejecutan. Ver Tabla 1.4.

Tabla 1.2 Clasificación de Ataques categoría Remoto a Local

Ataque	Descripción
ftp_write	Usuario FTP remoto crea un archivo .rhost y obtiene un login local.
guess_passwd	Trata de adivinar la contraseña con telnet para la cuenta de visitante.
Imap	Desbordamiento remoto del búfer utilizando el puerto imap.
Multihop	Escenario de varios días donde el atacante primero accede a una máquina que luego usa como trampolín para atacar a otras máquinas.
Phf	Script CGI que permite ejecutar comandos en una máquina con un servidor web mal configurado.
Spy	Analizador de protocolos LAN por la interfaz de red.
Warezclient	Los usuarios descargan software ilegal publicado a través de FTP anónimo por el warezmaster.
Warezmaster	Subida FTP anónima de Warez (copias ilegales de software).

Tabla 1.3 Clasificación de Ataques categoría de Usuario a Root

Ataque	Descripción
buffer_overflow	Desbordamiento de la pila del búfer.
Loadmodule	Ataque furtivo que reinicia la IFS para un usuario normal y crea un shell de root.
Perl	Establece el id de usuario como root en un script de perl y crea un shell de root.
Rootkit	Escenario de varios días donde un usuario instala componentes de un rootkit.

Tabla 1.4 Clasificación de Ataques categoría Probing

Ataque	Descripción
ipsweep	Sondeo con barrido de puertos o enviando pings a múltiples direcciones host.
nmap	Escaneo de redes mediante la herramienta nmap.
portsweep	Barrido de puertos para determinar qué servicios se apoyan en un único host.
satan	Herramienta de sondeo de redes que busca debilidades conocidas.

De acuerdo a la clasificación de ataques planteada por [3], coherente con el análisis de incidentes del CERT (*Computer Emergency Response Team*), el cual se describe en [4] y coincidente con lo indicado por los investigadores del "*Sandia National Laboratories*" en [5], se construyeron las Tablas 1.1, 1.2, 1.3 y 1.4 que muestran la clasificación de los 24 ataques más conocidos dentro de las categorías anteriormente enunciadas (DoS, R2L, U2R y *Probing*) y que están presentes en el *dataset* DARPA, que se describirá en la sección 1.4.

Con el propósito de ejecutar acciones preventivas que inhiban los efectos nocivos que pueden causar los ataques informáticos en los sistemas computacionales, se han desarrollado diferentes

mecanismos de defensa que pueden coexistir en los ámbitos de los sistemas de red.

1.1.2. Mecanismos de defensa tradicionales contra ataques informáticos

Para detener y prevenir accesos intrusivos en las redes informáticas se utilizan diferentes métodos de defensa tradicionales, como: Firewall, Proxies, Redes Privadas Virtuales – VPN y Listas de Control de Acceso – ACL.

Firewall: En el RFC 2979 [6] se define Firewall como un agente que filtra o bloquea el tráfico de red que considera inapropiado, peligroso, o ambos. Un Firewalls puede actuar como un protocolo de punto final o de relevo (por ejemplo, un cliente/servidor SMTP o un agente *proxy web*), como un filtro de paquetes, o una combinación de ambos. Cuando un firewall actúa como protocolo de punto final, podrá: (1) implementar un subconjunto "seguro" del protocolo, (2) realizar una extensa comprobación de validez del protocolo, (3) utilizar una metodología de implementación diseñada para reducir al mínimo la probabilidad de errores, (4) ejecutarse en un entorno seguro y aislado, y (5) utilizar una combinación de estas técnicas en tándem. Cuando un firewall actúa como filtro de paquetes no es visible como protocolo de punto final. El firewall examina cada paquete y luego: pasa el paquete hacia el otro extremo de la red sin cambios, descarta el paquete completo o maneja el propio paquete de alguna manera.

Proxy: se utilizan como intermediarios de la comunicación entre un cliente (que usan un navegador web) ubicado en la LAN y un router conectado a internet. Para resolver las peticiones de los usuarios que desean acceder a Internet, si un ordenador realiza una petición al servidor proxy, es el proxy quien realmente accede a Internet y posteriormente le envía los datos al ordenador del usuario para que éste los visualice en su pantalla. El ordenador del usuario no tendrá conexión directa con el router, sino que las peticiones irán dirigidas al proxy y este se las pasará al router. Con el propósito de analizar el tráfico de red que pasa por el proxy, en busca de contenido que intente violar la seguridad de la red. Las páginas o sitios web a los cuales accede el cliente por intermedio del proxy se conservan en la memoria caché del proxy, para facilitar el acceso posterior a dichas páginas, disminuyendo los tiempos de respuesta y visualización. Además los servidores proxy aumentan también la seguridad, ya que filtran el contenido web, definen estrategias de autenticación y posibilitan el rastreo de accesos mediante archivos log, con el propósito de detectar programas maliciosos. En el RFC 2068 que se describe en [8] se define formalmente el concepto de servidor proxy, sus consideraciones prácticas y los requerimientos en cuanto autenticación/autorización, entre otros.

Redes Privadas Virtuales – VPN: según [9] una VPN se construye dentro de una infraestructura de red pública. Utilizada para conectar de forma segura oficinas y usuarios remotos a través de accesos a Internet proporcionados por terceros, en lugar de costosos enlaces WAN dedicados o enlaces de marcación remota de larga distancia. Las VPN proporcionan niveles de seguridad mediante IP (IPsec) cifrada o túneles VPN de Secure Sockets Layer (SSL) y tecnologías de autenticación. Una descripción pormenorizada sobre VPN, en relación a consideraciones de seguridad, calidad de servicio, escalabilidad, entre otros, se detallada en el RFC 4364 en [10].

Listas de Control de Acceso – ACL: En el RFC 4314 descrito en [11], que actualizó al RFC 2086, se definen los fundamentos referidos a las ACL en relación a varios derechos de control de acceso y se clarifican los privilegios para diferentes comandos IMAP (Protocolo de Acceso a Mensajes de Internet). Las ACLs interceptan el tráfico de una red y evalúan cada paquete comparando sus

valores particulares con valores predefinidos por el administrador de la lista a partir de lo cual se aplica una política de seguridad que permite o niega el acceso a un determinado segmento de red.

Si bien estos métodos de defensa tradicionales, son importantes herramientas de detección y prevención de ataques, no se constituyen en una solución integral. Los routers con ACL, los Firewall, Proxy y conexiones VPN, si bien ofrecen protección perimetral, por estar configuradas en dispositivos frontera, no son una solución para implementar en el núcleo de la LAN, en aras de evaluar el tráfico malicioso que se pudiese generar desde el interior de la red. Dada la cantidad cada vez mayor de tráfico de red y la potencial amenaza existente en relación a la vulneración de políticas de seguridad informática (Confidencialidad, Integridad y Disponibilidad - CIA), el estudio de los sistemas de detección de intrusos (IDS) recibe cada vez mayor atención.

1.2 SISTEMAS DE DETECCIÓN DE INTRUSOS

En [12], se hace una clara distinción entre los términos: intrusión, detección de intrusión, sistema de detección de intrusiones (IDS) y sistema de prevención de intrusiones (IPS). Según [13] la **intrusión** es un intento de comprometer las políticas de seguridad informática CIA, o de eludir los mecanismos de seguridad de una computadora o red. Las intrusiones son causadas por atacantes que acceden al sistema desde Internet, por los usuarios autorizados al sistema que intentan obtener privilegios adicionales para los que no están autorizados, y por los usuarios autorizados que hacen mal uso de los privilegios otorgados. En cuanto a la **detección de intrusos,** esta es el proceso de monitorización de eventos que ocurren en un sistema informático o red, y el análisis de éstos buscando rastros de intrusiones. Además, según [13] los IDS son el software o hardware empleado para automatizar el proceso de detección de intrusiones. En [14] se complementa dicha definición, indicando que un IDS puede actuar como una segunda línea de defensa para proporcionar análisis de seguridad.

Según [14] los IPS además de poseer todas las capacidades de los IDS, pueden detener posibles incidentes. En la gran mayoría de artículos revisados se utilizan indistintamente los términos IDS e IPS como sinónimos, haciendo las aclaraciones respectivas. Muy raramente se utiliza el término IDPS (Sistema de Detección y Prevención de Intrusos) para referirse a un IPS. En este documento, se utilizará el término IDS para referirse al proceso tanto de detección como de prevención.

En coherencia con el propósito de esta investigación, es conveniente recopilar una sólida fundamentación teórica sobre los IDS, en cuanto a: su estandarización actual y las organizaciones que han promovido dicha estandarización, sus componentes, clasificación y las métricas utilizadas para evaluar su efectividad. Cada uno de estos temas será abordado con suficiente detalle en esta sección.

1.2.1. Estandarización de los IDSs

Se ha tratado de estandarizar la arquitectura de los IDSs, inicialmente mediante dos propuestas, la primera efectuada por el CIDF (Marco de Detección de Intrusiones Común) descrito en [15] y [16], y la segunda por autopost de AusCERT (Equipo de Respuesta Australiano a las Emergencias Computacionales) que se describe en [17], ambos proyectos culminaron infructuosamente, razón por la cual se han venido desarrollando otros esfuerzos de estandarización liderados por IDWG (Grupo de Trabajo de Detección de Intrusiones) cuyos detalles pueden ser consultados en [18],

[19], [20], [21] y [22]. De manera complementaria se ha planteado una propuesta de diccionario de nombres comunes, cuyo propósito es dar a conocer públicamente las vulnerabilidades relacionadas con la seguridad de la información, la cual se ha denominado CVE (Vulnerabilidades y Exposiciones Comunes), definida en [23]. A continuación se describen los componentes más relevantes de un IDS.

1.2.2. Componentes de los IDSs

En [12], se hace un minucioso estudio de los componentes de los IDS, allí se indica que éstos primordialmente están constituidos por **sensores** y **agentes**. Los **sensores** son normalmente utilizados para monitorizar redes, en IDS de Redes - NIDS, IDS de Redes Inalámbricas - WIDS e IDS que aplican Análisis del Comportamiento de la Red - NBA (todos éstos son tipos de IDS de acuerdo a la fuente u origen de la información y serán definidos posteriormente, de manera minuciosamente en este mismo capítulo). Los **agentes** se utilizan para monitorizar y analizar las actividades en IDS de Host - HIDS.

Tanto el sensor como el agente pueden entregar los datos al **Servidor de Administración** (MS) y al **servidor de base de datos** (DS), el MS es un dispositivo centralizado para el procesamiento de incidentes capturados y el DS es sólo un repositorio que almacena la información de eventos.

Por otra parte, existen dos tipos de arquitecturas de red, la **Red Gestionada** (MN) y la **Red Estándar** (SN). La primera es una red aislada, para el despliegue de la gestión del software de seguridad, que oculta la información del IDS a los intrusos, este tipo de arquitectura aumenta los costos, dado que requiere de hardware adicional y provoca ciertos inconvenientes para los administradores. Por otra parte, la **Red Estándar** (SN), es pública, sin protección y por lo tanto para mejorar su seguridad se debe construir una red aislada virtual mediante la configuración de una red de área local virtual.

De forma complementaria, la mayoría de las tecnologías IDS poseen cuatro capacidades comunes para mantener la seguridad: la recopilación de información, el registro de inicio de sesión (logging), la detección y la prevención. La **recopilación de información** colecta información en hosts/redes de actividades observadas. El **inicio de sesión** (logging), se refiere al registro de los datos de inicio de sesión relacionados para eventos detectados, se pueden utilizar para validar las alertas e investigar incidentes. Las metodologías de **detección** en la mayoría de los IDS generalmente necesitarán de mayor afinamiento en aras de conseguir mayor precisión. En relación a la **prevención**, un estudio en profundidad puede consultarse en [14].

1.2.3. Clasificación de los IDSs

En la Tabla 1.5 se clasifican los IDS de acuerdo con los criterios de: (1) estrategia o tipo de análisis, (2) fuente de información, (3) arquitectura o estructura, (4) respuesta o comportamiento, (5) tipo de predicción y (6) implementación. Por su parte [24] aporta una clasificación basada en la aplicación práctica de los IDS. De forma más detallada en [25] se enfocan en la clasificación de los Sistemas de Detección de Intrusos en Red basados en Anomalías (A-NIDS).

1.2.3.1. Estrategia de análisis

En [13], [26], [27], [28], [29] y [30] se muestran dos claras tendencias de IDSs en relación al

enfoque o tipo de análisis: detección basada en abusos y detección basada en anomalías. Por su parte, en [12], [31] y [32] se describe una tercera tendencia denominada análisis de protocolo de estado. A continuación se contextualiza cada una de estas estrategias de análisis (Figura 1.3).

Tabla 1.5 Clasificación de los IDS

Criterio clasificatorio	Tipo de IDS
Estrategia de análisis	Detección basada en abusos
	Detección basada en anomalías
Fuente de información	Basada en Host – HIDS
	Basada en Red – NIDS
	Registro de aplicación
	Redes Inalámbricas
	Alarmas de sensor
Arquitectura	Centralizada
	Distribuida (DIDS)
Respuesta	Activa
	Pasiva
Tiempo de predicción	En tiempo real
	Fuera de línea
Implementación	Basado en Inteligencia Artificial
	Basado en Cálculo Computacional
	Basado en conceptos biológicos

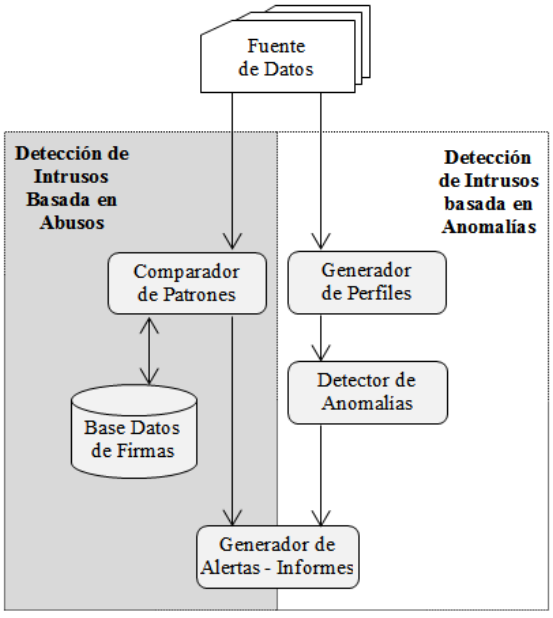

Figura 1.3 Esquema funcional de los IDS de acuerdo a su estrategia de análisis

Detección basada en abusos

Conocidos por la denominación detección de uso incorrecto, de abusos o basados en firmas y en inglés es llamado *misuse-based*. Su funcionamiento consiste en monitorizar las actividades que

ocurren en un sistema e irlas comparando con una base de datos de firmas de ataques predefinida y en el caso en que se detecte que un ataque coincide con una firma se genera una alarma. Según [33] esta técnica ha sido muy favorecida en productos comerciales, debido a su previsibilidad y alta precisión. Sin embargo es importante resaltar que para que el método sea efectivo se requiere tener una base de datos de firmas constantemente actualizada, razón por la cual la deficiencia de este método es su incapacidad de detectar nuevos ataques no presentes en dicha base de datos.

Las firmas o reglas básicamente están constituidas por cabecera y opciones. Esto permite identificar el tráfico. La cabecera de firma contiene el protocolo (TCP, UDP, IP o ICMP), las direcciones y puertos tanto de origen como de destino, el sentido de la comunicación y la acción que se ejecutará si coinciden las características del paquete con las condiciones de la regla (Figura 1.4).

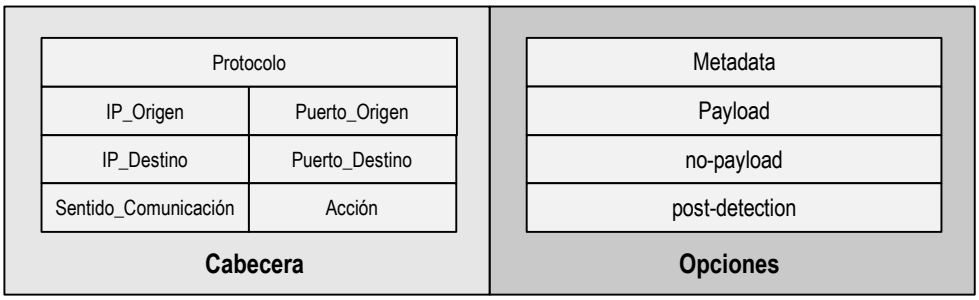

Figura 1.4 Estructura de una regla o firma

Adicionalmente existen cuatro opciones que hacen parte de la firma y son: (1) el campo ***metadata*** que proporciona información sobre la firma, (2) el campo ***payload*** o carga útil que se emplea para buscar información dentro de los datos del paquete, (3) el campo ***no-payload*** utilizado para buscar información no útil dentro del campo ***payload*** del paquete y (4) el campo ***post-detection*** donde se especifican reglas que ocurrirán después de ejecutada una regla en particular.

Ejemplos de IDSs que emplean las técnicas de detección basada de abusos son: Snort [34], NFR - Network Flight Recorder [35], NSM - Network Security Monitor [36], Cisco Intrusion Detection - NetRanger [37] y RealSecure [38].

Detección basada en anomalías

El proceso de detección de anomalías funciona asumiendo que los ataques son diferentes a la actividad normal. Se puede llegar a esta inferencia tras un proceso de entrenamiento, en el cual se identificará qué se considera como actividad normal analizando comportamientos inusuales en los host y en general en el tráfico de la red. Para ello se construyen perfiles generados a partir del análisis de asociación a patrones. Estos perfiles representan el comportamiento normal de los usuarios, hosts o conexiones de red.

Los perfiles se construyen durante el período normal de operación, a partir de la recolección de información histórica usando detectores que recogen los datos de los eventos y emplean una variedad de medidas con el objetivo de determinar cuándo la actividad monitorizada tiende a salirse de la normalidad. Las medidas y técnicas comúnmente utilizadas en la actualidad en los IDSs para la detección de anomalías son la detección de umbral, y el uso de medidas estadísticas.

La técnica de **detección de umbral** se aplica sobre atributos de comportamiento del usuario. Estos atributos pueden referirse al número de ficheros accedidos por un usuario en un periodo de tiempo dado, el número de intentos fallidos para entrar en el sistema o la cantidad de CPU utilizada por un proceso, entre otros. La identificación de dicho umbral puede ser estadística o heurística.

Las **medidas estadísticas** pueden ser paramétricas, cuando se asume que la distribución de los atributos perfilados encaja con un determinado patrón; o no paramétricas, cuando la distribución de los atributos perfilados se aprende de un conjunto de valores históricos, observados a lo largo del tiempo. Un ejemplo de sistemas basados en métodos estadísticos es el ISA-IDS (*Information and System Assurance Laboratory Intrusion Detection System*).

Técnicas de detección de anomalías como la aplicación de IDES (*Intrussion Detection Expert System*), redes neuronales, algoritmos genéticos, modelos de sistemas inmunes y NIDES (*next-generation intrusion detection expert*) con el que se pretende optimizar y rediseñar el IDES, no se emplean en la actualidad en los IDSs con fines comerciales, debido a que aún siguen siendo objeto de investigación.

Según [33], mediante un análisis **exhaustivo** de las **tendencias** en investigación en relación a la detección de anomalías, se han encontrado varios métodos de aprendizaje máquina, que tienen una muy alta tasa de detección (98%) pero con una tasa de falsas alarmas muy baja (1%). Tal estudio se muestra en [39].

Un análisis más minucioso respecto a las ventajas y desventajas de estos dos enfoques (basados en abuso y anomalías) y las diferentes técnicas utilizadas en los IDS en red basados en anomalías (A-NIDS) puede ser consultada en la revisión que se hace en [25].

Análisis de Protocolo de Estado

Según [12] en el Análisis de Protocolo de Estado (*Stateful Protocol Analysis* - SPA), también denominado detección basada en especificación, el estado indica que el IDS podría conocer y rastrear los estados del protocolo (por ejemplo, las solicitudes de emparejamiento de las respuestas). Aunque los procesos SPA son similares a los ejecutados en la detección basada en anomalías, en esta última se adopta la red precargada o perfiles específicos de host, mientras que SPA depende de perfiles genéricos desarrollados por proveedores para protocolos específicos. Es decir, los modelos de protocolo de red en SPA se basan originalmente en protocolos estándares de organizaciones internacionales de estándares, por ejemplo, la IETF.

Clasificación de los subtipos de técnicas basadas en anomalías

Retomando lo anteriormente indicado, en relación a la detección basada en anomalías, según [40] el tipo de procesamiento relacionado con el modelo de "comportamiento" del sistema destino, clasifica las técnicas de detección de anomalías en tres categorías principales, tal y como se aprecia en la Tabla 1.6.

- **Técnica de detección de anomalías basada en estadísticas.** Según [25], en las técnicas basadas en estadísticas se captura la actividad del tráfico de red y se crea un perfil que representa su comportamiento estocástico. Este perfil se basa en métricas tales como la tasa

de tráfico, el número de paquetes para cada protocolo, la tasa de conexiones y el número de diferentes direcciones IP, entre otras. Las técnicas de detección de anomalías basadas en estadística univariable, que modelan los parámetros como variables aleatorias gaussianas independientes, han sido definidas en [41]. Una revisión detallada de los modelos multivariables se realiza en [42]. Por último, en [43] se consideran los modelos de series de tiempo.

Tabla 1.6 Clasificación de los subtipos de cada técnica del A-NIDS

Técnicas de A-NIDS	Subtipos
Basada en estadísticas (comportamiento estocástico)	Modelos univariable (variables aleatorias gaussianas independientes)
	Modelos multivariable (correlaciones entre varias métricas)
	Series de tiempo (temporizadores, contadores, etc)
Basada en el conocimiento (disponibilidad a priori del conocimiento a partir de datos)	Máquinas de estado finito (estados y transiciones)
	Lenguajes de descripción (N-grams, UML,…)
	Sistemas expertos (clasificación basada en reglas)
Basada en aprendizaje automático (categorización de patrones)	Redes bayesianas (relaciones probabilísticas entre variables)
	Modelos de Markov (teoría estocástica de Markov)
	Redes Neuronales (fundamentadas en el cerebro humano)
	Lógica difusa (aproximación e incertidumbre)
	Algortimos Genéticos (inspirados en la biología evolutiva)
	Detección de agrupamientos y valores atípicos (agrupamiento de datos)

- **Técnica de detección de anomalías basada en el conocimiento.** El enfoque más utilizado, entre las técnicas de detección basadas en el conocimiento, lo constituyen los sistemas expertos. En [41] y [44] se definen y clasifican de acuerdo a diferentes categorías, indicando además cómo los sistemas expertos están destinados a clasificar los datos de auditoría de acuerdo con un conjunto de reglas. Por su parte, en los métodos de anomalías basados en las especificaciones, el modelo deseado es construido manualmente por un humano experto, basado en un conjunto de reglas o especificaciones que tratan de determinar el comportamiento del sistema legítimo. Tales especificaciones se podrían desarrollar mediante el uso de algún tipo de herramienta formal (la metodología de Máquina de Estados Finitos - FSM o lenguajes de descripción estándar tales como N-gramáticas y UML).

- **Técnica de detección de anomalías basada en aprendizaje automático.** Según [25], se basan en el establecimiento de un modelo explícito o implícito capaz de categorizar a los patrones analizados. Una característica singular de estos esquemas es la necesidad de datos etiquetados para entrenar el modelo de comportamiento. Se trata de un procedimiento que requiere demandas sustanciales sobre los recursos. Varios esquemas basados en el aprendizaje automático se han aplicado a IDS de red basados en anomalías. La Tabla 1.7 muestra los pros y contras de las técnicas de detección de anomalías basadas en aprendizaje automático.

Procedimientos Híbridos

Tanto los IDSs basados en detección de abusos, como los basados en detección de anomalías presentan ventajas e inconvenientes. Los primeros son más fiables, proporcionando mejor rendimiento frente a ataques conocidos gracias al uso de firmas, pero no poseen la capacidad de detectar nuevos ataques no incluidos en la base de datos de firmas. Por el contrario, los segundos poseen la capacidad de detectar ataques desconocidos, aunque su rendimiento es inferior. Por

tanto el mejor modelo para solucionar las carencias de cada uno de los IDSs anteriormente comentados es integrar las ventajas de ambos en un modelo híbrido capaz de detectar ataques previamente definidos y de interceptar ataques, que no han sido previamente definidos, analizando comportamientos anormales del sistema.

Tabla 1.7. Descripción de técnicas de detección de anomalías basadas en aprendizaje automático

Técnica	Descripción	Ventajas	Inconvenientes
Redes Bayesianas	Modelo que codifica las relaciones probabilísticas entre variables de interés. Esta técnica se utiliza generalmente para la detección de intrusiones en combinación con esquemas estadísticos	Capacidad de codificación de interdependencias entre las variables y de predicción de eventos. Capacidad de incorporar tanto conocimiento previo como datos. [46].	Sus resultados son similares a los derivados de los sistemas basados en umbrales, requiriendo considerablemente mayor esfuerzo computacional. [47].
Cadena de Markov	Conjunto de estados que están interconectados a través de probabilidades de transición, que determinan la topología y las capacidades del modelo. Durante una primera fase de entrenamiento, las probabilidades asociadas a las transiciones se estiman a partir del comportamiento normal del sistema destino. La detección de anomalías se lleva a cabo a continuación, mediante la comparación de la puntuación de anomalías (probabilidad asociada) obtenida para las secuencias observadas con un umbral fijo.	Se han utilizado ampliamente en el contexto de IDS de host, normalmente aplicados a las llamadas del sistema. [48]. En IDS de red, se han utilizado modelos de Markov en [49] y [50], que han proporcionado un buen enfoque para el perfil reivindicado.	Los resultados son altamente dependientes de los supuestos sobre el comportamiento aceptado para el sistema. [49] y [50].
Redes Neuronales	Simula el funcionamiento del cerebro humano para procesos de clasificación de datos, mediante la creación de perfiles, con el propósito de hacer predicciones. [51] [52].	Es ampliamente utilizado en IDS para identificar el comportamiento intrusivo de patrones de tráfico. [53].	Las redes neuronales recurrentes a los mapas auto-organizados no proporcionan un modelo descriptivo que explique por qué se ha tomado una decisión particular de detección. [54].
Lógica difusa	Se deriva de la teoría de conjuntos difusos en las que el razonamiento es aproximado en lugar de ser deducido de la lógica de predicados clásica. Este tipo de esquema de procesamiento considera una observación como normal si se encuentra dentro de un intervalo dado. [55].	Técnicas difusas son utilizadas en el campo de la detección de anomalías debido principalmente a que las características que deben ser consideradas pueden ser vistas como variables difusas. Aunque la lógica difusa ha demostrado ser eficaz, especialmente para el análisis de puertos y el sondeo. [56].	Alto consumo de recursos involucrados. La LD es controvertida en algunos círculos, que sostienen que la probabilidad es la única descripción matemática rigurosa de incertidumbre. [25].
Algoritmos Genéticos	Se clasifican como heurísticas de búsqueda global, y son una clase particular de algoritmos evolutivos que utiliza técnicas inspiradas en la biología evolutiva como la herencia, mutación, selección y recombinación. Se constituyen en otro tipo de técnicas basadas en el aprendizaje de automático, capaz de derivar reglas de clasificación y/o la selección de características apropiadas o parámetros óptimos para el proceso de detección. [57] y [56].	Utiliza un método de búsqueda global flexible y robusto, que converge a una solución desde múltiples direcciones, mientras no se asume conocimiento previo sobre el comportamiento del sistema. [56].	Alto consumo de recursos involucrados. [25]
Detección de agrupamientos y valores atípicos	Agrupan los datos observados en los conglomerados (clusters), de acuerdo con una similitud determinada o medida de distancia. El procedimiento más comúnmente utilizado para esto consiste en seleccionar un punto representativo para cada clúster. Luego, cada nuevo punto de datos es clasificado como perteneciente a un grupo determinado de acuerdo a la proximidad al punto representativo correspondiente [58]. Algunos puntos no pueden pertenecer a ningún grupo; éstos se denominan valores atípicos y representan las anomalías en el proceso de detección.	Se utiliza en la actualidad en el campo de los IDS [59], [60]. Las técnicas de agrupamiento (clustering) determinan la ocurrencia de eventos de intrusión sólo de los datos de auditoría en bruto, y por lo tanto el esfuerzo requerido para ajustar el IDS se reduce.	Alto consumo de recursos involucrados. [25]

El MINDS (*Minnesota Intrusion Detection System*), desarrollado en la Universidad de Minnesota es un IDS híbrido, debido a que complementa a *Snort* (IDS de detección de abusos basado en firmas), y detecta anomalías por medio del algoritmo SNN (*Shared Nearest Neighbour*). Una vez

detectadas las anomalías, realiza un resumen de las mismas mediante el análisis de asociación de patrones, recogiendo las características que determinan que se trata de un ataque y añadiéndolas a la base de datos como firmas nuevas. Las técnicas de detección de anomalías y de detección de abusos utilizan diferentes tipos de algoritmos como estrategias de análisis, para dotar al sistema de conocimiento y de aprendizaje autónomo ver [2] y [45].

1.2.3.2. Clasificación según las fuentes de información

De acuerdo con la procedencia u origen de los datos, según [12] existen cuatro clases de tecnologías de IDS. Los **NIDS** (*Network Intrusion Detection Systems*) analizan paquetes de red, capturados del backbone de la red o de segmentos de la LAN. Los **HIDS** (*Host Intrusion Detection Systems*) analizan eventos generados por los sistemas operativos o software de aplicación de los equipos de cómputo de la red en busca de señales de intrusión. Los **WIDS** (*Wireless Intrusion Detection Systems*) son similares a los NIDS, pero capturan el tráfico de redes inalámbricas, tales como redes *ad hoc*, redes de sensores inalámbricas y redes en malla (*mesh*) inalámbricas. Además, los sistemas **NBA** (*Network Behavior Analysis*) inspeccionan el tráfico de red para reconocer ataques con inesperados flujos de tráfico. A continuación se proporciona una sucinta descripción de cada uno.

NIDS

Según [2] los Sistemas de Detección de Intrusos basados en Red analizan la información obtenida mediante la monitorización de las infraestructuras de red. Las fuentes típicas de este tipo de información son los paquetes de conexión de red recogidos por rastreadores (*sniffers*) de red y de la información administrada entre los dispositivos de red recolectada debido al uso del SNMP (*Simple Network Management Protocol*).

El análisis del tráfico de red que normalmente hacen los NIDS en tiempo real, es posible gracias a la configuración del dispositivo sensor de red en modo promiscuo, capturando todos los paquetes que pasan por el dispositivo y posteriormente almacenando la información de los paquetes en un repositorio para su subsecuente análisis en busca de patrones indicativos de ataques. Los NIDS y cortafuegos deben ser usados como mecanismos complementarios en la seguridad de la red, debido a que los primeros detectan tráfico potencialmente dañino procedente del interior de la red o que procede del exterior, habiendo pasado por el cortafuegos, y éste último, filtra el tráfico procedente del exterior de la red, mediante la restricción de servicios asociados a determinados puertos. La Figura 1.5 describe el esquema funcional de los NIDSs.

HIDS

Los Sistemas de Detección de Intrusos en Host (HIDS), han sido los primeros en ser desarrollados y su funcionalidad radica en evaluar la información que generan los usuarios, el sistema operativo y las aplicaciones de una computadora (host), conectada a una red informática, con el objetivo de identificar amenazas e intrusiones ejecutadas a nivel del host local. Los HIDSs, a diferencia de los NIDSs, pueden ver el resultado de un intento de ataque, e incluso acceder directamente y monitorizar los ficheros de datos y procesos del sistema atacado. Han sido diseñados para monitorizar, detectar y responder a los datos generados por un usuario o un sistema en un determinado host.

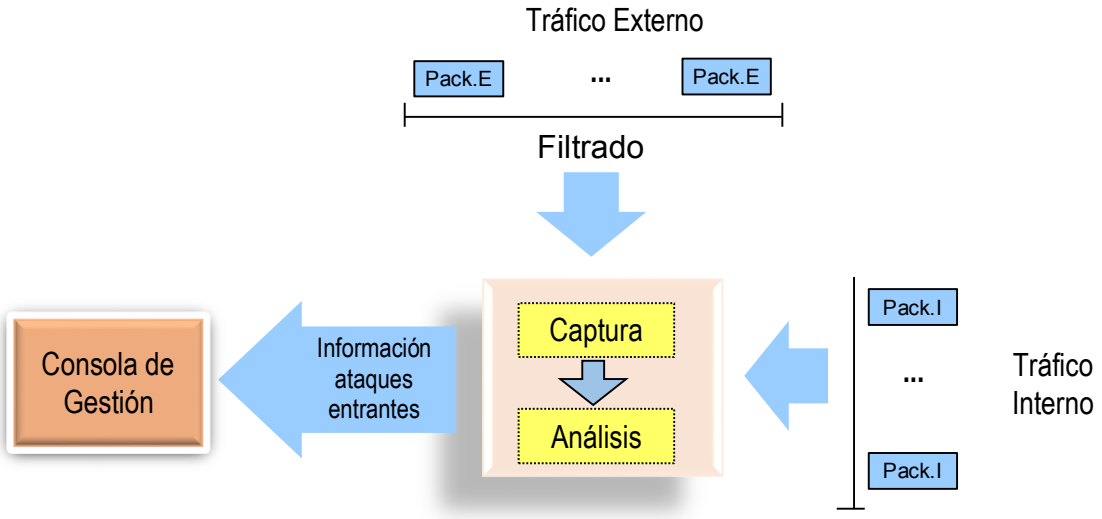

Figura 1.5 Esquema funcional de un NIDS

Los HIDSs impactan el sistema que protegen, debido a que comparten los mismos recursos (sistema operativo y aplicaciones) de éste. Pero además se ven afectados por el sistema donde residen, presentando vulnerabilidades ante un ataque directo a dicho sistema. Las bitácoras del sistema son generadas de forma automática por diferentes aplicaciones o por el propio núcleo del sistema operativo, siendo usadas por los HIDSs para el análisis de los registros relativos a servicios ejecutados en segundo plano, y para verificar la integridad de determinados ficheros de importancia vital para el sistema (contraseñas). La Figura 1.6 muestra el esquema funcional de un HIDS. En [2] se puede ampliar información referida a los HIDSs.

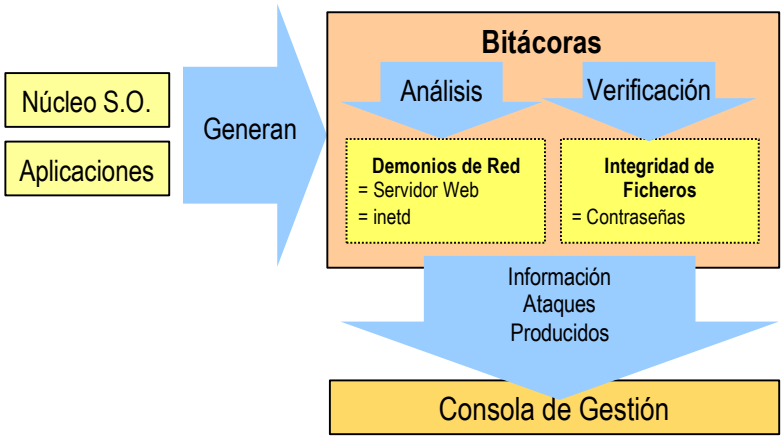

Figura 1.6 Esquema funcional de un HIDS

WIDS

Según [61], un Sistema de Detección de Intrusos en Redes Inalámbricas monitoriza el tráfico de la red inalámbrica y analiza sus protocolos de redes inalámbricas para identificar actividad sospechosa que involucre esos protocolos. Se utilizan primordialmente para monitorizar el tráfico

de redes de área local inalámbricas (WLAN), basadas en la familia de estándares IEEE 802.11 para WLAN [62], en un rango de red bastante limitado, como un edificio de oficinas o un campus corporativo. Las WLAN IEEE 802.11 tienen dos componentes arquitectónicos fundamentales una estación, que es un dispositivo para el acceso a la red inalámbrica, y un punto de acceso, que conecta lógicamente las estaciones con la infraestructura de la red cableada. Cada punto de acceso en una WLAN tiene un nombre asignado llamado un identificador de conjunto de servicios (SSID), que permite distinguir un punto de acceso de otro.

Los IDSs inalámbricos pueden detectar ataques, errores de configuración, y violaciones de políticas a nivel de protocolo WLAN, examinando principalmente el protocolo de comunicación IEEE 802.11 [63]. Los IDSs inalámbricos no examinan las comunicaciones en los niveles superiores (por ejemplo, direcciones IP, cargas útiles –*payload*- de aplicación). Algunos productos realizan sólo una detección basada en firmas, mientras que otros utilizan una combinación de técnicas de detección basada en firmas, basada anomalías, y de análisis de protocolo de estado.

NBA

Un sistema de Análisis del Comportamiento de la Red (NBA) examina el tráfico de red para identificar los flujos de tráfico inusuales, tales como ataques DDoS, ciertas formas de *malware* y violaciones de políticas. Los sistemas NBA también han sido denominados *software* de detección de anomalías de comportamiento de red (NBAD). Los NBA suelen tener sensores y consolas, y algunos productos también ofrecen servidores de administración [61].

Pese a que la mayoría de las tecnologías NBA no ofrecen capacidades de detección basadas en firmas, si permiten a los administradores configurar manualmente filtros personalizados (patrones) que son esencialmente firmas para detectar o detener ataques específicos. La mayoría de los productos NBA utilizan principalmente detección basada en anomalías junto con algunas técnicas de análisis de protocolo de estado [64]. Los eventos comúnmente detectados por los sensores NBA son ataques DoS basados en red, escaneo en red, gusanos, uso de servicios de aplicaciones inesperadas, y violaciones de política.

Procedimientos Híbridos

Los sistemas híbridos combinan características de los NIDSs (poseer sensores por segmentos de red) y de los HIDS (poseer sensores por cada host). Así permiten obtener información general y específica de un ataque, dadas sus características de NIDS y HIDS. Prelude, es un Sistema de Detección de Intrusos híbrido, distribuido bajo licencia GPL y desarrollado principalmente bajo GNU/LINUX en [65] hay mayor información al respecto.

En [12] se efectúa un análisis comparativo de las tecnologías IDS clasificadas según la fuente de información, valorando los componentes funcionales que las constituyen, el alcance de la detección, la arquitectura y las capacidades de seguridad. Se presenta un resumen en la Tabla 1.8.

1.2.3.3. Arquitectura de los IDS

En [66] se clasifican los IDS en función de su arquitectura o estructura topológica, en centralizados y distribuidos. La principal diferencia entre éstos, es que los del primer tipo analizan los datos

recolectados desde un sistema de monitorización único y el segundo recolecta la información desde múltiples sistemas de monitorización con el propósito de realizar una investigación global, distribuida y coordinada de los ataques. A continuación se proporciona una descripción sucinta de cada uno.

Tabla 1.8 Análisis comparativo de tecnologías IDS clasificadas por fuente de información

	Tecnologías			
Variables a valorar	**HIDS**	**NIDS**	**WIDS**	**NBA**
Componentes	Agente (software en línea). Servidor de Administración (uno o más). Servidor de Base de datos (uno o más – opcional).	Varios sensores (en línea y pasivos). Servidor de Administración (uno o más) Servidor de Base de datos (uno o más – opcional)	Varios sensores (pasivos). Servidor de Administración (uno o más) Servidor de Base de datos (uno o más – opcional)	Varios sensores (más pasivos). Servidor de Administración (uno o más opcional) Servidor de Base de datos (opcional)
Alcance de la detección del sensor o agente	Sólo host.	Varias subredes. Varios hosts.	Varias WLAN. Varios clientes WLAN.	Varias subredes. Varios hosts.
Arquitectura	Red Gestionada o Red Estándar.	Red Gestionada	Red Gestionada o Red Estándar	Red Gestionada o Red Estándar.
Fortalezas	Sólo los HIDS pueden analizar la actividad de las comunicaciones encriptadas punto a punto.	Capacidad de análisis de alcances más amplios de protocolos de aplicación.	WIDS es más exacto debido a su enfoque limitado. Sólo WIDS pueden supervisar la actividad de protocolo inalámbrico.	Poderosa detección superior en la exploración de reconocimiento, reconstruyendo las infecciones de malware y ataques de denegación de servicio.
Limitaciones tecnológicas	Más desafiante en la detección de precisión debido a una falta de conocimiento del contexto. Retrasos en la generación de alertas y de informes centralizados. Consume los recursos de host. Conflicto con los controles de seguridad existentes.	No puede monitorizar protocolos inalámbricos. Alta tasa de falsos positivos y falsos negativos. No puede detectar ataques dentro de tráfico encriptado. No tiene soporte de análisis completo bajo cargas elevadas.	No puede monitorizar actividades de protocolo de las capas de aplicación, transporte y red. No puede evitar técnicas evasión. Los sensores son susceptibles a ataques de interferencia física. No se puede compensar para protocolos inalámbricos inseguros.	La principal limitación es el retraso en la detección de ataques, causados por la transferencia de datos de flujo a la NBA en lotes, pero no en tiempo real.
Capacidades de seguridad				
Recogida de información	El tráfico de red, llamadas al sistema, la actividad del sistema de archivos.	Hosts, Sistemas Operativos, Aplicaciones, Tráfico de Red.	WLAN, dispositivos (por ejemplo: aplicaciones, clientes).	Hosts, Sistema Operativo, servicios (IP, TCP, UDP, etc).
Logging	Referencia [14]	Referencia [14]	Referencia [14]	Referencia [14]
Metodología de detección	Basada en Firmas y basada en Anomalías (combinada).	Basada en Firmas (principalmente), basada en Anomalías y Análisis de Protocolo de Estado.	Basada en Anomalías (principalmente), basada en firmas y análisis de protocolo de estado.	Basada en Anomalías (principalmente) y Análisis de Protocolo de Estado.
Tipo de eventos sospechosos detectados	Tráfico de red de las capas de Aplicación, Transporte y Red, registros de eventos (por ejemplo: actividades de aplicación, actividades del sistema de archivo), registros del sistema (por ejemplo: configuraciones, actividades del sistema operativo).	Capas de Aplicación, Transporte y Red, Hardware. Reconocimiento y ataques. Servicios inesperados de aplicación, violaciones de políticas.	Actividad de protocolo en inalámbricas, WLAN y dispositivos inseguros, ataques de denegación de servicios, escaneo de la red, violación de políticas.	Flujo de tráfico anómalo en las capas de aplicación, transporte y red (ataques de denegación de servicios, malware) servicios de aplicación no esperados, escaneo de red, violación de políticas.

Centralizados

Según [66], los IDS centralizados utilizan varios sensores que capturan y analizan la información local, retransmitiéndola a un IDS central que posee el control total del sistema. Esta estructura produce un ahorro de equipamientos, permitiendo reducir el mantenimiento de un amplio conjunto de sensores, desde donde se recoge la información de las posibles amenazas. La Figura 1.7 ilustra la topología de los IDSs centralizados. En [67] se proporciona con detalle la arquitectura de un IDS centralizado.

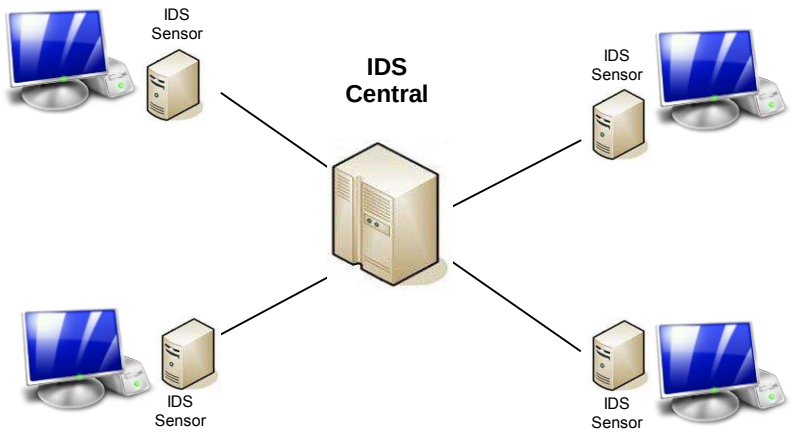

Figura 1.7 Topología del IDS Centralizado

Ejemplos de NIDSs centralizados, son ISOA descrito en [68] y [69], e IDES descrito en [70]. Tales sistemas transfieren la información de múltiples sensores IDS a un IDS central para el tratamiento de la información concerniente a los ataques. Los sensores IDS utilizan los mismos algoritmos que se emplean en los HIDS sin supervisar ningún tráfico de red.

Distribuidos (DIDS)

Los Sistemas de Detección de Intrusión Distribuidos (DIDS) nacen como un proyecto conjunto entre la UC.Davis – University of California, el Laboratorio Nacional de la Universidad de California en Berkeley, el Laboratorio de Alimar, y las Fuerzas Aéreas de los EE.UU. La intención de este proyecto fue subsanar la deficiencia existente en los sistemas centralizados, debido a que en sus estructuras de red, se genera una degradación del rendimiento, producto del análisis de todo el tráfico de información en un solo punto de la red. Por ello se planteó la idea de instalar sistemas distribuidos, que dispongan de varios sensores ubicados en diferentes puntos de la red, los cuales se comunican con un nodo central encargado de recibir toda la información relevante y donde se cruzan los datos para disponer de una visión más amplia del sistema como conjunto, y detectar con mayor fiabilidad los ataques.

Mediante este tipo de IDS se pretende ampliar la información disponible para favorecer la detección de incidentes en el sistema, implementando agentes distintos por toda la red y unificando la producción de respuestas de intrusiones que se detectan desde varios puntos de la red.

Según [40], a diferencia de un IDS centralizado, donde se realiza el análisis de datos sobre un número fijo de ubicaciones (independiente de la cantidad de hosts que están siendo

monitorizados), en un IDS distribuido el análisis de los datos se realiza en un número de localizaciones que es proporcional al número de los hosts que están siendo monitorizados. Una excelente comparación de IDSs centralizados y distribuidos, con sus ventajas y desventajas, se proporciona en [71]. A pesar de varios inconvenientes en los IDS distribuidos, muchos proveedores comerciales han planteado la necesidad de la detección de ataques cibernéticos coordinados desde ubicaciones distribuidas, y han adaptado sus sistemas para hacer frente a estos desafíos [72] y [73].

A partir de las primeras propuestas de IDS distribuidos [74], las arquitecturas más típicas de IDS distribuidos suponen el empleo de agentes inteligentes. Hay varias ventajas de utilizar sistemas de detección de intrusos basados en agentes inteligentes o móviles, a través de otros enfoques para la detección de intrusiones distribuida tal como se indica en [75]. En primer lugar, los agentes ejecutan de forma independiente entidades y se pueden añadir, eliminar y volver a configurar sin alterar otros componentes, y sin necesidad de reiniciar el IDS local. En segundo lugar, los agentes pueden comprobarse ellos mismos antes de introducirlos en un entorno más complejo. Finalmente, los agentes pueden intercambiar información para derivar resultados más complejos que cualquiera de ellos puede ser capaz de obtener por su cuenta. Aunque el estudio de los IDS basados en agentes móviles es un área de investigación relativamente nueva y los sistemas totalmente implementados están apareciendo ahora, hay muchos IDS distribuidos basados en agentes [76] y [77]. Ejemplos típicos son DIDS [78], AAFID [71], Argus [79], IDA [80], Micael [81].

La Figura 1.8 muestra la arquitectura funcional de un DIDS. La Consola de Eventos establece la interfaz con el operador y la comunicación con el Maestro que en este caso solo es un nodo. El maestro a su vez centraliza los datos de los n transceptores que constituyen n subsistemas, cada uno de los cuales está compuesto por un número determinado de agentes. Estos agentes se encargan de efectuar la monitorización de las actividades que se ejecutan en el sistema. Para mayor información respecto a la arquitectura de los IDS distribuidos se puede consultar [67].

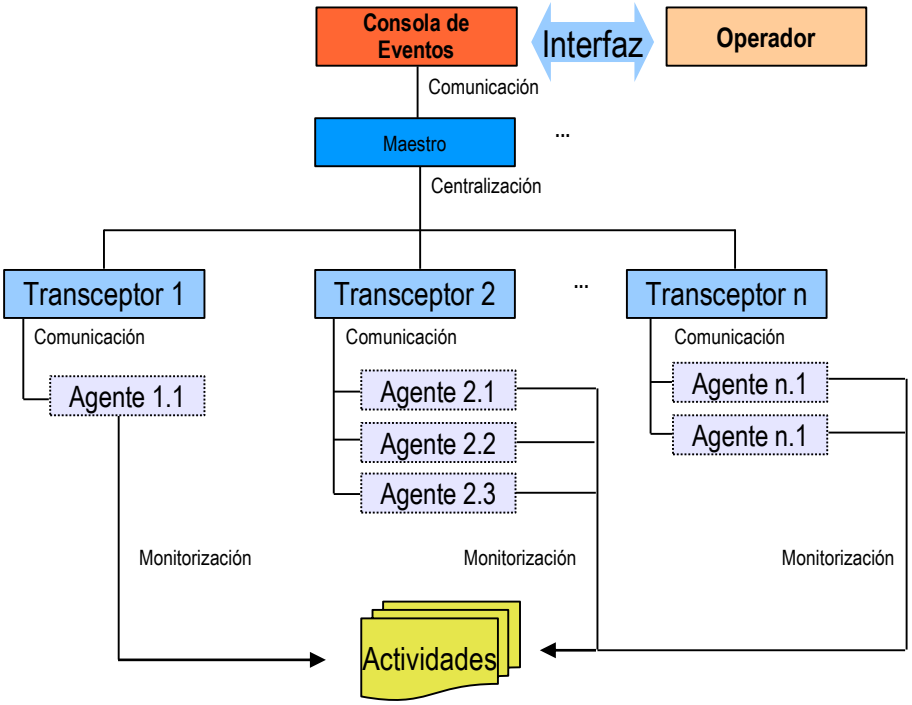

Figura 1.8 Arquitectura funcional de un DIDS

1.2.3.4. IDS según la respuesta

Según el comportamiento o tipo de respuesta, los IDSs se clasifican en Activos o Pasivos. A partir del análisis de eventos y la consecuente detección de un ataque, estos IDS pueden reaccionar de forma **pasiva,** enviando informes a personas u organizaciones que administran listas de eventos de ataques que se encargarán de tomar a futuro las acciones que procedan, o de forma **activa,** lanzando automáticamente respuestas a dichos ataques.

IDS de Respuesta Pasiva

En los IDSs pasivos, una vez detectado el ataque se procede a notificar al responsable de la seguridad de la organización, al usuario del sistema atacado, o a algún organismo externo tipo CERT respecto al ataque efectuado. Es posible también avisar al administrador del sitio desde el cual se produjo el ataque informándole de los detalles del ataque perpetrado. En este último caso es factible que el atacante monitorice el correo electrónico de esa organización, o que haya usado una IP falsa para su ataque.

IDS de Respuesta Activa

Los IDSs activos, también conocidos como Sistemas de Prevención de Intrusión (IPS, *Intrusion Prevention System*) se plantean como una interesante alternativa reactiva que busca solucionar los ataques en las redes informáticas. Los IDSs activos reaccionan cuando se detecta cierto tipo de intrusión, gracias a la escritura de reglas personalizadas en cortafuegos o en enrutadores. Esas reglas (bloqueos o contra ataques) se escriben de forma automática una vez se ha detectado el problema.

Los IDSs activos actualmente no poseen motores inteligentes, sino que se basan en mecanismos de respuesta muy simples. Por eso es peligroso emplear dichos sistemas, porque pueden bloquear o restringir el acceso a recursos del sistema informático debido a falsos positivos o a análisis erróneos de los datos de entrada. Un ejemplo de IDS activo de libre distribución es *Inline Snort* de *Jed Haile*, un módulo gratuito para el NIDS *Snort*.

1.2.3.5. IDS según el tiempo de predicción

En [2] se difinen los IDS basados en el tiempo de predicción, se puede distinguir entre los IDS *on-line* que detectan intrusiones en tiempo real y los IDS *off-line* que por lo general primero almacenan los datos monitorizados y luego los analizan, haciendo uso de la técnica de procesamiento por lotes, para detectar señales de intrusión.

IDS *on-line*

Intentan detectar intrusiones en tiempo real o casi en tiempo real, por ello también se denominan IDS en tiempo real. Operan con fuentes de información de flujos de datos continuos y tienen la capacidad de analizar los datos, mientras que las sesiones están en curso (por ejemplo, sesiones de red para la detección de intrusiones en la red, las sesiones de inicio de sesión *-login-* para la detección de intrusiones basado en host). Los IDS en tiempo real deben dar la alarma en cuanto se produce un ataque, por lo que la acción afecta el progreso del ataque detectado. La mayoría

de los IDS comerciales demandan este tipo de capacidades de procesamiento continuo. Un ejemplo se muestra en [82] y [83].

IDS *off-line*

Los IDS *off-line* realizan un análisis posterior de los datos auditados. Este método de análisis de datos auditados es común entre los analistas de seguridad que, a menudo, examinan el comportamiento de la red, así como el comportamiento de los diferentes atacantes, en modo *off-line* (por lotes). Muchos de los primeros IDS basado en *host* utilizaron este esquema de tiempo, empleando pistas de auditoría del sistema operativo que se registran como archivos [84] y [85].

El análisis *off-line* también se realiza a menudo utilizando herramientas estáticas que analizan la instantánea (*snapshot*) del ambiente (por ejemplo, *host* versus entorno de red), buscando vulnerabiilidades, errores de configuración y evaluando el nivel de seguridad de la configuración del entorno actual. Ejemplos de estas herramientas incluyen COPS [86] y Tiger [87] para entornos de host, y Satan [88] y CyberCop Scanner [89] y [90] para redes. Los detectores de virus también pertenecen a las herramientas estáticas y escanean los discos en busca de patrones que coincidan con virus conocidos. Aunque las herramientas estáticas son muy populares y ampliamente utilizadas por los administradores de sistemas, no son normalmente suficientes para garantizar una alta seguridad [91].

Las herramientas estáticas también pueden ser diseñadas específicamente para la investigación activa de vulnerabilidades a través de Internet. Por ejemplo Tripwire [92] o ATP [93] se pueden utilizar para controlar un conjunto designado de archivos y detectar las intrusiones que explotaban aplicaciones vulnerables antiguas. Estas intrusiones también deben ser identificadas y comunicadas al administrador del sistema como agujeros de seguridad potenciales usando otras herramientas como COPS [86] o Tiger [87].

1.2.3.6. IDS según su implementación

La forma en que se implementa un IDS influirá en su funcionamiento y en el efecto que produce sobre el recurso supervisado. Ya se base en *host* o en red, un IDS no debe obstaculizar el rendimiento del *host* o la red de tal forma que afecte a la calidad de servicio que reciben los usuarios. Según [24] los IDS se clasifican de acuerdo a su implementación en basados en Inteligencia Artificial (como las redes neuronales y los sistemas expertos), basados en cálculo computacional (como lenguajes para fines específicos y bayesiano) y basados en conceptos biológicos (como sistemas inmunológicos y genéticos).

Basados en Inteligencia Artificial

- **IDS basados en Redes Neurales**

  Las redes neuronales artificiales (ANN - *artificial neural networks*), son una técnica que se puede situar dentro de la Inteligencia Artificial, que permite la identificación y clasificación de actividades de comunicación basadas en fuentes de datos incompletas y limitadas. Aunque el tema será abordado con mayor profundidad en el Capítulo 2, a continuación se realiza una sucinta descripción de su funcionamiento.

- **IDS basados en Sistemas Expertos**

Entre los IDS basados en sistemas expertos, están los basados en reglas, que proporcionan una respuesta coherente para decisiones, procesos y tareas repetitivas. Ellos sostienen y mantienen niveles significativos de información, reduciendo los costos de entrenamiento, centralizando el proceso de toma de decisiones, y reduciendo el tiempo necesario para resolver problemas. También reducen la cantidad de errores humanos y pueden revisar las transacciones que los expertos humanos pueden pasar por alto. Sin embargo, debido a que se basan en reglas, necesitan actualizaciones frecuentes. Por otra parte, la adquisición de estas reglas es un proceso tedioso y propenso a errores. También carecen de sentido común, creatividad y capacidad para adaptarse a entornos cambiantes [94].

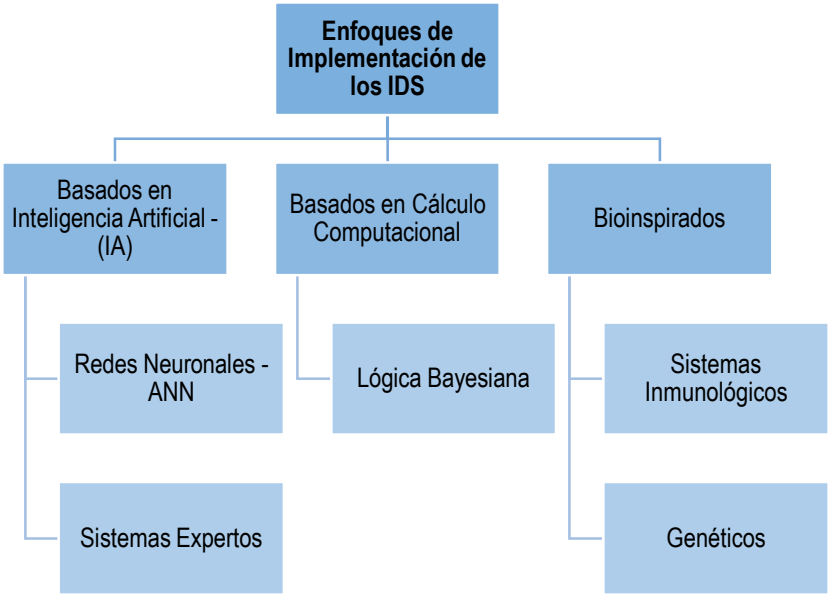

Figura 1.9 Enfoques de clasificación de los IDS de acuerdo a su implementación

En [95] se presentan un enfoque para detectar intrusiones en tiempo real basado en el análisis de transición de estado. El modelo se representa como una serie de cambios de estado que conducen de un estado seguro inicial a un estado objetivo comprometido. Los gráficos de transición de estado identifican los requisitos y los describen como eventos críticos que deben ocurrir para que se complete con éxito el ataque. La herramienta de análisis de transición de estado (STAT) es un sistema experto basado en reglas que se generan a partir de los diagramas. Los autores desarrollan USTAD que es un prototipo específico en UNIX de este sistema experto.

Basados en cálculo computacional

- **IDS basados en el uso de Lógica Bayesiana**

La lógica bayesiana, tema que se abordará con mayor profundidad en un capítulo posterior precisamente por ser parte integral de la solución propuesta en este estudio, es una rama de la lógica que se aplica a la toma de decisiones y la estadística inferencial y que se ocupa de la inferencia de probabilidad. La inferencia bayesiana utiliza el conocimiento de los eventos

anteriores para predecir eventos futuros. Mediante un IDS basado en lógica bayesiana, se puede establecer un modelo estadístico óptimo para ajustar los datos experimentales inicialmente obtenidos de acuerdo a lo planteado en [94].

IDS bioinspirados

- **IDS basados en Sistemas Inmunológicos**

  Los IDS basados en Sistemas Inmunológicos imitan la capacidad del sistema inmunológico innato para detectar intrusiones y detenerlas. Tienen la capacidad de detectar nuevos tipos de intrusiones que no se han visto antes y no requieren un experto humano para indicar si la intrusión es realmente cierta. En [98] se implementa un sistema nativo artificial (NAIS), basado en un sistema inmunológico artificial aplicado a la protección de redes informáticas.

- **IDS basados en Algoritmos Genéticos**

  Los algoritmos genéticos pertenecen a la categoría de algoritmos evolutivos y se utilizan para resolver problemas de optimización. Su funcionamiento se basa en generar una población inicial de individuos (también denominados cromosomas) que irán evolucionando en cada generación. Las nuevas generaciones de cromosomas mejoran tras aplicar operadores genéticos de cruce y mutación hasta llegar a un punto de convergencia en la solución de un determinado problema. Los algoritmos genéticos han surgido como una herramienta muy eficaz en aplicaciones de minería de datos, y dado que un IDS necesita efectuar la clasificación de un *dataset* en tráfico normal y ataques, los algoritmos genéticos son una herramienta adecuada para esta tarea, por la alta precisión que evidencian en relación a su capacidad clasificatoria.

  En [99] se propone un IDS denominado GANIDS, basado en un algoritmo genético. Su arquitectura está basada en un esquema de cruce de dos puntos, con una estrategia de reemplazo de estado estacionario, lo que posibilita el remplazo parcial de la población y una estrategia de mutación está basada en probabilidad. En el estudio se compara GAINDS con PAYL [100], evaluando las tasas de detección respecto a las conexiones con tráfico http, ftp, telnet y smtp. Se observa que GANIDS supera a PAYL en cada protocolo. Por otra parte, en [101] se presenta un IDS basado en Máquinas de Soporte Vectorial – SVM y algoritmos genéticos. El sistema está controlado por un programa cliente y un programa de servidor. El programa cliente es responsable de registrar el comportamiento de los usuarios en un archivo denominado de origen de datos, éste archivo se transmite luego al programa servidor, que lo enviará a la SVM para ser analizado. El resultado del analisis, se transmite de nuevo al programa cliente y el programa cliente luego decide sobre el curso de las acciones a tomar. Además, el algoritmo genético se utiliza para optimizar la información a extraer a partir del archivo de origen de datos para que el tiempo de detección puede ser optimizado.

Para valorar la eficacia y eficiencia funcional de los IDS y tener un punto de referencia para evaluar las diferentes propuestas presentadas por la comunidad científica es preciso definir métricas de desempeño. A continuación se consideran las más relevantes.

1.2.4. Evaluación de los IDS

Como se ha mencionado anteriormente, los IDS basados en firmas con detección de abusos analizan el tráfico de datos, que es comparado con una base de datos de firmas, si coincide con alguna de ellas, se considerará como ataque. En los IDS basados en anomalías se evaluarán los comportamientos inusuales en los *host* y en el tráfico de la red, para ello se construyen perfiles mediante asociación de patrones, para representar el comportamiento normal de los usuarios, *hosts* o conexiones de red.

Hasta el momento no ha sido posible crear un IDS que distinga en su totalidad, el tráfico normal y el malicioso, debido a la gran variedad de ataques existentes y al hecho de que cada día se generan más ataques desconocidos producto del comportamiento de nuevas aplicaciones y del uso de malas prácticas. Por ello, en el momento en que un IDS analiza el tráfico de datos podría tomar decisiones incorrectas respecto a su clasificación, identificando ataques como tráfico normal o viceversa. Para evaluar el desempeño de un IDS se han identificado cuatro métricas asociadas a la naturaleza del evento o clase actual y el estado de la detección o clase predicha, esas métricas son verdadero positivo (VP - ataque correctamente identificado como ataque), verdadero negativo (VN - tráfico normal correctamente identificado como tráfico normal), falso positivo (FP - tráfico normal identificado incorrectamente como ataque) y falso negativo (FN - ataque identificado incorrectamente como tráfico normal), definidas en [67]. La Figura 1.10 se denomina matriz de confusión y ha sido definida en [27] y [66]. Esta tabla muestra la definición de las métricas mencionadas. Entendiendo que los VN y VP corresponden a un funcionamiento correcto de los IDS, en cambio los FN y FP corresponden a un funcionamiento erróneo de los IDS.

Obviamente, un IDS es más eficiente cuando acierta en mayor medida respecto a la clasificación del tráfico de datos (VN y VP) y presenta baja tasa de fallos en dicha clasificación (FP y FN). Para evaluar formalmente la eficiencia de un IDS es necesario conocer la probabilidad de detectar un ataque (tasa de verdaderos positivos) y de emitir una falsa alarma (tasa de falsos positivos), estas dos métricas serán definidas a continuación. A partir de estos dos valores se podrá construir la curva ROC (*Receiver Operating Characteristic*) que permitirá valorar el IDS. En la curva ROC de la Figura 1.11, tomada de [66] y conceptualizada en [67] y [102], se aprecia que el IDS perfecto será aquel que detecte todo el tráfico de forma correcta, no generando ninguna falsa alarma.

		Predicción de Clase	
		Clase negativa (Tráfico Normal)	Clase Positiva (Ataque)
Clase actual	Clase negativa (Tráfico Normal)	Verdadero Negativo (VN)	Falso Positivo (FP)
	Clase positiva (Ataque)	Falso Negativo (FN)	Verdadero Positivo (VP)

Figura 1.10 Matriz de confusión

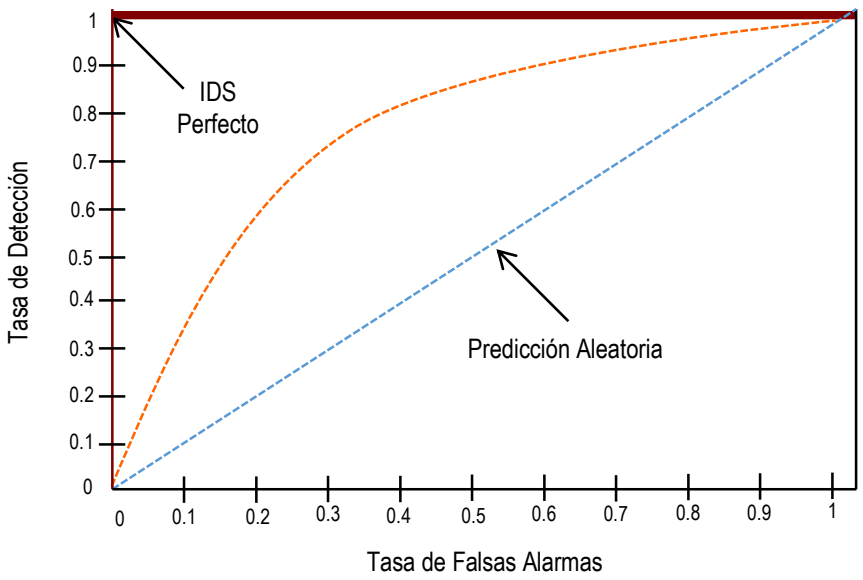

Figura 1.11 Curva ROC

1.2.4.1. Métricas de desempeño

La sensibilidad (*sensitivity*) y especificidad (*specificity*) son medidas estadísticas del desempeño de una prueba de clasificación binaria, también conocida en estadística como la función de clasificación. La sensibilidad (también llamada tasa de recuperación - *recall rate*) mide la proporción de "verdaderos positivos" que son correctamente identificados como tales. La especificidad mide la proporción de "verdaderos negativos" que se han identificado correctamente.

A partir de lo anterior, entendiendo que la sensibilidad es la capacidad de una prueba para identificar resultados VP (también denominada Tasa de Verdaderos Positivos o Tasa de Detección – DR o recall) y la especificidad es la capacidad de la prueba para identificar los resultados negativos (también denominada Tasa de Verdaderos Negativos). Se tiene que:

$$Sensibilidad = \frac{VP}{VP + FN} \tag{1.1}$$

$$Especificidad = \frac{VN}{VN + FP} \tag{1.2}$$

Si una prueba tiene una sensibilidad o Tasa de Detección (DR – por su sigla en inglés) del 100% significa que la prueba reconoce todos los VP, es decir, todos los ataques se detectan efectivamente como ataques. En contraste una prueba que tiene un 100% de especificidad, detecta todos los VN, es decir, todo el tráfico normal es correctamente identificado como tal.

Otras dos métricas complementarias son la Tasa de Falsos Positivos (TFP), también denominada Tasa de Falsas Alarmas (FAR – por sus siglas en inglés) y la Tasa de Falsos Negrativos (TFN).

$$TFP = \frac{FP}{VN + FP} = 1 - Especificidad \tag{1.3}$$

$$TFN = \frac{FN}{VP + FN} = 1 - Sensibilidad \tag{1.4}$$

Por otra parte, en los sistemas de medición, la exactitud (*accuracy*) [103] es el grado de cercanía de las mediciones de una cantidad al valor de la magnitud real y la precisión, reproducibilidad o repetibilidad (*precision*) [103] es el grado en que las mediciones repetidas en condiciones iguales muestran los mismos resultados. Un sistema de medición se denomina válido si es a la vez exacto y preciso, sin embargo un sistema de medición puede ser exacto pero no preciso, preciso pero no exacto, ninguno de los dos, o ambos. La Figura 1.12 describe estas dos métricas.

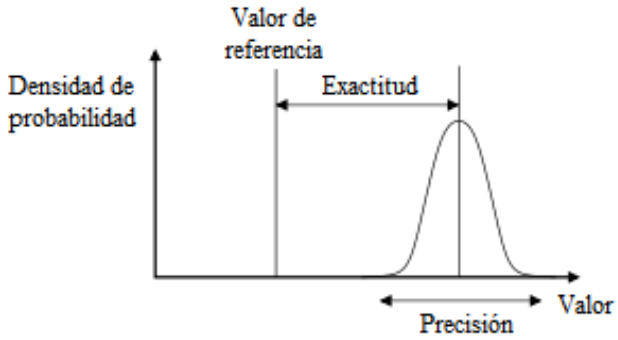

Figura 1.12 Exactitud y Precisión

La exactitud es la proporción de resultados verdaderos (tanto verdaderos positivos como verdaderos negativos) en la población, Una exactitud del 100% significa que los valores medidos son exactamente los mismos que los valores dados. La exactitud es definida en la ecuación (1.5).

$$Exactitud = \frac{VP + VN}{VP + FP + FN + VN} \tag{1.5}$$

Por otro parte, el valor de la precisión o valor predictivo positivo se define como la proporción de VP contra todos los resultados positivos, definida en:

$$Precisión = \frac{VP}{VP + FP} \tag{1.6}$$

Además, tal como se define en [104] el $F_{measure}$ es definido como la media harmónica del *recall* (tasa de detección o sensibilidad) y la precisión, de acuerdo a la siguiente ecuación:

$$F_{measure} = \frac{2 * Sensibilidad * Precisión}{Sensibilidad + Precisión} \tag{1.7}$$

De todas las métricas de rendimiento anteriormente definidas, las más populares son la tasa de detección (DR) o sensibilidad, junto con la tasa de falsas alarmas (FAR) o Tasa de Falsos Positivos (TFP). Un IDS debe tener un alto DR y un bajo FAR. Otras combinaciones de uso general incluyen precisión y sensibilidad o recall, o la sensibilidad y la especificidad. En la Tabla 1.9, se muestra la relación existente entre las métricas de desempeño de los sistemas de clasificación de acuerdo con los resultados y condiciones.

Tabla 1.9 Relación de métricas de desempeño de un clasificador binario

		Condición (según lo determinado por el "Gold Standard")		
		Condición **Positiva**	Condición **Negativa**	
Resultado de la prueba	Resultados **Positivos** de la prueba	**Verdadero Positivo**	Falso Positivo (error tipo I)	**Valor predictivo positivo (precisión)=** $\frac{\sum VerdaderosPositivos}{\sum ResultadosPositivos}$
	Resultados **Negativos** de la prueba	Falso Negativo (error tipo II)	**Verdadero Negativo**	**Valor predictivo negativo=** $\frac{\sum VerdaderosNegativos}{\sum ResultadosNegativos}$
		Sensibiidad= $\frac{\sum VerdaderosPositivos}{\sum CondiciónPositiva}$	**Especificidad=** $\frac{\sum VerdaderosNegativos}{\sum CondiciónNegativa}$	**Exactitud=** $\frac{\sum Verdaderos}{\sum ResultadosVerdaderosyNegativos}$

Las métricas anteriormente detalladas, son utilizadas para validar la funcionalidad de un IDS, mediante el análisis de registros de datos o conexiones de red. Típicamente tales conexiones se representan como paquetes de capa tres del stack TCP/IP. Sin embargo, se ha documentado un *dataset* denominada NSL-KDD [108] y cuya constitución se describe posteriormente en la sección 1.4, que contiene registros de conexiones de red, cada uno de los cuales ocupa 100 bytes, distribuidos en 41 atributos. A partir de la implementación de técnicas de clasificación sobre tales registros-atributos se puede determinar si la conexión es normal o es algún tipo de ataque de las categorías que ya han sido definidas en [2] y descritas en la sección 1.1.1 (DoS - *Denial Of Service*, R2L - *Remote to Local*, U2R - *Usert to Root* y *Probing* - Vigilancia y otros tipos de sondeo de redes).

1.3 ATRIBUTOS O CARACTERÍSTICAS

A partir de lo contemplado en [109] se puede indicar que existe una clasificación de los atributos, en relación al tipo de evaluación que posibilitan las conexiones. Los **atributos de contenido** permiten evaluar el número de intentos de acceso fallidos. Los atributos de **mismo *host*** tienen en cuenta sólo las conexiones en los dos últimos segundos que tengan el mismo destino que la conexión actual y estadísticas relacionadas con el protocolo y los servicios. Los atributos de **mismo servicio** examinan sólo las conexiones en los dos últimos segundos que tienen el mismo servicio que la conexión actual. Tanto los atributos de **mismo *host*** como los de **mismo servicio** posibilitan la evaluación del tráfico de las conexiones en el tiempo.

Dado que los ataques de *Probing* escanean los puertos con un intervalo de tiempo mucho mayor de dos segundos (posiblemente una vez por minuto). Se debe hacer una clasificación de los registros por *host* de destino, produciendo una serie de atributos denominados **tráfico basado en *host***.

Los ataques de las categorías R2L y U2R generan registros de conexión en los cuales los atributos no generan patrones secuenciales frecuentes, debido a que tales tipos de ataques solo requieren de una única conexión, a diferencia de los ataques de las categorías DoS y *Probing* que requieren muchas conexiones con un mismo host en un período muy corto de tiempo, lo que permite evaluar patrones secuenciales frecuentes. En la Tabla 1.14 se listan los atributos que permiten caracterizar los tipos de conexión, clasificados por atributos básicos, especiales, y aquellos que hacen posible la evaluación con una ventana de tiempo de dos segundos.

Tabla 1.14 Atributos básicos de las conexiones TCP

Atributos básicos de las conexiones TCP		
Atributo	**Descripción**	**Tipo**
Duration	Longitud (número de segundos) de la conexión.	Continuo
protocol_type	Tipo de protocolo (tcp...).	Discreto
Service	Tipo de servicio de destino (HTTP, Telnet, SMTP...).	Discreto
src_bytes	Número de bytes de datos de fuente a destino.	Continuo
dst_bytes	Número de bytes de datos de destino a la fuente.	Continuo
Flag	Estado de la conexión (SF, SI, REJ...).	Discreto
Land	1 si la conexión corresponde mismo host/puerto; 0 de otro modo.	Discreto
wrong_fragment	Número de fragmentos erróneos.	Continuo
Urgent	Número de paquetes urgentes.	Continuo

Los atributos anteriormente descritos pueden ser de dos tipos: numéricos (real o entero) o simbólicos (valores discretos a partir de una lista especificada).

A partir de lo anterior, se ha podido comprender que los sistemas de detección y prevención de intrusos no son una solución completa a la identificación y prevención de ataques a una red informática, más bien complementan, en gran medida, las restricciones que presentan otros mecanismos de seguridad como los cortafuegos o VPNs, analizando el tráfico de la red, con el propósito de identificar el ataque (IDS pasivo), o reaccionar contra el ataque (IDS activo). Para tal detección se requiere evaluar las diferentes características que constituyen las respectivas conexiones que se refieren a los ataques.

Tabla 1.15 Atributos Especiales

Atributos Especiales		
Atributo	**Descripción**	**Tipo**
Hot	Número de indicadores “hot”.	Continuo
num_failed_logins	Número de intentos de acceso fallidos.	Continuo
logged_in	1 si acceso exitoso; 0 de otro modo.	Discreto
num_compromised	Número de condiciones “sospechosas”.	Continuo
root_shell	1 si se obtiene superusuario para acceso a root; 0 de otro modo.	Discreto
su_attempted	1 si se intenta el comando “su root”; 0 de otro modo.	Discreto
num_root	Número de accesos a root.	Continuo
num_file_creations	Número de operaciones de creación de ficheros.	Continuo
num_shells	Número de shell prompts.	Continuo
num_access_files	Número de operaciones de control de acceso a ficheros.	Continuo
num_outbound_cmds	Número de comandos de salida en una sesión ftp.	Continuo
is_hot_login	1 si el login pertenece a la lista “hot”; 0 de otro modo.	Discreto
is_guest_login	1 si el acceso es un “guest” login; 0 de otro modo.	Discreto

Tabla 1.16 Atributos con ventana de dos segundos

Atributos con ventana de dos segundos		
Atributo	**Descripción**	**Tipo**
Count	Número de conexiones a la misma máquina que la conexión actual en los dos últimos segundos.	Continuo
Los siguientes atributos se refieren a las conexiones de mismo host.		
serror_rate	Porcentajes de conexiones que tienen errores “SYN”.	Continuo
rerror_rate	Porcentaje de conexiones que tienen errores “REJ”.	Continuo
same_srv_rate	Porcentaje de conexiones con el mismo servicio.	Continuo
diff_srv_rate	Porcentaje de conexiones con diferentes servicios.	Continuo
srv_count	Número de conexiones al mismo servicio que la conexión actual en los dos últimos segundos.	Continuo
Los siguientes atributos se refieren a las conexiones de mismo servicio.		
srv_serror_rate	Porcentaje de conexiones que tienen errores “SYN”.	Continuo
srv_rerror_rate	Porcentaje de conexiones que tienen errores “REJ”.	Continuo
srv_diff_host_rate	Porcentaje de conexiones a diferentes hosts.	Continuo

Tabla 1.17 Atributos de tipo simbólico y sus respectivos valores

Atributos de tipo simbólico y sus respectivos valores	
Atributo	**Valores**
protocol_type	Tcp, udp, icmp.
Service	http, mtp, smtp, finger, domain, domain_u, auth, telnet, ftp, eco_i, ntp_u, ecr_i, other, private, pop_3, ftp_data, rje, time, link, remote_job, gopher, ssh, name, whois, login, imap4, daytime, ctf, nntp, shell, IRC, nnsp, http_443, exec, printer, icmp, efs, courier, uucp, klogin, kshell, echo, discard, systat, supdup, iso_tsap, hostnames, csnet_ns, pop_2, sunrpc, uucp_path, netbios_ns, netbios_ssn, netbios_dgm, sql_net, vmnet, bgp, Z39_50, ldap, netstat, urh_i, X11, urp_i, pm_dump, tftp_u, tim_i, red_i .
Flag	SF, S1, REJ, S2, S0, S3, RSTO, RSTR, RSTOS0, OTH, SH.
class (ataque)	back, buffer_overflow, ftp_write, guess_passwd, imap, ipsweep, land, loadmodule, multihop, neptune, nmap, normal, perl, phf, pod, portsweep, rootkit, satan, smurf, spy, teardrop, warezclient, warezmaster, snmpgetattack, named, xlock, xsnoop, sendmail, saint, apache2, udpstorm, xterm, mscan, processtable, httptunnel, worm, mailbomb, sqlattack, snmpguess, ps .

Los investigadores siguen valorando diferentes metodologías y técnicas con el objeto de desarrollar una solución IDS/IPS cada vez mejor, para ello se requiere un ambiente que permita simular el tráfico de red de la forma más real posible. Razón por la cual organizaciones tan relevantes como el MIT (*Massachusetts Institute of Technology*) y DARPA (*Defense Advanced Research Projects Agency*) han simulado diferentes escenarios, alimentando colecciones de datos, con el propósito de dotar a los investigadores de una base de datos de tráfico de red, que sirva de apoyo a futuras investigaciones. A continuación se hace una caracterización de los *dataset* más comunmente utilizados en procesos relativos a la detección de intrusos.

1.4 DATASETS DE SOPORTE A IDS

El Grupo de Tecnología de Sistemas de Información (IST), del Laboratorio Lincoln del Instituto Tecnológico de Massachusetts (LL-MIT), con la cooperación de la Agencia de Proyectos de Investigación Avanzada de Defensa (DARPA ITO) y el Laboratorio de Investigación de las Fuerzas Aéreas (AFRL/SNHS), recopiló el primer *dataset* que contiene tráfico de red con una variada colección de conexiones. El *dataset* se utiliza para la evaluación de la eficiencia de los sistemas de detección de intrusos en redes informáticas. Los criterios medibles son la probabilidad de detección y la probabilidad de falsas alarmas del respectivo sistema testeado.

Los *dataset* publicados por LL-MIT en su web oficial, son los resultados de las evaluaciones en detección de intrusiones efectuadas por DARPA en 1998 y 1999. También se encuentran experimentos dirigidos a escenarios específicos realizados en 2000. El LL-MIT distribuye libremente los *dataset*, la documentación, publicaciones, evaluaciones de resultados y herramientas de software relacionadas, disponibles en [110].

El *dataset* DARPA 1998 contiene un conjunto de ataques realistas, integrando un conjunto de conexiones normales, lo cual suministra la base de datos que permite evaluar las falsas alarmas y las tasas de detección del IDS. Para construir este dataset se efectuaron dos evaluaciones, una *off-line* y otra en tiempo real. La primera consta de tráfico de red y *logs* de auditoría recogidos en una red de simulación, para la segunda se insertaron sistemas de detección de intrusión en el

banco de pruebas de la red AFRL con la intención de identificar sesiones de ataque en medio de actividades normales, en tiempo real.

La red física usada para la simulación incluye una subred interna y una externa separadas por un enrutador. La externa incluye dos estaciones de trabajo que simulan gateways en un Internet exterior virtual. Una estación de trabajo simula varias estaciones usando modificaciones del software cliente del kernel de Linux, proporcionados por el grupo Air Force ESC. Un gateway controla a cien estaciones y otro a miles de sitios web cuyo contenido se actualiza diariamente. La subred interna incluye máquinas críticas de muchos tipos (Linux, Solaris, Sun OS) y un gateway para muchas otras estaciones de trabajo internas. Los datos fueron recogidos desde un host interno que ejecuta Solaris y desde un sniffer externo.

En el escenario anteriormente descrito se envían emails, broadcasts, correo simple, y listas de servidores de dominio, así como tráfico ftp a través de la descarga de usuarios de variedad de código original y archivos de documentación de sitios ftp anónimos, tanto internos como externos. Se registró además la actividad de seis usuarios con identidades específicas profesionales (programador, secretario, administrador de sistema y gerente), los cuales diariamente realizaron sesiones telnet ejecutando tareas, accediendo con su identidad; algunas de esas acciones pueden considerarse actividades anómalas. Tales actividades comprenden denegación de servicios, acceso desautorizado, transición desautorizada, obtención de privilegios de root por un usuario sin privilegios (para esto fue necesario efectuar vigilancia y testeo), y en general diferentes comportamientos anómalos de usuarios.

A partir de todos los datos recolectados se organizaron distintos subconjuntos de datos que componen el *dataset* DARPA 1998, tales como datos de ejemplo, cuatro horas de subconjuntos de datos de entrenamiento, datos de entrenamiento (contienen siete semanas de ataques basados en red en medio de datos en segundo plano, normales) y datos de test (contiene dos semanas de ataques basados en red en medio de actividad normal en segundo plano).

El *dataset* DARPA 1999, al igual que su predecesor, está constituido por una evaluación *off-line* y una evaluación en tiempo real, basándose en los mismos principios que en el conjunto de datos del año anterior e incluyendo adicionalmente las siguientes características: ataques y tráfico desde ordenadores que ejecutan *Windows NT*, ataques en la red interna, archivos de sistema *dump* que proporcionan importantes componentes desde sistema de ficheros de cinco víctimas cada noche, incluyendo logs de auditoría de *Windows NT* y archivos de *sniffing* que proporcionan datos de *sniffing* de la red interna.

DARPA 1999 centra la evaluación de actividades de estaciones de trabajo *UNIX*, *Windows NT* y a partir de los siguientes eventos: Denegación de Servicios (DoS), Remoto a Local (R2L), Usuario a Root (U2R) y acceso no autorizado o modificación de datos en un *host* local o remoto.

Estos ataques ocurren en el contexto de uso normal de computadores y redes en una base militar. La organización de los datos utiliza un esquema similar al seguido por el *dataset* DARPA de 1998 con algunas modificaciones (no hay datos de ejemplo ni subconjuntos de datos de entrenamiento). Quedando el *dataset* DARPA 1999 constituido por datos de entrenamiento (tres semanas de ataques, teniendo en cuenta que la primera y la tercera semana no contienen ataques, la segunda semana contiene un subconjunto selecto de ataques que van desde los ataques de 1998 a otros ataques nuevos), datos de *test* (dos semanas de ataques basados en red en medio de actividad

normal en segundo plano).

En DARPA 2000 los datos se obtuvieron a partir de varios escenarios: (1) dos escenarios compuesto de múltiples sesiones de red y auditoría agrupadas en cinco fases de ataques, en las cuales el atacante testea la red, interrumpe la vulnerabilidad de un host ejecutando Solaris, instala el software del troyano mstream DDoS, y lanza un ataque de DDoS en un servidor del sitio desde el host comprometido y (2) un tercer escenario en el que se almacenan las trazas recogidas del tráfico de un día y los ataques que afectan a una máquina windows NT.

El *dataset* NSL-KDD es una colección de datos construido con el objeto de solventar los problemas que presenta el conjunto KDD'99 [111], pese a no ser una representación perfecta de los datos reales, debido a que no contiene conjuntos de datos públicos de los IDS, sin embargo, demuestra mucha utilidad al ser aplicado como un conjunto de datos de referencia eficaz para ayudar a los investigadores en el proceso de comparación de diferentes métodos de detección de intrusos.

Las mejoras que presenta el NSL-KDD respecto a sus predecesores, son las siguientes:

- La colección de datos no incluye registros redundantes; por lo tanto los clasificadores no realizarán correcciones con tanta frecuencia.
- No existen registros duplicados en los conjuntos de pruebas propuestos; por lo tanto, el rendimiento del aprendizaje no está sesgado por los métodos que tienen mejores tasas de detección en los registros frecuentes.
- El número de registros seleccionados de cada grupo de nivel de dificultad es inversamente proporcional al porcentaje de registros en el conjunto original de datos KDD. Como resultado, las tasas de clasificación de los distintos métodos de aprendizaje de máquinas varían en un rango más amplio, lo que hace que sea más eficiente para tener una evaluación precisa de las diferentes técnicas de aprendizaje.

En [108] se encuentran los archivos de datos del NSL-KDD tanto en formato .txt como en formato .arff, cuya descripción se aprecia en la Tabla 1.18. Este último formato (*Attribute Relation File Format*) es usado por compatibilidad con el software WEKA (*Waikato Environment for Knowledge Analysis*) [112] con el objeto de poder efectuar el análisis de datos de los *dataset* KDDTrain+ y KDDTest+. WEKA es un entorno de trabajo desarrollado por la Universidad de Waikato (Nueva Zelanda), construido en JAVA y con licenciamiento GPL, que se utiliza para procesos de experimentación de análisis de datos que hagan posible la aplicación, análisis y evaluación, sobre un dataset empleando técnicas relativas al aprendizaje automático.

Aunque existe una amplia variedad de *dataset* los investigadores comúnmente se han decantado por el uso de DARPA NSL-KDD, por las ventajas que ofrece con respecto a otros *dataset* de su misma familia y de otras fuentes. En [113] y [114] se hace un análisis en profundidad de las discrepancias encontradas en KDD cup '99, a partir de lo cual se aprecian las mejoras obtenidas en relación a la eliminación de datos redundantes e inconsistentes en NSL-KDD. La Tabla 1.19 muestra un listado de los *datasets* más destacados en procesos de simulación de sistemas de detección de intrusiones. Esta información complementa el resumen de los conjuntos de datos más populares en el dominio de detección de intrusos de [27].

Tabla 1.18 Archivos del dataset NSL-KDD

Archivo	Descripción
KDDTrain+.arff	El conjunto de datos completo para el entrenamiento (train NSL-KDD), con etiquetas binarias y en formato ARFF.
KDDTrain+.txt	El conjunto de datos completo para el entrenamiento (train NSL-KDD), incluyendo etiquetas de tipos de ataques y el nivel de dificultad, en formato CSV.
KDDTrain+_20Percent.arff	Un subconjunto del 20% del archivo KDDTrain+.arff.
KDDTrain+_20Percent.txt	Un subconjunto del 20% del archivo KDDTrain+.txt.
KDDTest+.arff	El conjunto de datos completo para el test con etiquetas binarias y en formato ARFF.
KDDTest+.txt	El conjunto de datos completo para el test, incluyendo etiquetas de tipos de ataques y el nivel de dificultad, en formato CSV.
KDDTest-21.arff	Un subconjunto del archivo KDDTest+.arff el cual no contiene registros con el nivel de dificultad 21 de un total de 21.
KDDTest-21.txt	Un subconjunto del archivo KDDTest+.txt el cual no contiene registros con el nivel de dificultad 21 de un total de 21.

Tabla 1.19 Datasets utilizados en procesos de simulación de sistemas de detección de intrusiones

Dataset	Patrocinadores - Miembros
Dataset DARPA	• IST-LLMIT (Grupo de Tecnologías de Sistemas de Información - Laboratorio del Instituto de Tecnología de Massachusetts). • DARPA ITO (Agencia de Proyectos de Investigación Avanzada de Defensa - Oficina de Tecnología de la Información). • AFRL/SNHS (Laboratorio de Investigación de las Fuerzas Aéreas).
Datasets USC/ISI ANT Programa PREDICT (Repositorio de Protección para la Defensa de la infraestructura frente a las amenazas informáticas). Proyecto LANDER.	• ANT (Grupo de Investigación de Análisis de Tráfico de Red). • ISI (Instituto de Ciencias de la Información). • USC (Universidad del Sur de California). • Departamento de Ciencias Computacionales de la Universidad • Estatal de Colorado. • Departamento de Ingeniería Eléctrica de USC. • Servicios de Tecnologías de la Información de la USC.
Datasets CAIDA Asociación Cooperativa para el Análisis de Datos en Internet	**Patrocinadores:** ARIN (American Registry for Internet Numbers), CISCO, Endance Measurement Systems, U.S. Departament of Homeland Security, NSF (National Science Fundation). **Miembros:** Digital Envoy, Intel, NTT (Nippon Telegraph and Telephone Corporation), Ripe NCC, University of California San Diego.
Datasets CRAWDAD Comunidad de recurso para archivar datos inalámbricos en Dartmouth	• ACM SIGMOBILE. • Intel Corporation. • Fundación Nacional de Ciencias.
Dataset DRDC Defense Research and Development Canada	• Sección de Operaciones de Información de Red (NIO) de la DRDC Ottawa, Canadá. • Red de Establecimiento para la Investigación y Defensa (DREnet).
NIST SAMATE Reference Dataset Project NIST : National Institute for Standard and Technology SAMATE : Software Assurance Metrics and Tools Evaluation	• Departamento de Estado de EE.UU.
Virtual Dataset Repository	• MERIT NETWORK INC. Programa PREDICT (Protected Repository for the Defense of Infrastructure against Cyber Threats).

Una vez seleccionado el *dataset* más conveniente para el subsiguiente proceso de simulación, es necesario comprender las diferentes fases que ello implica, desde el análisis preliminar de los datos (preprocesamiento), pasando por el entrenamiento y prueba del IDS, y culminando con la clasificación del tráfico. A continuación se detallarán cada una de estas fases.

1.5 FASES DEL PROCESO DE SIMULACIÓN EN DETECCIÓN DE INTRUSIONES

Para obtener una visión panorámica del objeto de estudio, se hace necesario recrear la detección de intrusos mediante procesos de simulación susceptibles de ser evaluados en un ambiente de laboratorio. Ello requiere de la ejecución de varias fases: preprocesamiento del dataset, entrenamiento y prueba (Training and Testing) del clasificador de conexiones de red, y clasificación del tráfico, tal como se ilustra y describe a continuación. Esta metodología experimental ha sido definida en [115] y cada una de las fases han sido identificadas y documentadas tomando como fundamento la dinámica investigativa planteada en [2], [116], [117], [118], [119], [120], [121], [122] y [123].

Fase de preprocesamiento

El preprocesamiento ha sido definido en [102] e implica la ejecución de dos etapas la normalización y la selección o reducción de características. A continuación se hace una breve descripción de estas dos etapas.

En cuanto a la **normalización**, los datos procedentes del *dataset* deberían estar en el rango [0 a 1] o [-1 a 1]. Sin embargo, no lo están, debido a que todas las conexiones en sus 41 características poseen valores continuos, discretos o simbólicos y en diferentes rangos de significación. Con el propósito de estandarizar dichos valores para que puedan ser eficazmente procesados por los algoritmos de aprendizaje automático, se debe hacer un preprocesamiento y normalización de los datos contenidos en las conexiones.

Figura 1.13 Fases del Proceso de Simulación de Detección de Intrusiones

Para la conversión de los símbolos en formato numérico, un código entero se asigna a cada símbolo. Por ejemplo, en el caso de la característica *protocol_type*, se asigna **"0"** a TCP, **"1"** a UDP, y **"2"** a ICMP. De forma similar los nombres de ataque fueron mapeados asignando valores enteros a las cinco categorías asi: **"0"** para tráfico normal, **"1"** para el ataque de sondeo (probe), **"2"** para la Denegación de Servicios (DoS), **"3"** para U2R y **"4"** para R2L. Por otra parte, existen características cuyos valores se extienden en un rango de números enteros muy grande.

Así, *src_bytes* toma valores entre 0 y 1.3 billones igual que *dst_bytes*. Se aplica entonces una escala logarítmica (de base 10) a estas características para reducir el rango de 0.0 a 9.14. Todas las demás características son booleanas, en el rango de [0.0 a 1.0]. Por lo tanto, el escalado no es necesario para estos atributos. En cuanto a la subfase de **selección de características** se debe identificar la técnica de reducción de características a utilizar, debido a que no es conveniente efectuar el entrenamiento y prueba de la red con la totalidad de características dado que ello podría ralentizar considerablemente el procesamiento, sin suponer una mejora significativa en la clasificación del tráfico.

Como ya se ha indicado, la fase de preprocesamiento implica la ejecución de dos etapas la normalización y la selección o reducción de características. Para la implementación de esta última es necesario seleccionar el algoritmo de selección que permita efectuar el proceso de identificación de características, de tal forma que se determine el nivel de relevancia de las mismas, con el propósito de no utilizar la totalidad de las características del dataset, sino sólo aquellas que incidan directamente en el proceso de clasificación y en consecuencia disminuir la carga computacional que agregaría al proceso, el innecesario uso de la totalidad de las características. Esto se estudiará con más detalle en la sección 1.6.

Fase de entrenamiento y prueba

En esta fase, en primera instancia se procede a entrenar el clasificador considerado, a partir del algoritmo de aprendizaje seleccionado y utilizando el *dataset* ya normalizado y reducido a las características relevantes para generar un aprendizaje eficiente. El proceso de entrenamiento se realiza con el *dataset* correspondiente, que no es el mismo a utilizar en el proceso de *test*, producto de ejecuciones iterativas que generarán un modelo de aprendizaje. Debe evitarse el sobre entrenamiento (*overfitting*) del modelo, dado que ello conllevaría a posteriores problemas en el proceso de clasificación.

Fase de clasificación

Una vez el modelo está entrenado, se procede con la fase de clasificación en la que el clasificador determina que tráfico es normal y cual es un ataque, efectuando la subsiguiente clasificación de cada una de las conexiones del *dataset*. Gracias a esto se podrá presentar la información de resumen del proceso, de forma estadística, calculando las métricas de desempeño que van a permitir evaluar el algoritmo de clasificación.

1.6 EXTRACCIÓN Y SELECCIÓN DE CARACTERÍSTICAS

El dataset etiquetado juega un papel importante en el proceso de validación y evaluación de las técnicas de aprendizaje automático en los IDS. Con el fin de obtener una buena precisión en la evaluación de grandes colecciones de datos, constituidas por registros o conexiones (tráfico de intrusiones y tráfico normal), se valora la eficiencia de este proceso, con base en el análisis de las características que constituyen cada conexión. Sin embargo no todas estas características contribuyen a un análisis eficiente del tráfico. Por lo tanto, se requiere la utilización de técnicas de selección y/o extracción de características. Aunque, la extracción y selección de características son procesos diferentes, ambos pueden ser utilizados para obtener un subconjunto discriminativo de características. Por ello se hace necesario una breve descripción de estos procesos.

La ***extracción de características*** aplica una transformación a las características existentes para obtener otras mejores. Un ejemplo de extracción de características a través de un mapeo lineal es el Análisis de Componentes Principales (PCA). Este genera un vector de base ortonormal que indica las direcciones de varianza máxima. Por lo tanto, se proyecta sobre esta base la dispersión maximizada de la muestra. Las muestras de datos proyectados en el espacio de bajo dimensionalidad abarcado por los componentes centrales se utilizan en la tarea de clasificación.

Otra técnica popular de extracción de características que utiliza un criterio de clasificación en lugar de la representación de error (como en PCA), es el Análisis Discriminante Lineal (LDA) [102]. En este caso, las muestras no se pueden representar con precisión por las características proyectadas (es decir, no se minimiza el error de reconstrucción), pero la información discriminativa de clase se mejora. PCA y LDA se han utilizado en problemas clásicos, como el reconocimiento facial, *eigenfaces* [212] y *Fisherfaces* [213], respectivamente. Otras técnicas, como el Análisis de Componentes Independientes (ICA) [214], tienen por objeto encontrar una representación lineal no gaussiana de los datos, de manera tal que los componentes son tan estadísticamente independientes como sea posible.

A diferencia de la extracción, ***la selección de características*** no transforma las características existentes, sólo busca el subconjunto más representativo de ellas. En [115] y [124] se describen tres metodologías para agrupar las diferentes técnicas de selección de características, denominadas: filtrado (*filters*), envoltorios (*wrappers*) y el método embebido (*embedded*). A continuación se hace una descripción de cada una de ellas.

Según [115] **la técnica de selección de características basada en filtrado** (*filter*) se utiliza para encontrar el mejor subconjunto características del conjunto original de características, los métodos de filtrado parecen ser menos óptimos, sin embargo son buenos en la selección de un gran subconjunto de datos, no dependen del algoritmo de clasificación y su costo computacional es menor para grandes conjuntos de datos. **La técnica de selección de características *wrappers*** [115], utiliza la predicción del rendimiento del algoritmo de aprendizaje para la selección de las características. Mejora los resultados de los predictores correspondientes, y logra mejores tasas de reconocimiento, incluso superando a la de los filtros, pero depende del algoritmo de clasificación y para un conjunto de datos grandes, el costo computacional es mayor en el método de *wrapper*. Por último, **los métodos embebidos** [115] se basan en la evaluación del desempeño de la métrica calculada directamente de los datos, sin referencia directa a los resultados de los sistemas de análisis de datos, en ellos hay una unión de las técnicas de selección de características con el proceso de aprendizaje para un algoritmo de aprendizaje determinado. Los métodos *embedded* son menos propensos al sobreajuste (*overfitting*) y también dependen del algoritmo de clasificación.

El estudio realizado en [124] evalúa once métodos de selección de características (siete métodos filter, dos métodos embedded y dos wrapper), aplicados sobre once *datasets* sintéticos y dos *datasets* reales, y probados por cuatro clasificadores (C4.5, NB, IB1 y SVM). A la luz de los resultados que obtuvieron, los autores sugieren el uso del método de selección de características basados en filtrado (ReliefF), ya que es independiente del algoritmo de inducción y es más rápido que los métodos embebidos y de envoltorio, además, tiene una buena capacidad de generalización. La Tabla 1.20, extraida de [124], describe las ventajas, desventajas y ejemplos, de cada una de las categorías de métodos de selección de características.

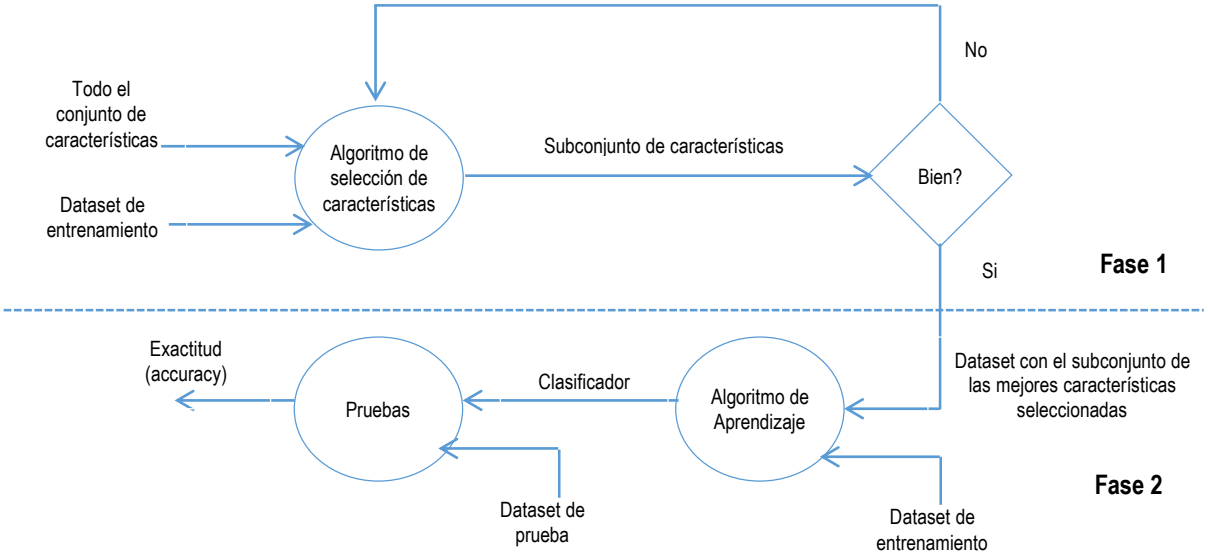

Figura 1.14 Selección de características basada en filters

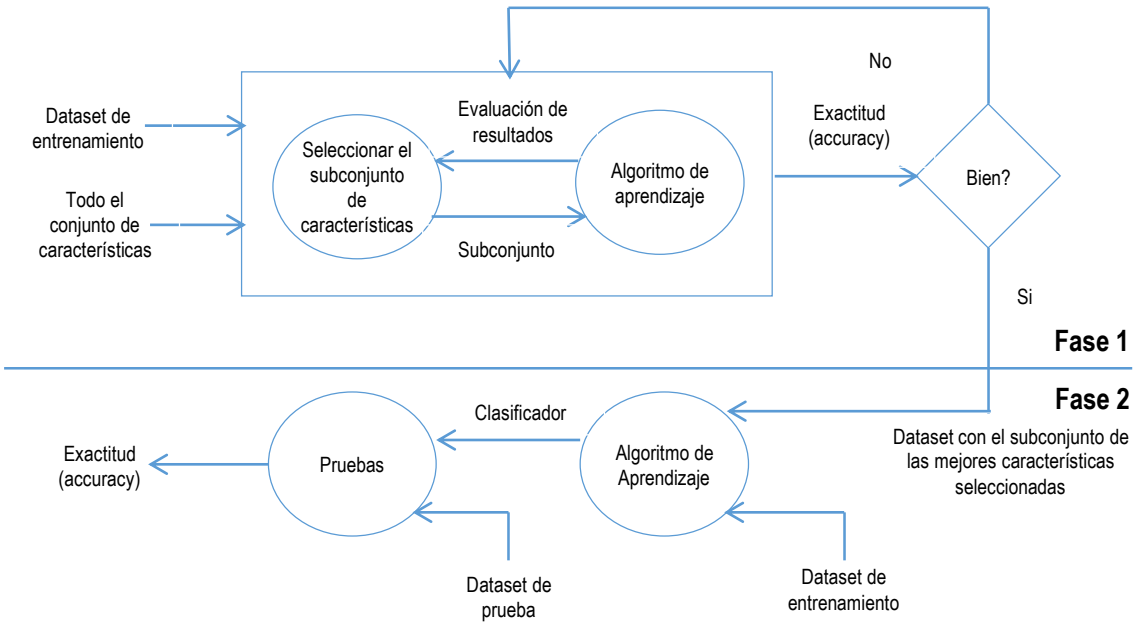

Figura 1.15 Selección de características basada en wrapper

Tabla 1.20 Ventajas e inconvenientes de las categorías de métodos de selección de características

Método	Ventajas	Inconvenientes	Ejemplos
Filters	• Independiente del clasificador. • Mas bajo costo computacional que los de tipo envoltorio. • Rápido. • Buena capacidad de generalización.	No interactúa con el clasificador.	Consistency-based CFS Interact ReliefF Md Information Gain mRMR
Embedded	• Interactúa con el clasificador. • Más bajo costo computacional que el de envoltorio. • Captura dependencias de las características.	La selección depende del clasificador.	FS-Percepton SVM-RFE
Wrapper	• Interactúa con el clasificador. • Captura dependencias de las características.	Es costoso a nivel computacional. Riesgo de sobreajuste. La selección depende del clasificador.	Wrapper-C4.5 Wrapper SVM

1.7 CONCLUSIONES

No existe en la actualidad un IDS/IPS 100% efectivo. Una interesante propuesta seria integrar un IDS basado en abusos (*misuse-based*) con un IDS basados en anomalías, posibilitando de forma distribuida la evaluación del tráfico cuya fuente de información sea tanto el *host* como la red.

Dado que la aplicación de la técnica de selección de características permite la reducción de la dimensionalidad del *dataset*, en el subsecuente proceso de clasificación se deberán efectuar comparaciones con el *dataset* frente al *dataset* reducido, para identificar los mejores resultados. Por ello la correcta elección de la técnica de selección de características es uno de los objetivos fundamentales, lo que requerirá un estudio más detallado de las técnicas de selección, buscando su integración con algoritmos de optimización, con el propósito de alcanzar mejores resultados respecto a las métricas usadas.

En [102] y [210] se hace un acertado análisis de las ventajas de las técnicas de selección y extracción de características, en relación a (1) la representación de los datos en un espacio dimensional mas bajo, evita la *"curse of dimensionality"*, dado que disminuye el número de instancias necesarias para entrenar un clasificador, dicho número crece exponencialmente con la dimensionalidad de los datos, además, evita el sobreajuste (*overfitting*) y mejora el rendimiento en la generalización del clasificador, (2) las características altamente informativas representarán las diferentes muestras de clase, con bastante separación en el espacio de características, mientras

que las muestras similares estarán representadas de forma cercana una de la otra, (3) el uso de menos características mejora la eficiencia computacional, y (4) la visualización de datos es más fácil y más intuitiva en espacios dimensionales inferiores (por ejemplo, en 2D o 3D).

CAPÍTULO 2. MAPAS AUTORGANIZATIVOS JERÁRQUICOS Y ALGORITMOS EVOLUTIVOS MULTIOBJETIVO

2.1 INTRODUCCIÓN

Los Mapas Auto-organizativos son una técnica que comúnmente se utiliza en la solución de problemas de clasificación, como ya se ha indicado los procesos de clasificación requieren de la aplicación de técnicas de extracción o selección de características, de tal forma que permitan la identificación de las características más discriminantes y al permitir usar un número reducido de éstas, consecuentemente mejora la eficiencia computacional del clasificador. En la propuesta que se plantea en esta tesis, se utiliza una técnica de optimización multiobjetivo mediante la aplicación del algoritmo NSGA-II, de tal forma que permita optimizar la escogencia de las características. Por ello en este capítulo se fundamentan tanto las redes neuronalas artificiales como las técnicas de optimización multiobjetivo que soportan la propuesta que aquí se plantea.

En este capítulo se abordan los siguientes temas: en primera instancia los fundamentos de las redes neuronales artificiales - ANN (sección 2.2). A continuación se fundamenta el SOM (sección 2.3), en cuanto a su algoritmo de aprendizaje, campos de aplicación, arquitecturas basadas en SOM y sus inconvenientes. Posteriormente, se fundamenta el GHSOM (sección 2.4), en cuanto a sus principios funcionales, su algoritmo de entrenamiento y la forma como éste soluciona los problemas que presenta SOM. Consecuentemente se fundamenta la optimización como estrategia para definir problemas, mediante modelos y para resolverlos mediante algoritmos (sección 2.5). Se aborda además el tema de los algoritmos de optimización multiobjetivo basados en computación evolutiva (MOEA), utilizados para encontrar múltiples soluciones óptimas de Pareto en una única ejecución de la simulación (sección 2.6). Sin embargo, los MOEA pese a su utilidad han evidenciado una serie de inconvenientes, posteriormente descritos y que son subsanados por el algoritmo NSGA-II (*Non-dominated Sorting Genetic Algorithm-II*) el cual se describe en (sección 2.7). El capítulo culmina con las correspondientes conclusiones.

2.2 FUNDAMENTOS DE LAS REDES NEURONALES ARTIFICIALES

Para comprender el diseño de las redes neuronales, se debe analizar la forma en que el cerebro utiliza sistemas neuronales para el procesamiento de patrones. El componente fundamental de estos sistemas son las neuronas, las cuales, biológicamente hablando, se componen de un cuerpo celular, unas extensiones de ramificación denominadas dendritas (para la recepción de entradas) y un axón (que lleva la salida de una neurona hasta las dendritas de otras neuronas), véase la Figura 2.1 (Tomada de: Biología/Kennet Miller – Prentice Hall – 2004 www.mediateca.cl). La unión entre un axón y una dendrita se llama sinapsis y se cree que para que una neurona lleve a cabo un cálculo simple, la neurona recoge señales en sus sinapsis, las resume y si la intensidad de la señal combinada supera un determinado umbral, la neurona envía una señal de salida (ver Figura 2.2).

En términos simples, el cerebro es una computadora natural-paralela-masiva, compuesta por entre 10 y 100 billones de células cerebrales, cada una de estas neuronas está conectada con otras 10.000 aproximadamente. Las neuronas aparentemente realizan cálculos simples. Sin embargo, el cerebro es capaz de resolver complejos problemas de visión y del lenguaje en 500 milisegundos o menos.

Dado que el tiempo de respuesta de una sola neurona está en el rango de milisegundos y teniendo en cuenta los retardos de propagación entre neuronas. Esto le da a sistemas neuronales un procesamiento de 100 pasos para realizar una tarea compleja de procesamiento de la visión o de la voz, que tomaría en un ordenador, miles de millones de pasos de procesamiento. La investigación en redes neuronales se inicia con la implementación de modelos simples y sólo se añade complejidad cuando el modelo resulta insuficiente. Este enfoque ha sido clave para avanzar en este ámbito de trabajo.

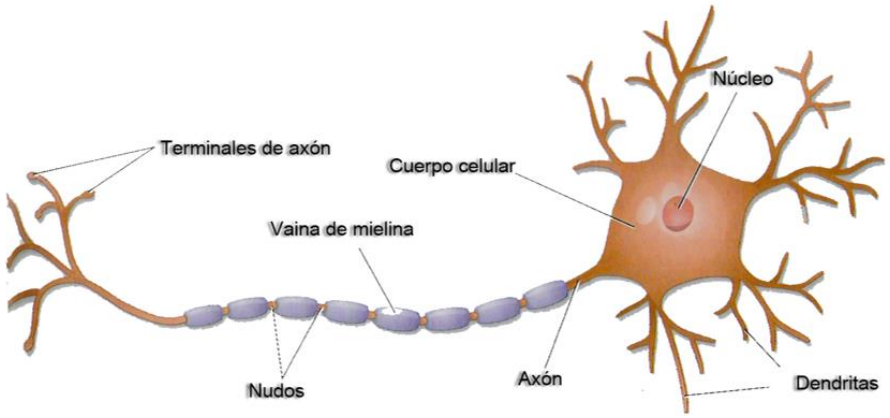

Figura 2.1 Estructura Biológica de una Neurona

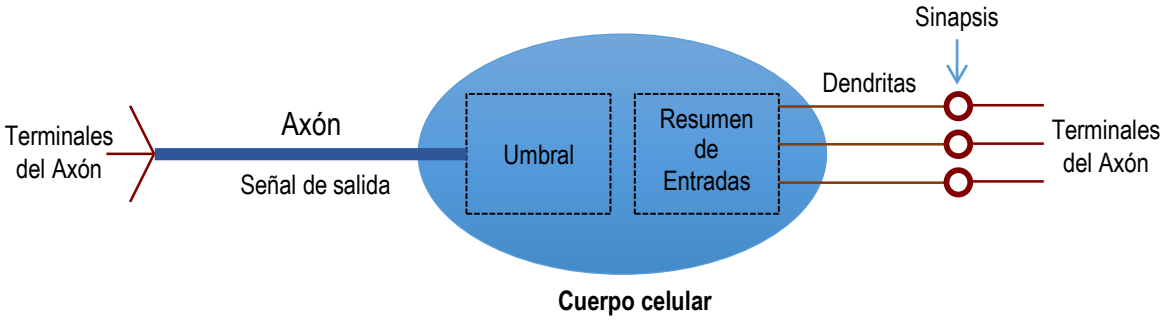

Figura 2.2 Diagrama de bloques funcionales de una Neurona

2.2.1. Modelos de red neuronal

En [137], [138] y [139] se describen los modelos basados en neuronas simples, denominados redes neuronales. Estos modelos se representan mediante programas de ordenador que manejan una estructura constituida por interconexiones, primitivas y elementos de procesamiento. Asociada a cada conexión hay un peso W_i y a cada elemento de procesamiento un estado (normalmente apagado o encendido). El conjunto de estos pesos y estados representan la distribución de datos de la red. Un PE (*Processing Element*) funciona como un dispositivo de umbral sencillo (ver Figura 2.3), que depende de los estados S_i (elementos de entrada) y los pesos de las conexiones W_i. La programación de una red, en primer lugar implica especificar la función de los elementos de procesamiento y su interconexión, y en segundo lugar definir el proceso de entrenamiento de la red con los patrones de entrada.

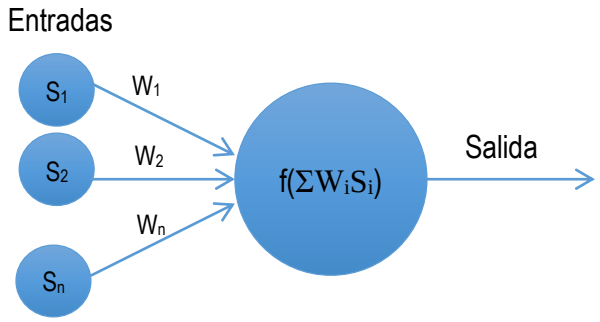

Figura 2.3 PE de una Red Neuronal

Típicamente, en las redes neurales, los PE se organizan en capas donde cada PE de una capa tiene una conexión ponderada (con un peso) en la capa siguiente. Esta organización de los PE y las conexiones ponderadas crea una red neuronal, también conocida como ANN (*Artificial Neuronal Networks*). Una ANN que aprende patrones, lo hace mediante el ajuste de los pesos de las conexiones entre los PE, análogos a los pesos sinápticos. A través de estos ajustes una red neuronal presenta propiedades de generalización y clasificación. A continuación se revisan las propiedades del procesamiento de información neuronal.

2.2.2. Computación Neuronal

En lo que respecta a este ámbito de estudio, según [140] existen tres paradigmas para el procesamiento de información: convencional, cerebral y de neurocomputación. El primero representa la información mediante instrucciones y datos, realiza el procesamiento digital de los datos y se programa mediante instrucciones y datos iniciales de entrada. El segundo representa la información mediante conexiones neuronales internas, realiza un procesamiento análogo de los datos dado que es un sistema biológico y se programa mediante el entrenamiento con patrones. El último paradigma intenta emular el paradigma cerebral representando la información mediante conexiones de red, utilizando pesos y funciones en los PE.

El procesamiento de información neuronal o neurocomputación, exhibe muchas propiedades análogas al procesamiento cerebral, tales como: la asociación, la generalización, la búsqueda paralela, el aprendizaje y la flexibilidad.

Como anteriormente se ha indicado una ANN está constituida por PE y conexiones ponderadas y su propósito es el aprendizaje de patrones, mediante el ajuste de los pesos de las conexiones entre los PE. Según [140] los modelos ANN más comunes son: *Hopfield/Kohonen*, *Perceptron*, *Delta Rule*, *Back Propagation*, *Boltzmann Machine*, *Counter-Propagation*, *Self Organizing Map* y *Neocognitron*.

2.2.3. Características de las ANN

Las redes neuronales individuales se caracterizan por una serie de propiedades clave como: la topología de la red (patrón de interconexión de las neuronas), los valores de entrada, la definición de la fase de recordación (*recall*) y el funcionamiento de la red (fase de aprendizaje y entrenamiento donde se establecen los valores de los pesos).

Para comprender la descripción de las características de las ANN se procede a discriminar los componentes que constituyen una Neurona Artificial Genérica (ver Figura 2.4). Esta neurona consta de un conjunto de entradas de estado S, y una salida de estado s. Para los modelos que actualizan sus pesos, durante el aprendizaje, la neurona también podría tener un conjunto de entradas de error E, y una salida de error e. Asimismo, la neurona genérica se compone de tres funciones: la función de activación f_1, la función de actualización de peso f_2, y la función para el cálculo del error f_3. La neurona funciona en dos fases: recuerdo (*recall*) y aprendizaje (*learning*). Durante el *recall* se aplica f_1, como entrada de estado modificada por sus pesos asociados para evaluar el nuevo estado de la neurona:

$$s = f_1(S, W) \tag{2.1}$$

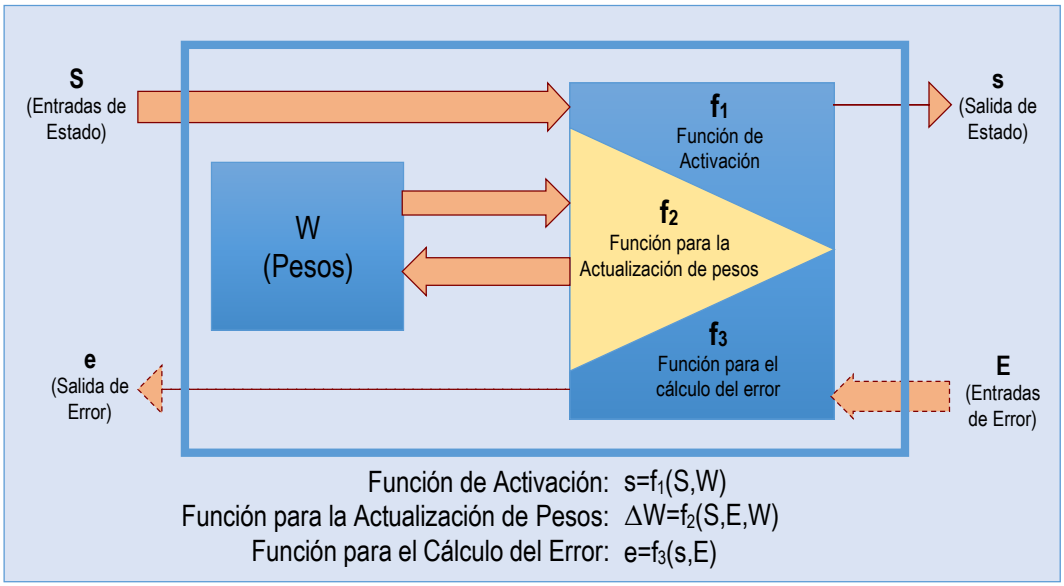

Figura 2.4. Neurona Artificial Genérica

Durante la fase de aprendizaje pueden ejecutarse las tres posibles funciones. La actualización de pesos suele ser una función de las entradas de estado S, las entradas de error E y los pesos viejos W:

$$\Delta W = f_2(S, E, W) \tag{2.2}$$

El cálculo del error es típicamente una función del estado actual, s y las entradas de error E:

$$e = f_3(s, E) \tag{2.3}$$

La topología de la red o patrón de interconexión de neuronas es quizás la característica más distintiva de los modelos de redes neuronales. Muchos de los primeros modelos, como el *Perceptron* [141] son redes de alimentación hacia adelante (*feed-forward*) de una sola capa. Esta

topología se compone de una sola capa de PE, en el que cada entrada está conectada a todos los PE pero sin realimentación de salida. Los modelos más recientes han ampliado su topología de tres maneras diferentes. En primer lugar, pueden existir las conexiones de red de PE a PE dentro de una sola capa, como en el modelo de *Hopfield* [142]. En segundo lugar, las redes pueden tener múltiples capas de alimentación hacia adelante (*feed-forward*) en la que los PE están ocultos en una capa media entre las entradas y salidas externas, como en el *Multilayer Perceptron* [139]. En tercer lugar, las redes pueden conectarse mediante alimentación hacia atrás (*feed-backward*), normalmente utilizadas para entrenar a la red, como en *Back Propagation* [139]. **Los valores de entrada de una red** son otra característica distintiva, éstos se clasifican en valores de entrada binarios y continuos.

L**a fase de recuerdo** (*recall*), se define a partir de la función de activación, que utiliza la regla de propagación **net=f(S,W)** y la función de umbral (*threshold*) **s=T(net).** La regla de propagación calcula la suma ponderada de las entradas modificadas por alguna compensación**.** En algoritmos neuronales más elaborados, la regla de propagación puede implicar operaciones matemáticas complejas, o incluso dependencias de tiempo. La función de umbral es una función no linealidad, con un elemento de procesamiento tipo limitador fuerte, sigmoide o pseudo lineal, como se muestra en la Figura 2.5.

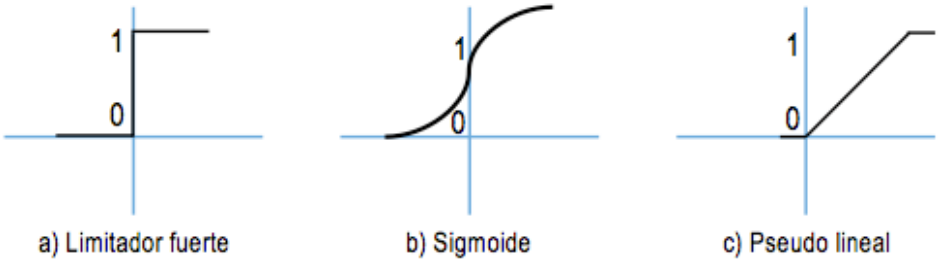

Figura 2.5 Funciones de umbral del Elemento de Procesamiento - EP

El funcionamiento de la red se refiere a la forma como la red neuronal adquiere el conocimiento durante la fase de entrenamiento/aprendizaje, es decir, cómo se modifican los pesos. Durante la fase de aprendizaje el peso requerido para cada conexión o bien se especifica directamente o los pesos se ajustan dinámicamente en respuesta a los patrones de entrada de las muestras. Durante el aprendizaje, la información se presenta repetidamente de acuerdo con un conjunto de reglas, y la tarea del sistema es extraer el patrón subyacente, y en consecuencia ajustar los pesos. El aprendizaje puede clasificarse como aprendizaje supervisado, en el que no se repite la presentación del patrón de entrada más la respuesta esperada y se corrigen los errores producidos por el sistema, y aprendizaje no supervisado, en que el sistema debe encontrar la agrupación de patrones por si mismo.

Durante la fase de aprendizaje, se utilizan funciones más complicadas que la operación de suma ponderada anteriormente discutida. La mayoría de los modelos se basan en algunas modificaciones de la regla de aprendizaje de Hebbs [139], que establece que los pesos de entrada deberían aumentar sólo si la suma ponderada está por encima de un umbral y la conexión de entrada está activa. Otras reglas de aprendizaje, detalladas en [143] son la regla Delta, el aprendizaje competitivo, la regla de energía mínima de *Hopfield*, la regla delta generalizada, las unidades *Sigma-Pi* y el algoritmo de aprendizaje de *Boltzmann*.

2.2.4. Taxonomía

Una vez introducidas las características de las ANN se procede a describir brevemente las propiedades de los modelos de ANN más conocidos.

Hopfield/Kohonen [142] es un modelo de memoria asociativa que tiene una sola capa de PE, con cada salida PE siendo alimentada por todos sus vecinos. En este modelo los pesos se establecen basándose en la regla de Hebbs. Utiliza una función de umbral de limitador fuerte y entradas binarias. El modelo de Hopfield/Kohonen asocia su potencialidad a los estados de una red neuronal y una vez se identifica un patrón de entrada a la red neural, el sistema busca un mínimo. El número de patrones que se pueden almacenar y precisa recordar debe limitarse para obtener redes estables.

Perceptron [139] es una de las primeras y más simples ANN con la capacidad de reconocer patrones simples. Se trata esencialmente de una red con interconexiones de alimentación hacia adelante (*feed-forward*), con una sola capa de PE, una función de umbral con limitador fuerte y entradas que pueden ser tanto continuas como binarias. El principal inconveniente del modelo, asociado con el procedimiento de aprendizaje, es que exige que las entradas deben ser separables para converger (es decir, deben caer en los lados opuestos de algún hiperplano).

Para superar el problema de la convergencia en el *perceptron*, se propuso el algoritmo de aprendizaje denominado *Widrow-Hoff* o Regla Delta [139] y [144]. Esta solución minimiza el error cuadrático medio entre la salida deseada de una red *perceptron* y la salida actual. La diferencia básica del *perceptron* clásico es el uso de una función de umbral lineal en lugar de la función con limitador fuerte. Aunque esta regla de aprendizaje resolvió el problema de la convergencia del Perceptrón, los modelos de una sola capa todavía no pueden realizar una serie de cálculos simples, como el *OR* exclusivo. Para resolver estas limitaciones es necesario utilizar las redes multi-capa, como el *multilayer perceptron* con una o más capas ocultas entre los PE de entrada y salida.

La red de propagación hacia atrás, *Back Propagation* [145], es un *multilayer perceptron* que emplea aprendizaje supervisado basado en la regla delta generalizada. La idea básica del modelo de propagación hacia atrás es propagar errores hacia atrás para unidades ocultas que no reciben información directa de los patrones de entrenamiento del mundo exterior. Los PE de salida calculan sus errores, e_{j0}, en la diferencia entre la salida de destino y la salida calculada, y en la derivada de la función de umbral, luego estos errores se propagan hacia atrás en los PE ocultos y se utilizan para evaluar sus errores, e_{jh}. El modelo de propagación hacia atrás utiliza la función de umbral sigmoide y acepta valores de entrada continuos.

Otro modelo importante es la máquina de *Boltzmann*, también es un *multilayer perceptron*, que utiliza la técnica de enfriamiento simulado (*simulated annealing technique*) [145] y [146] para modificar las conexiones internas. Esta técnica ayuda a la construcción de detectores de características de orden superior. La máquina de *Boltzmann* utiliza valores de entrada binarios y aplica la distribución de *Boltzmann*, con un parámetro de temperatura para determinar la función de probabilidad usada para ajustar los pesos. El algoritmo de aprendizaje consiste en ciclos de aprendizaje de dos fases. La primera es la fase supervisada donde los patrones replican al sistema y la red se estabiliza utilizando el conjunto actual de pesos. La segunda es la fase no supervisada

donde no hay salida destino y el sistema calcula de nuevo los estados hasta que se alcanza el equilibrio. Los pesos se actualizan utilizando la información estadística obtenida de las dos fases anteriores. Este proceso se continúa hasta que el cambio promedio de los pesos se convierte en cero.

Los **Mapas Auto-Organizativos** (*Self-Organising Map*) [139] tienen un vector de neuronas de entrada conectado a una red bidimensional de neuronas de salida. Los nodos de salida están ampliamente interconectados con muchas conexiones locales. El sistema aplica valores de entrada continuos y se entrena con aprendizaje no supervisado. El algoritmo de aprendizaje requiere, para cada nodo, una definición de un vecindario que disminuye lentamente de tamaño con el tiempo. Se basa en la comprobación del PE más activo y la actualización de sus pesos. Dado que nuestra propuesta se basa en los SOM, se analizarán más detenidamente en la sección 2.3.

El **Neocognitrón** [148] es uno de los modelos más complejos. Es una red de alimentación hacia adelante (*feed-forward*), multicapa, jerárquica, que consta de una cascada de capas alternas de dos tipos, capas de las células, S, que son las células características de extracción, y las capas de células, C, que ayudan a corregir los errores de posición en los patrones que hay que aprender. La operación de tolerancia de error posicional da a la red la capacidad de reconocer patrones incluso distorsionados. La red cuenta con pesos variables de células, C, a células, S, que se actualizan sólo si: (1) la célula que recibe la conexión es la célula más activa en su vecindad; y (2) la célula que envía la conexión está también activa. Cada célula, S recibe dos tipos de entradas: entradas excitadoras, S_e, y una entrada inhibitoria, S_h, a partir de una célula adyacente, llamada célula, V.

2.3 SOM

Los Mapas Auto-organizativos [178] fueron desarrollados por Teuvo Kohonen, científico del "Centro de Investigación de Tecnologías en Redes Neuronales" de la Universidad de Helsinki en Finlandia. Estos mapas hacen posible la representación de datos multidimensionales en espacios de menor dimensión, por lo general una o dos dimensiones. La reducción de la dimensionalidad de los vectores, es una técnica de compresión de datos conocida como cuantificación vectorial. Los SOM hacen posible la creación de una red que almacena la información de tal manera que todas las relaciones topológicas en el conjunto de entrenamiento se mantienen.

La característica más preponderante de los SOM es que aprenden a clasificar los datos mediante un algoritmo de aprendizaje no supervisado. En el enfoque SOM, un vector de entrada se presenta a la red se compara iterativamente con cada uno de los vectores de pesos asociados a los nodos de la estructura del mapa, de tal forma que los pesos de cada vector se recalculen en relación al vector de entrada. Esto se repite muchas veces y con varios conjuntos de pares de vectores hasta que la red converja en el resultado deseado.

2.3.1. Fundamentación conceptual

Durante la fase de entrenamiento, en cada iteración, t, se calcula la distancia entre un vector de entrada y los pesos asociados a las unidades en el mapa de salida. Por lo general, se utiliza la distancia euclídeana, mediante la siguiente ecuación:

$$U(t) = \arg\min ||x(t) - \omega_i(t)|| \tag{2.4}$$

que proporciona el índice, $U(t)$, de la unidad ganadora, también llamada Unidad de Mejor Ajuste (BMU), es decir, la unidad con la distancia euclidiana más pequeña al patrón de entrada $x(t)$. Luego, los pesos de las unidades en el vecindario de la unidad ganadora también se actualizan utilizando:

$$\omega_i(t+1) = \omega_i(t) + \alpha(t) * h_{u(t)}(t)(x(t) - \omega_i(t)) \tag{2.5}$$

Donde $\alpha(t)$ es el factor de aprendizaje, que disminuye con el tiempo, lineal o exponencialmente, y $h_{u(t)}$ es la función de vecindad topológica, por lo general una función Gaussiana:

$$h_{u(t)}(t) = e^{\left(\frac{\|rU(t) - r_i\|^2}{2\sigma t^2}\right)} \tag{2.6}$$

En la que el tamaño de la vecindad topológica, $\alpha(t)$, se reduce con el tiempo como se muestra en:

$$\sigma(t) = \sigma_0\, e^{-\frac{t}{\tau_1}} \tag{2.7}$$

En (2.33), r_i representa la posición en el espacio de salida (2D o 3D) y $\|rU(t) - r_i\|$ es la distancia euclídeana entre la unidad ganadora $U(t)$ y la unidad i-ésima en el espacio de salida (2D o 3D). Las prestaciones de una SOM se puede evaluar mediante el uso de dos medidas. La primera es el error de cuantificación "q", que es una medida de la resolución del mapa:

$$q = \frac{1}{n_{C_i}} \sum_{x_j \epsilon C_i} \|w_i - x_j\| \tag{2.8}$$

Donde, C_i, es el conjunto de vectores de entrada mapeados en la unidad i. x_j, es el vector de entrada j-ésimo perteneciente a, C_i. w_i es el peso asociado a la unidad i, y, $n_{Ci,}$ es el número de vectores en, C_i. La otra medida es el error topográfico, T, que evalúa las características de preservación topológica de la calidad del SOM, tal como se define en la ecuación siguiente:

$$T = \frac{1}{N} \sum_{i=1}^{N} u(\overrightarrow{x_i}) \tag{2.9}$$

Donde, N, es el número total de vectores de entrada y, $u(\overrightarrow{x_i})$, es 1 si la primera y segunda BMU para el vector de entrada, $\overrightarrow{x_i}$, son unidades adyacentes, y es 0 en caso contrario [178]. Los, q y t, más bajos, adaptan mejor el SOM a los patrones de entrada.

2.3.2. Algoritmo de Aprendizaje

El proceso inicia con una distribución inicial de pesos al azar en cada nodo del mapa, y luego de un número dado de iteraciones, el SOM eventualmente se asienta en un mapa de zonas estables. Cada zona es un clasificador de características y se puede pensar en el SOM como un mapa de características del espacio de entrada, de allí su nombre.

El entrenamiento [178] ocurre en varios pasos y en un número determinado de iteraciones, a continuación se describen estos pasos:

1. Cada nodo se inicializa con su respectivo vector de pesos.
2. Se elige un vector al azar entre el conjunto de datos de entrenamiento y se presenta a la red o al mapa (vector de entrada).
3. Cada nodo se evalúa para calcular cual es el peso que más se aproxima al vector de entrada. El nodo ganador se conoce comúnmente como la mejor unidad de ajuste (BMU – *Best Matching Unit*).
4. Se calcula el radio de la vecindad del BMU, que es un valor inicialmente grande, asociado al tamaño del mapa o de la red, y va disminuyendo con el paso del tiempo. Los nodos que se localizan dentro de este radio se consideran vecinos de la BMU.
5. Cada nodo vecino a la BMU ajusta sus pesos para hacerlos más próximas al vector de entrada. Cuanto más cerca esté el nodo de la BMU, más se modificará su peso.
6. Se repite el paso 2 a lo largo de un número de iteraciones determinadas, n.

2.3.3. Campos de Aplicación

Los Mapas Autorganizativos de Kohonen facilitan la comprensión de las relaciones existentes en grandes conjuntos de datos, poseen aplicaciones en una variada gama de áreas. Algunos ejemplos puntuales de aplicación de los SOM, se refieren al agrupamiento gráfico basado en SOM para la extracción de ideas principales en documentos [179], un sistema de correlación basado en la ingeniería de *kansei* y en la evaluación de características reales de automóviles, como soporte a la compra de coches, basado en SOM [180]; un sistema hibrido basado en mapas autorganizativos de *Kohonen*, para la predicción de variación de precios de las acciones de la bolsa de valores [181]; un sistema de alto rendimiento SOM sobre núcleos IP en FPGAs, para el procesamiento de miles de elementos en tiempo real, clasificando datos relacionados con la genómica o proteómica [182]; un sistema de reconocimiento de patrones de fallas en sistemas termodinámicos basados en SOM aplicados a centrales de energía térmica [183]; la extracción automática de caminos mediante el censado de imágenes de alta resolución, basado en SOM [184]; la aplicación de mapas auto-organizativos (SOM) para determinar la composición de productos químicos [185]; un sistemas de detección de intrusiones en redes informáticas [186]; técnicas para visualización de ADN usando microarrays para el análisis de datos [187] y el modelado del contorno de imágenes [188].

2.3.4. Arquitecturas basadas en SOM

Durante los últimos años, se han sugerido modificaciones para mejorar la utilidad del SOM en diversas aplicaciones [189], [190], [191], [192], [193] y [194]. Entre las mejores propuestas, nos centramos en los mapas jerárquicos por ser los que están más relacionados con las propuestas de esta tesis.

Un mapa de características jerárquico [195] trata de descubrir la estructura jerárquica de los datos mediante una arquitectura SOM modificada. En lugar de entrenar un SOM *plano*, se entrena una estructura jerárquica equilibrada de SOM. Los datos mapeados en una sola unidad se representan en un nivel adicional de detalle en el mapa de nivel inferior asignado a esta unidad. Sin embargo, este modelo sólo representa los datos de una manera jerárquica, en lugar de realmente reflejar la estructura jerárquica de los datos. Esto se debe a que la arquitectura de la red, es decir, el número de capas y los tamaños de los mapas en cada capa se fijan antes del entrenamiento de la red. Normalmente se propone un árbol equilibrado que permite representar los datos. Sin embargo, lo deseable es tener una arquitectura de red definida a partir de las peculiaridades de los datos de entrada.

En [196] y [197] se introduce un SOM con estructura de árbol y una modificación jerárquica del SOM. Sin embargo, el objetivo de este modelo, es más bien el aumento de velocidad de cálculo durante la selección del ganador mediante el uso de una organización jerárquica de las unidades. Este modelo no proporciona una descomposición jerárquica del espacio de entrada y los patrones de entrada, de nuevo, se organizan en un SOM *plano*.

El inconveniente de tener que definir el tamaño del SOM de antemano ha sido abordado a través de diferentes modelos. Entre ellas están, por ejemplo, la rejilla incremental de crecimiento [198], la rejilla de crecimiento [199], el SOM de crecimiento [200], y el SOM hipercúbico [201]. La rejilla incremental de crecimiento permite la adición de nuevas unidades en el límite del mapa. Además, las conexiones entre las unidades del mapa se pueden establecer y eliminar de acuerdo con algunos ajustes de umbral basados en la similitud de sus respectivos vectores modelo. Esto puede generar varias estructuras de mapa separadas de forma irregular. El SOM creciente, bastante similar en esencia, utiliza un factor de propagación para controlar el proceso de crecimiento del mapa. El supervisor puede decidir entrenar SOM separadas para unidades específicas con el fin de obtener una representación más detallada. Obviamente, esto da lugar a jerarquías creadas manualmente. Este enfoque es posible cualquier variante del SOM.

La rejilla de crecimiento, por otro lado, añade filas y columnas de unidades durante el proceso de entrenamiento, comenzando con un SOM de inicialmente 2x2 unidades. La decisión de dónde insertar nuevas unidades se rige por el cálculo de alguna medida para cada unidad, por ejemplo, el contador ganador. Como una extensión, el SOM hipercúbico permite un proceso de crecimiento de la SOM en más de dos dimensiones, proporcionando así una mejor representación de los datos, a cambio de una menor interpretabilidad visual.

Se puede afirmar, sin embargo, que el objetivo principal de cada una de estas variantes adaptativas del SOM se encuentra en obtener una distribución equitativa de los patrones de entrada a través del mapa mediante la adición de nuevas unidades en la vecindad de unidades que representan un número desproporcionadamente alto de datos de entrada. Por lo tanto, no reflejan el concepto de representación en un cierto nivel de detalle, que se expresa en término del error de cuantificación total, en lugar del número de datos de entrada mapeados en áreas específicas. Por otra parte, ninguno de estos modelos de adaptación tiene en cuenta la estructura jerárquica inherentemente de los datos.

2.3.5. Inconvenientes

Aunque un SOM es una herramienta muy útil para el descubrimiento de estructuras y similitudes

en datos de alta dimensión, no es capaz de expresar su inherente estructura jerárquica [202], y su rendimiento depende de que el tamaño del mapa se fije correctamente de antemano. GHSOM [202] es una estructura jerárquica y no fija, propuesta para resolver estos problemas. La estructura GHSOM consta de varias capas compuesta por SOM independientes, cuyo número y tamaño se determinan durante la fase de entrenamiento. El proceso de crecimiento adaptativo se controla por dos parámetros que determinan la profundidad de la jerarquía y la amplitud de cada mapa. Por lo tanto, estos dos parámetros son los únicos que tienen que ser configurados inicialmente.

2.4 GHSOM

Tal como se ha indicado anteriormente, pese a que los SOM han demostrado ser una herramienta adecuada para detectar estructuras de datos de alta dimensionalidad y organizar estos datos en espacios vectoriales de salida 2-D, presentan algunas deficiencias, relativas a su incapacidad para captar estructuras de datos inherentemente jerárquicas. Además, el tamaño del mapa tiene que ser prefijado de antemano, antes incluso de tener una visión adecuada de las características de la distribución de los datos.

Estos inconvenientes se han tratado por separado en varias arquitecturas SOM modificadas. Sin embargo, ninguno de los enfoques propuestos han proporcionado una arquitectura que se adapte totalmente a las características de los datos de entrada. Para superar las limitaciones tanto de arquitecturas de tamaño fijo como de arquitecturas adaptativas no jerárquicas, se desarrolló GHSOM, que ajusta dinámicamente su arquitectura multicapa de acuerdo a la estructura de los datos [203].

A partir de lo definido en [204] y [205], a continuación se detallan los principios que fundamentan a GHSOM y la descripción del algoritmo de entrenamiento que le sustenta, en particular lo referido a la configuración inicial del GHSOM y el control global de la red, su entrenamiento y crecimiento, la finalización de la fase de crecimiento, cómo es el crecimiento jerárquico del GHSOM con la orientación del mapa global, y el análisis de las características del GHSOM.

2.4.1. Principios

Un GHSOM tiene una estructura jerárquica de múltiples capas, donde cada capa se compone de varias SOM crecientes independientes. A partir de un mapa de nivel superior, cada mapa, similar al modelo de cuadrícula o rejilla de crecimiento, crece en tamaño para representar una colección de datos en un nivel específico de detalle. Después de alcanzar una cierta mejora con respecto a la granularidad de representación de los datos, se analizan las unidades para ver si representan los datos al nivel de granularidad mínimo que se ha especificado. Aquellas unidades que representan datos de entrada demasiado diversos se expanden para formar un nuevo SOM en una capa posterior, donde estarán representados los datos respectivos con más detalle. Estos nuevos mapas crecen de nuevo en tamaño hasta que se alcanza una mejora de la calidad especificada de representación de los datos. Cuando se logran identificar unidades que representen un conjunto lo suficientemente homogéneo de los datos, no se requerirá ninguna nueva expansión en las capas subsiguientes. Por lo tanto, el GHSOM resultante, tiene una arquitectura que permite reflejar la estructura jerárquica inherente a los datos, mediante la asignación de más espacio para la representación de las zonas homogéneas en el espacio de entrada.

Una representación gráfica de un GHSOM se muestra en la Figura 2.6, tomada de [159]. El mapa de la capa 1 se compone de 3x2 unidades y proporciona una organización muy refinada de los principales grupos de datos de entrada. Los seis mapas independientes en la segunda capa ofrecen una vista más detallada de los datos. Los datos de entrada para un mapa son el subconjunto que ha sido mapeado a la unidad correspondiente en la capa superior. Adicionalmente se han expandido dos unidades desde la segunda capa del mapa, hacia la tercera capa, para proporcionar una representación lo suficientemente granular de los datos de entrada. Los mapas tienen diferentes tamaños de acuerdo con la estructura de los datos, lo que libera de la carga de predefinir la estructura de la arquitectura. La capa 0 sirve como una representación del conjunto de datos completo y es necesaria para el control del proceso de crecimiento.

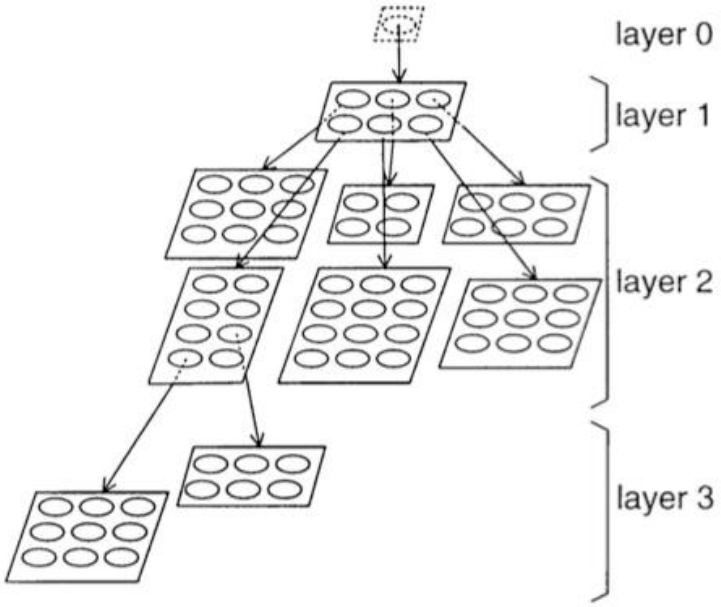

Figura 2.6. Representación gráfica de un GHSOM

2.4.2. Algoritmo de Entrenamiento

En esta sección se describe el algoritmo de entrenamiento a través de sus diversas fases y aspectos a considerar, que han sido defnidos en [159].

Configuración inicial del GHSOM y control global de la Red. El principio de la arquitectura GHSOM es su adaptación a los datos de entrenamiento. La calidad de esta adaptación se mide en términos de la desviación entre el vector modelo de la unidad y los vectores de entrada representados por esta unidad en particular. Se utilizan básicamente dos estrategias diferentes para el control del proceso de crecimiento, el error de cuantificación medio, **mqe**, de una unidad (que se emplea comúnmente como una medida de calidad para la representación de datos con SOM), y el valor absoluto, es decir, el error de cuantificación, **qe**, de una unidad.

Más formalmente, el **mqe**, de una unidad, **i**, se calcula mediante (2.10), como la distancia euclídea media entre su vector modelo, $\mathbf{m}_i$, y los, $\mathbf{n}_c$, vectores de entrada, $\mathbf{x}_j$, que son elementos del conjunto de vectores de entrada, $\mathbf{C}_i$, mapeado a esta unidad, **i.**

$$mqe_i = \frac{1}{n_C} \cdot \sum_{x_j \epsilon C_i} \|m_i - x_j\|, \quad n_C = |C_i|, \qquad C_i \neq \emptyset \tag{2.10}$$

El punto de partida para el proceso de entrenamiento del GHSOM es el cálculo de un, **mqe_0,** de la unidad formada en la capa 0 del mapa tal como se indica en (2.11). Siendo, **n_I,** el número de todos los vectores de entrada**,** **x_i,** de los datos de entrada**,** **I,** y **m_0,** denota la media de los datos de entrada.

$$mqe_0 = \frac{1}{n_I} \cdot \sum_{x_i \epsilon I} \|m_0 - x_i\|, \quad n_I = |I| \tag{2.11}$$

El, **mqe**, mide la disimilitud de todos los datos de entrada mapeados en una unidad particular y se utiliza para controlar el proceso de crecimiento de la red neuronal. Específicamente, la calidad mínima de la representación de los datos de cada unidad será especificada como una fracción, indicada por un parámetro, **τ_2**, de, **mqe_0**.

En otras palabras, todas las unidades deben representar a sus respectivos subconjuntos de datos mediante un, **mqe,** más pequeño que una fracción, **τ_2**, de, **mqe_0**. Es decir, deben satisfacer el **criterio de parada global** que se especifica como:

$$mqe_i < \tau_2 \cdot mqe_0 \tag{2.12}$$

Para todas las unidades que no respondan a esta condición, se requiere una representación más detallada de datos, lo que lleva a la adición de otras unidades para proporcionar más espacio en el mapa de representación de datos.

Alternativamente, en lugar de la, **mqe**, se puede utilizar el error de cuantificación de la unidad, denotado por, **qe**, como se indica en (2.14). Así se tiene el criterio de parada global indicado en (2.13):

$$qe_i < \tau_2 \cdot qe_0 \tag{2.13}$$

$$\text{Siendo} \quad qe_i = \sum_{x_j \in C_i} \|m_i - x_j\| \tag{2.14}$$

El uso de la, **mqe**, de la distribución de datos como una medida de calidad global para el proceso de entrenamiento del GHSOM, puede ser más intuitivo si se utiliza el, **qe**, de acuerdo con el principio característico de SOM de proporcionar más espacio de mapa para las regiones más densamente pobladas del espacio de entrada. Es el denominado **factor de ampliación**. Por lo tanto, el uso de (2.13) en lugar de (2.12) como un criterio de parada global, suele producir mapas que reflejan de forma más intuitiva las características de las distribuciones de datos mediante la captura de las diferencias más sutiles en los cúmulos más densamente poblados. Este efecto es especialmente importante cuando se utilizan los mapas resultantes como interfaces exploratorias para los *dataset*, ya que es con frecuencia el caso de arquitecturas SOM similares. Por lo tanto,

en la propuesta de [204] se utiliza, **qe**, como base para el entrenamiento del GHSOM.

Tras la decisión de usar, **mqe**, o, **qe**, para el control global del proceso de entrenamiento, la arquitectura GHSOM se inicializa mediante la creación de un nuevo SOM debajo de la capa 0 del mapa. El tamaño inicial del mapa, de la primera capa, es de 2x2 unidades, con sus vectores inicializados a valores aleatorios.

Proceso de entrenamiento y crecimiento de un SOM jerárquico. El entrenamiento de un mapa nuevo se realiza de acuerdo con el procedimiento utilizado para un SOM estándar, como se ha indicado anteriormente. Después de un número fijo de iteraciones de entrenamiento, **λ**, se analizan los, **qe**, de todas las unidades según (2.13). Un, **qe**, alto indica que una parte no homogénea del espacio de entrada que contiene datos diferentes, o al menos un conjunto bastante grande de datos de entrada de una porción más homogénea del espacio de entrada, está representada por esta unidad. Por lo tanto, se necesitan nuevas unidades para proporcionar más espacio para una representación de los datos apropiada. Así, se selecciona la unidad con el más alto, **qe**, denotada como la unidad de error, **e**. A continuación, se selecciona la unidad vecina más diferente, **d**, en términos de la distancia en el espacio de entrada. Para ello, se comparan los *vectores modelo* de todas las unidades vecinas con el *vector modelo* de la unidad de error, **e**. Posteriormente se inserta una nueva fila o columna de unidades entre, **e**, y su vecino más diferente, **d**, teniendo en cuenta que los vectores modelo de las nuevas unidades se inicializan como el promedio de sus correspondientes vecinos. La Figura 2.7 ilustra el proceso de inserción de unidades en una SOM creciente, donde se representan las unidades insertadas como círculos grises. Las flechas apuntan a las unidades vecinas respectivas, utilizadas para la inicialización de los vectores modelo. En la Figura 2.7 (a) se observa la inserción de una fila y en la 2.7 (b) de una columna de unidades (círculos sombreados de gris). Se han insertado entre la unidad de error, **e**, y la unidad vecina, **d**, con la mayor distancia, en el espacio euclidiano, entre sus respectivos vectores modelo.

Más formalmente, en el proceso de crecimiento de una SOM jerárquica, dado, $\mathbf{C_i}$, como el subconjunto de vectores, $\mathbf{x_j}$, de los datos de entrada que se mapean a la unidad, **i**, y, $\mathbf{m_i}$, el vector modelo de la unidad, **i**, entonces, la unidad de error, **e**, se determina como la unidad con el**, mqe**, siguiente.

$$mqe = \arg max_i \left(\sum_{x_j \in C_i} \|m_i - x_j\| \right),\ n_C = |C_i|,\ \ C_i \neq \emptyset \qquad (2.15)$$

Teniendo en cuenta que, **mqe**, se puede utilizar en lugar de, **qe**, se tiene una arquitectura GHSOM centrada en la homogeneidad general de la representación de los datos, en lugar de buscar un mayor grado de detalle para las áreas más densamente pobladas del espacio de datos.

Tras seleccionar la unidad, **e**, en la Figura 2.7, se procede a determinar su vecino más diferente, denominado, **d**, en la Figura 2.7, tal como se describe en (2.43), donde, $\mathbf{N_e}$, es el conjunto de unidades vecinas de, **e.**

$$d = \arg max_i \left(\|m_e - m_i\| \right),\ m_i \in \mathcal{N}_e \qquad (2.16)$$

Una vez identificada la unidad**, d**, en la Figura 2.7, se inserta una fila o columna de unidades entre las unidades, **d,** y, **e**. Teniendo en cuenta que para obtener un buen posicionamiento de las unidades recién añadidas en el espacio de entrada, los vectores modelo de éstas se inicializan

como la media de sus vecinos respectivos. Tras la inserción se restablecen los valores originales de la tasa de aprendizaje y del rango del vecindario, y se continúa el proceso de entrenamiento tal como se hace en el caso de un SOM, para las iteraciones siguientes.

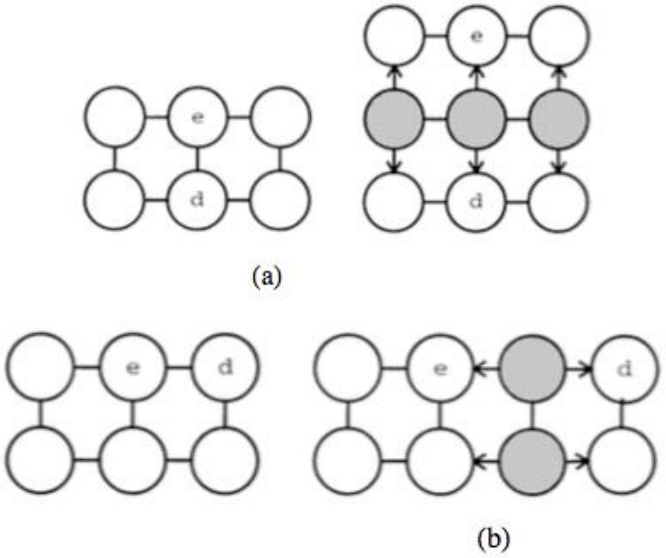

Figura 2.7. Inserción de unidades

Este proceso de entrenamiento de un SOM creciente es muy similar al modelo de rejilla de crecimiento propuesto en [206]. La diferencia, hasta el momento, es que en GHSOM se utilizan una tasa de aprendizaje y rango de vecindario decrecientes, en lugar de valores fijos.

Finalización del proceso de crecimiento

A medida que se añaden más unidades a la SOM creciente los valores de los, **qe**, correspondientes disminuyen, dado que las unidades cada vez representan un subconjunto más pequeño y conciso del espacio de entrada. Básicamente, el proceso de entrenamiento podría continuar hasta que todas las unidades cumplan el criterio de parada global. Es decir, hasta que representen a sus respectivos subconjuntos de datos con una granularidad menor que una cierta fracción de la desviación estándar inicial de los datos, dando lugar a un SOM que represente todos los datos con la granularidad requerida en una capa.

Sin embargo, a fin de revelar la estructura jerárquica presente en los datos, cada mapa se "explicará" solamente como una porción de similitud de datos. Por lo tanto, el proceso de crecimiento continúa sólo hasta que el, **mqe**, del mapa, designado ahora como, **MQE** (en letras mayúsculas), alcance una cierta fracción, τ_1, de la, **qe**, de la unidad correspondiente, **u,** en la capa superior, $\mathbf{qe_u}$ (es decir, la unidad que constituye la capa 0 del mapa, para el mapa de la primera capa). El, **MQE**, de un mapa se calcula como la media de todos los errores de cuantificación de las unidades, $\mathbf{qe_i}$**,** tal como se indica en (2.13), del subconjunto, **u**, de las unidades de los mapas en la que los datos *se mapean*.

$$MQE_m = \frac{1}{n_u} \cdot \sum_{i \in u} qe_i \ , \quad n_u = |u| \tag{2.17}$$

En términos generales, el criterio de parada para el crecimiento de un solo mapa, **m**, se define como:

$$MQE_m < \tau_1 \cdot qe_u \quad (2.18)$$

Donde, **qe_u,** es el**, qe**, de la unidad correspondiente, **u,** en la capa superior. Obviamente, cuanto menor es el parámetro**, τ_1**, mayor será el mapa resultante elegido, "explicando" sus datos una granularidad más alta. Para un mayor**, τ_1**, la representación de los datos es más detallada y de reproducirá en mapas adicionales más abajo en la jerarquía. Por lo tanto, el parámetro**, τ_1,** sirve para controlar la profundidad de la arquitectura jerárquica GHSOM resultante.

Como se deduce en (2.18) en el caso de la primera capa del mapa, el criterio de parada para el proceso de entrenamiento es **$MQE_1 < \tau_1 . qe_0$**.

Crecimiento jerárquico del GHSOM con la orientación del mapa global

Cuando el entrenamiento de un mapa finaliza de acuerdo al criterio especificado en (2.18), cada unidad tiene que ser revisada para verificar el cumplimiento del criterio de parada global dado en (2.13). Las unidades que representan un conjunto de vectores de entrada demasiado diverso, se expanden para formar un nuevo mapa en una capa subsecuente de la jerarquía. Las unidades que satisfacen el criterio de parada global no requieren una mayor expansión.

El procedimiento es similar al elegido para la creación del mapa de la capa 1, que se origina desde una sola unidad del mapa de la capa 0, para crear un nuevo mapa con 2x2 unidades inicialmente. De nuevo, la orientación del vector modelo se inicializa aleatoriamente, esto generalmente distorsionará la topología global de los mapas de vecinos, debido a la orientación de los datos en este mapa. Para proporcionar una orientación global de los mapas individuales en las distintas capas de la jerarquía, su orientación debe tener en cuenta la orientación de la distribución de los datos en el mapa de sus padres. Esto puede lograrse mediante la creación de una inicialización coherente de las unidades de un mapa creado recientemente según se plantea en [207].

Si se supone**, p,** que se expande para formar un nuevo mapa de 2x2 en la capa subsecuente de la jerarquía, para este mapa se inicializan cuatro vectores modelo de, **s1**, a, **s4**, para reflejar la orientación de las unidades vecinas de sus padres, **p**. Geométricamente hablando, los vectores modelo de las cuatro unidades de la esquina se mueven en el espacio de datos hacia las direcciones de los vecinos de sus respectivos padres. La orientación inicial del mapa se conserva durante el proceso de entrenamiento. Mientras que las nuevas unidades se pueden insertar durante el entrenamiento, las cuatro unidades de las esquinas se mantienen similares a las respectivas unidades de las esquinas de los mapas en las ramas vecinas. La cantidad exacta por la cual estas unidades de la esquina se mueven en las direcciones respectivas no influyen en la característica de la inicialización de la preservación de la topología. Por lo tanto, se puede optar por configurarlas a la media del padre, y sus vecinos en las direcciones respectivas, por ejemplo, la asignación de **e1=(e+a+b+d)/4**. En el caso más simple, los vectores modelo de los vecinos se pueden utilizar directamente como posiciones de esquina iniciales de los nuevos mapas en el espacio de datos.

Los vectores de entrada para entrenar el mapa recién agregado son los mapeados sobre cada

unidad que acaba de ser expandida. Es decir, el subconjunto del espacio de datos asignado a su padre. Este mapa, de nuevo, seguirá creciendo mediante los procedimientos anteriormente descritos y todo el proceso se repite para las capas subsiguientes hasta que el criterio de parada global dado en (2.14) se cumple para todas las unidades hojas.

Análisis de las características del GHSOM

Un SOM jerárquico facilita el análisis de grandes colecciones de datos de dimensiones elevadas, permitiendo el uso de ciertas funcionalidades para acelerar el proceso de entrenamiento. En [208] se detalla este proceso. Además, debido a su estructura jerárquica y la partición de los datos de entrada, el GHSOM proporciona una mejor escalabilidad. En la transición de una capa a la siguiente, el número de vectores de entrada utilizado para el entrenamiento de un mapa particular disminuye al subconjunto de vectores mapeados sobre la unidad de capa superior respectiva.

Además, cada mapa en la jerarquía representa un conjunto particular de características de sus datos de entrada. Posteriormente, se puede esperar que algunos componentes del vector de entrada (es decir, las características del conjunto de datos) sean casi idénticos para todos los vectores de entrada mapeados en una unidad específica. Estas características pueden ser ignoradas en la transición a una capa subsecuente de la jerarquía, lo que permite acortar los vectores de entrada. Especialmente para los conjuntos de datos dispersos y de muy alta dimensionalidad, este efecto permite una reducción significativa del esfuerzo computacional.

Los procesos de entrenamiento y crecimiento de un GHSOM están totalmente basados en los datos, y no requieren conocimientos previos o estimaciones para la especificación de parámetros. La estructura jerárquica de los datos permite representar: (1) jerarquías inferiores con refinamientos más detallados presentados en cada capa posterior o (2) jerarquías más profundas con una separación estricta de los distintos subgrupos, mediante la asignación de mapas separados. El parámetro, τ_1, se utiliza para controlar este equilibrio entre jerarquías poco profundas o muy profundas.

En el primer caso se prefieren mapas más grandes en cada capa, para “explicar” porciones más grandes de los datos en su representación plana. Sin embargo, proporcionan una estructura menos jerárquica. Como un ejemplo extremo, se podría considerar una red cada vez mayor, que crece en tamaño para “explicar” la estructura completa de los datos en un solo mapa plano. Por otro lado, se podría considerar la asignación de un, $\boldsymbol{\tau 1}$, bastante grande, que implica el crecimiento limitado de los mapas individuales, y origina una estructura jerárquica más profunda de mapas pequeños. Básicamente, se puede esperar que el número total de unidades en los mapas de nivel más bajo sea similar en ambos casos, ya que éste es el número de unidades de procesamiento neuronales necesarios para la representación de los datos en el nivel de granularidad requerido.

En principio, la elección de, $\boldsymbol{\tau 1}$, puede parecer crucial, ya que podría dar lugar a una separación más bien arbitraria de un grupo más grande, con la distribución de datos homogénea en dos o más subclusters en diferentes ramas. Sin embargo, debido a la orientación global proporcionada por la inicialización de los vectores modelo de nuevos mapas, se facilita la navegación a través de mapas en la misma capa de la jerarquía, compensando el daño de la separación de grupos. Además, el análisis de las distancias en el espacio de entrada de vectores modelo en los vecinos límites del mapa, permite detectar la similitud de los mapas de vecinos.

Para el análisis exploratorio de datos, es preferible una distribución homogénea de muestras de datos a través del espacio del mapa, lo que permite capturar las diferencias más sutiles entre los clusters en las zonas más densamente pobladas del espacio de datos. Por lo tanto, utilizar el**, qe**, de una unidad, en lugar de su**, mqe**, ha demostrado que produce resultados más favorables. El criterio de parada global, ya sea un valor absoluto o una fracción especificada por el parámetro, **τ2**, influye directamente en el tamaño total de la GHSOM resultante, es decir, en el número de unidades disponibles para la representación del espacio de datos.

Además, debe tenerse en cuenta que el proceso de entrenamiento generalmente no conduce a una jerarquía equilibrada en términos de todas las ramas que tienen la misma profundidad. Esta es una de las principales ventajas del GHSOM sobre un mapa jerárquico característico [209], debido a que la estructura de la jerarquía se adapta a si misma de acuerdo con los requisitos del espacio de entrada. Por lo tanto, las áreas en el espacio de entrada que requieren más unidades para la representación apropiada de los datos, crean ramas más profundas que otras.

2.5 OPTIMIZACIÓN

Los problemas de optimización plantean objetivos que necesitan ser minimizados (costos, pérdida de energía, errores, etc) o maximizados (beneficios, calidad, eficiencia, etc), sujetos a ciertas restricciones. El problema es considerado de único objetivo si sólo hay que optimizar una función objetivo. Por el contrario, los problemas de optimización multiobjetivo requieren la optimización simultánea de varias funciones objetivo. Cuando un problema de optimización tiene un único objetivo, la definición de "mejor solución" es unidimensional, y es a menudo una única mejor solución. Por el contrario, un problema multi-objetivo suele presentar objetivos entre los que existen conflictos y no tiene una única solución óptima sino un conjunto de soluciones óptimas. Hay muchos métodos tradicionalmente disponibles para resolver problemas de optimización. En [225] se presentó una clasificación entre los modelos de optimización y los algoritmos de optimización.

2.5.1. Modelos de Optimización

Los modelos de optimización definen el problema de optimización, utilizando diferentes técnicas de modelado. Una clasificación típica de nivel superior son los programas matemáticos frente a los modelos de simulación. Un programa matemático formula el problema en una representación matemática, si tanto la función objetivo como las restricciones son funciones lineales de “x”, el problema se designa en la categoría de Programación Lineal (LP), por el contrario, tener funciones objetivo no lineales y/o restricciones, da como resultado un Programa No Lineal (NLP). Adicionalmente, si todas las variables del problema son de tipo entero el problema se clasifica como un Programa Entero (IP), pero si sólo un subconjunto de las variables necesita ser entero, el programa se conoce como un Programa entero mixto (MIP). Actualmente existen métodos computacionales muy potentes para la solución problemas LP [226]. La diferencia entre resolver un NLP y un LP es análoga a la diferencia entre resolver un conjunto de ecuaciones lineales y resolver un conjunto de ecuaciones no lineales [226]. Los LP normalmente se pueden resolver para el óptimo eficiente, los NLP normalmente no pueden.

2.5.2. Algoritmos de optimización

Para resolver los problemas de optimización que se formulan usando “modelos de optimización”,

se utilizan “algoritmos de optimización”. Éstos se clasifican en exactos e inexactos. Los algoritmos exactos localizan un óptimo global, pero por lo general no se escalan de manera eficiente a problemas grandes. Los algoritmos inexactos bien pueden garantizar o no el alcance de un cierto nivel de calidad en la solución. El primero es el llamado algoritmo de aproximación y el último es denominado heurístico.

Los algoritmos heurísticos típicamente escalan mucho más eficientemente problemas grandes. Estos algoritmos proporcionan buenas soluciones, aunque no garantizan su óptimalidad, o incluso qué tan cerca las soluciones están del óptimo global [227]. Aun así, se han desarrollado algunos algoritmos heurísticos modificados que, de acuerdo a sus autores, garantizan soluciones óptimas [228].

La heurística puede clasificarse en heurística simple y metaheurísticas. Las heurísticas simples son típicamente del tipo de búsquedas locales, algoritmos codiciosos (greedy), y/o escaladores de montaña (hill climbers). Las heurísticas simples son muy rápidas y ágilmente capaces de mejorar una determinada solución, pero son muy propensas a quedar atrapadas en óptimos locales. Las metaheurísticas tratan de buscar más allá de la mejor solución en un vecindario predefinido. Ejemplos típicos son los algoritmos basados en conjuntos y los algoritmos evolutivos. Muchos métodos metaheurísticos se basan en alguna metáfora biológica, básicamente todos ellos parten de un punto o un conjunto de puntos, avanzando hacia una mejor solución, guiados por normas lógicas o empíricas y/o sensibilidades (reglas heurísticas). Estas reglas se utilizan para generar y clasificar las opciones durante la búsqueda. El proceso heurístico se lleva a cabo hasta que el algoritmo no es capaz de encontrar una mejor solución. Los principales métodos heurísticos se listan a continuación.

- Algoritmos evolutivos: basados en la genética y la evolución [229].
- Simulación Annealing: basado en algunos principios de la termodinámica [230].
- Enjambre de partículas (Particle Swarm): basado en el movimiento de las aves y peces [231].
- Búsqueda Tabú: basado en la respuesta de la memoria [232].
- Colonia de Hormigas: en función de cómo se comportan las hormigas [233].
- Algoritmos híbridos: combinación de metaheurísticas.

Es difícil concluir qué algoritmo (en combinación con un modelo de optimización) presenta el mejor enfoque, ya que estos algoritmos no son determinísticos, y en muchos casos, la eficacia depende de varios temas (por ejemplo, la sintonización de parámetros de control, la formulación del problema, y, sobre todo, la forma en que el problema se puede adaptar al enfoque seleccionado) [234]. Sin embargo, muchos modelos propuestos en la literatura confirman la superioridad del enfoque heurístico en relación al manejo de problemas: (1) complejos y con gran tamaño, (2) no lineales y combinatorios, y (3) cuyas alternativas y métodos exactos aumentan de manera significativa al esfuerzo computacional requerido para el dimensionamiento de la red. Los algoritmos exactos sufren de la "maldición de la dimensionalidad” (es decir, la complejidad en el tiempo para grandes problemas). En conclusión, los métodos de optimización exactos logran el óptimo matemático, pero tienen que sacrificar el modelado de precisión en algunos casos.

La presencia de múltiples objetivos en un problema, en principio, da lugar a un conjunto de soluciones óptimas (conocidas como soluciones óptimas de Pareto), en lugar de una única solución óptima. A falta de más información, una de esas soluciones óptimas de Pareto no se puede decir que sea mejor que otra. Esto requiere que un usuario encuentre tantas soluciones

óptimas de Pareto como sea posible. Métodos de optimización clásicos (incluyendo métodos de toma de decisiones multicriterio) sugieren convertir el problema de optimización multiobjetivo a un problema de optimización de un solo objetivo, enfatizando en una solución óptima de Pareto particular, a la vez. Cuando un método de este tipo se va a utilizar para encontrar múltiples soluciones, tiene que ser aplicado muchas veces, y es posible que se encuentre una solución diferente en cada ejecución de la simulación.

2.6 ALGORITMOS DE OPTIMIZACIÓN MULTIOBJETIVO BASADOS EN COMPUTACIÓN EVOLUTIVA

Haciendo un barrido del estado del arte, en materia Algoritmos Evolutivos Multiobjetivo (MOEA), se evidencian los siguientes [235], [236], [237], [238] y [239]. La característica más preponderante que les define es su capacidad de encontrar múltiples soluciones óptimas de Pareto en una única ejecución de la simulación. Dado que los algoritmos evolutivos (EA) que trabajan con una población de soluciones, un simple EA puede extender y mantener un conjunto diverso de soluciones. Un EA se puede utilizar para encontrar múltiples soluciones óptimas de Pareto en una única ejecución.

Durante 1993-1995, surgieron una serie de diferentes EA para resolver problemas de optimización multiobjetivo. El MOGA de *Fonseca* y *Fleming* [236], el NSGA de *Srinivas* y *Deb* [238] y el NPGA de *Horn* y otros [243]. Estos algoritmos requirieron de operadores adicionales para convertir un sencillo EA a un MOEA. Dos características comunes de los tres EAs anteriormente mencionados son que la asignación del fitness para miembros de la población se basó en la clasificación de no dominados y que se preserva la diversidad entre las soluciones del mismo frente no dominado.

El NSGA fue uno de los primeros EA multiobjetivo. Sus principales defectos son su complejidad computacional debido a su método de clasificación de no dominadas, su falta de elitismo, y su necesidad de especificar un parámetro de intercambio.

Aunque se han encontrado múltiples soluciones no dominadas en muchos problemas de prueba y un número de problemas de diseño de ingeniería, los investigadores dieron cuenta de la necesidad de introducir operadores más útiles (utilizando EA de un solo objetivo) con el fin de resolver mejor, problemas de optimización multiobjetivo. En particular, el interés ha sido la introducción de elitismo para mejorar las propiedades de convergencia de un MOEA. En [240] se muestra que el elitismo ayuda a lograr una mejor convergencia en MOEA. Los MOEA elitistas existentes son el SPEA de *Zitzler* y *Thiele* [239], y el PAES de *Knowles* y *Corne* [244].

2.7 NSGA-II

El algoritmo NSGA, ha sido criticado principalmente por su alta complejidad computacional $O(MN^3)$, donde M es el número de objetivos, y N el tamaño de la población, por su enfoque no elitista, y porque requieren de la especificación previa de un parámetro distribuir uniformemente las soluciones.

En [245] se propone NSGA-II, que soluciona las dificultades anteriormente mencionadas y tiene una complejidad computacional de $O(MN^2)$. Además, presenta un operador de selección que crea un grupo de coincidencias (*mating pool*) mediante la combinación de los padres y las poblaciones descendientes y la selección de las, N, mejores soluciones (con respecto al fitness y

la propagación). Los resultados de NSGA-II mostrados en [245], indican que en la mayoría de los problemas considerados el algoritmo es capaz de encontrar una mucho mejor difusión de soluciones y una mejor convergencia cercana al *frente de pareto óptimo verdadero* en comparación con los otros dos MOEA elitistas: PAES [244] y SPEA [246], que prestan especial atención a la creación de un *frente óptimo de pareto diverso*.

2.7.1. El enfoque de NSGA-II

En [245] se describe inicialmente un procedimiento sencillo pero lento para clasificar una población en diferentes niveles de no dominancia. A partir de éste, se describe un enfoque rápido.

En el enfoque sencillo, con el fin de identificar soluciones del primer frente de soluciones de no dominancia en una población de tamaño, N, cada solución se puede comparar con cualquier otra solución en la población a buscar, si está dominada. Esto requiere $O(MN)$ comparaciones para cada solución, donde M, es el número de objetivos. Cuando se sigue este proceso de encontrar todos los miembros del primer nivel de no dominancia en la población, la complejidad total es $O(MN^2)$. Con el fin de buscar los individuos en el próximo frente de no dominancia, las soluciones del primer frente se eliminan temporalmente y el procedimiento anterior se repite. En el peor de los casos, la tarea de encontrar el segundo frente también requiere $O(MN^2)$ cálculos, particularmente cuando un número $O(N)$ de soluciones pertenecen al segundo y niveles más altos de no dominancia.

Este argumento es válido para la búsqueda de los niveles tercero y superiores de no dominancia. Por lo tanto, el peor de los casos es cuando hay, N, frentes y existe sólo una solución en cada frente. En general esto requiere $O(MN^3)$ cálculos, teniendo en cuenta que se requiere un almacenamiento de $O(N)$ para este procedimiento. En los párrafos siguientes y en la Tabla 2.1, se describe un enfoque de clasificación de no dominancia rápida que necesita del orden de $O(MN^2)$ cálculos.

En primer lugar, para cada solución se calculan dos entidades: el contador de dominancia, n_p (el número de soluciones que dominan la solución), y el conjunto de soluciones que la solución domina S_p. Esto requiere $O(MN^2)$ comparaciones.

Todas las soluciones en el primer frente de no dominancia tendrán su contador de dominancia a cero. Ahora, para cada solución p, con $n_p = 0$, se visita cada miembro q, de su conjunto S_p, y se reduce su contador de dominación en uno. De este modo, si para cualquier miembro q, el contador de dominancia llega a cero, se pone en una lista separada Q. Estos miembros pertenecen al segundo frente de no dominancia. Ahora, el procedimiento anterior continúa con cada miembro de Q y se identifica el tercer frente. Este proceso continúa hasta que se identifican todos los frentes.

Para cada solución p en el segundo o mayor nivel de no dominancia, el contador de dominancia n_p, puede ser como máximo $N - 1$. Por lo tanto, cada solución se podrá visitar a lo sumo $N - 1$ veces antes de que su contador de dominancia se convierte en cero. En este punto, se le asigna a la solución un nivel de no dominancia y nunca se visitará de nuevo. Puesto que hay por lo menos $N - 1$ de este tipo de soluciones, la complejidad total es $O(N^2)$. Por lo tanto, la complejidad global del procedimiento es $O(MN^2)$. Otra forma de calcular esta complejidad es

que se cumpla que el cuerpo del primer bucle interior ($for\ each\ p \in \mathcal{F}_i$) se ejecute exactamente, N, veces, ya que cada individuo puede ser el miembro de a lo sumo, un frente y el segundo bucle interior ($for\ each\ q \in S_p$) se puede ejecutar como máximo $(N-1)$ veces para cada individuo, resultando en total $O(MN^2)$ cálculos, teniendo en cuenta que cada individuo domina $(N-1)$ individuos como máximo y cada verificación de dominancia requiere al menos M comparaciones. Es importante tener en cuenta que, aunque la complejidad de tiempo se ha reducido a $O(MN^2)$ el requisito de almacenamiento ha aumentado a $O(N^2)$.

Tabla 2.1 Enfoque de clasificación de no dominancia rápida

Clasificación_No_Dominada_Rápida(P)	
$\boldsymbol{for\ each\ p \in P}$	
$S_p = \emptyset$	
$n_p = 0$	
$for\ each\ q \in P$	
$if(p \prec q)\ then$	Si, p, domina, q
$S_p = S_p \cup \{q\}$	Añade, q, al conjunto de soluciones dominadas por, p
$else\ \ if(q \prec p)\ then$	
$n_p = n_p + 1$	Incremente el contador de dominancia de, p
$if(n_p = 0)\ then$	"p" pertenece al primer frente
$p_{rank} = 1$	
$\mathcal{F}_1 = \mathcal{F}_1 \cup \{p\}$	
$\boldsymbol{i = 1}$	Inicialice el contador de frente
$\boldsymbol{while\ \mathcal{F}_i \neq \emptyset}$	
$Q = \emptyset$	Usado para almacenar los miembros del próximo frente
$for\ each\ p \in \mathcal{F}_i$	
$for\ each\ q \in S_p$	
$n_q = n_q - 1$	
$if(n_q = 0)\ then$	q, pertenece al próximo frente
$q_{rank} = i + 1$	
$Q = Q \cup \{q\}$	
$i = i + 1$	
$\mathcal{F}_i = Q$	

2.7.2. Preservación de la diversidad

Junto con la convergencia del conjunto *óptimo de pareto*, es deseable que el EA mantenga una buena distribución de soluciones en el conjunto de soluciones obtenidas. El NSGA original utilizó una función de para obtener la distribución de las soluciones (*sharing function*), con que busca mantener la diversidad sostenible en una población con el ajuste adecuado de sus parámetros asociados. El método implica un parámetro σ_{share}. Este parámetro está relacionado con la métrica de distancia elegida para calcular la medida de proximidad entre dos miembros de la población. El parámetro σ_{share} indica el valor más grande de esa métrica de distancia en el que cualquiera de las dos soluciones comparte el fitness de cada una. Este parámetro normalmente se establece por el usuario, si bien existen algunas directrices [247]. Hay dos problemas con este enfoque. El primer problema se refiere a que el desempeño del método en el mantenimiento de la extensión de las soluciones, depende en gran medida del valor σ_{share} elegido. El segundo problema se debe a que, dado que cada solución debe ser comparada con todas las demás soluciones en la población, la complejidad general del enfoque de función de intercambio es $O(N^2)$.

En NSGA-II [245], se proporciona un enfoque de comparación de hacinamiento *crowding* que

elimina las dificultades anteriores para mantener la diversidad en cierta medida. El nuevo enfoque no requiere ningún parámetro definido por el usuario para el mantenimiento de la diversidad entre los miembros de la población. Además, el enfoque sugerido tiene una mejor complejidad computacional. Para describirlo, primero se define una métrica de estimación de la densidad y luego se utiliza el operador de comparación de hacinamiento, que se detalla a continuación.

Estimación de la Densidad: Para obtener una estimación de la densidad de las soluciones que rodean una solución particular en la población, se calcula la distancia media a dos puntos a cada lado de un punto, a lo largo de cada uno de los objetivos. Esta cantidad $i_{distance}$ sirve como una estimación del perímetro del cuboide (paralelepípedo) formado por el uso de los vecinos más cercanos como vértices (llaman a esto la distancia de hacinamiento). En la Figura 2.8, la distancia de hacinamiento *crowding* de la i-ésima solución en su frente (marcada con círculos sólidos) es la media de la longitud de los lados del cuboide (se muestra con un cuadro de líneas discontinuas). Aunque la Figura 2.8 ilustra el cálculo de la distancia de hacinamiento para dos objetivos, el procedimiento también es aplicable a más de dos objetivos.

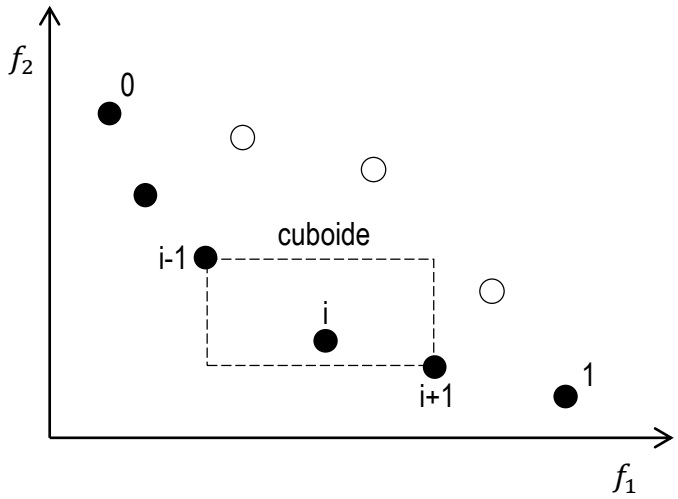

Figura 2.8. Cálculo de la distancia de hacinamiento

El cálculo de la distancia de hacinamiento, o distancia *crowding*, requiere la clasificación de la población de acuerdo al valor de cada función objetivo, en el orden ascendente de la magnitud. A partir de ello, a las soluciones de contorno de cada función objetivo (soluciones con valores de la función menor y mayor) se les asigna un valor de distancia infinita. A todos las demás valores de la función, para las soluciones intermedias, se les asignan un valor de distancia igual a la diferencia normalizada absoluta de dos soluciones adyacentes. Se continúa con este cálculo para las otras funciones objetivo. El valor de la distancia de hacinamiento total se calcula como la suma de los valores individuales de las distancias correspondientes a cada objetivo. Cada función objetivo se normaliza antes de calcular la distancia de hacinamiento. El algoritmo, como se muestra en la Tabla 2.2, describe el procedimiento de cálculo de la distancia de hacinamiento de todas las soluciones en un conjunto de soluciones no dominadas $\mathfrak{T}$.

Aquí, $\mathfrak{T}[i] * m$ se refiere al valor de la función objetivo m-ésima del individuo i-ésimo en el conjunto $\mathfrak{T}$, y los parámetros f_m^{max} y f_m^{min} son los valores máximos y mínimos de la función objetivo m-ésima. La complejidad de este procedimiento viene determinado por el algoritmo de

clasificación. Donde están involucrados M clasificaciones independientes en la mayoría de las N soluciones (cuando todos los miembros de la población están en un frente $\mathfrak{T}$). El algoritmo que se muestra en la Tabla 2.2 tiene una complejidad computacional $O(MNlogN)$.

Tabla 2.2 Algoritmo para el Cálculo de la Distancia de hacinamiento

Asignación_Distancia_Hacinamiento($\mathfrak{T}$)	
$\boldsymbol{l = \mathfrak{T}}$	Número de soluciones en $\mathfrak{T}$
$\boldsymbol{for\ each\ i, set\ \mathfrak{T}[i]_{distance} = 0}$	Inicializar la distancia
$\boldsymbol{for\ each\ objective\ m}$	
$\mathfrak{T} = sort(\mathfrak{T}, m)$	Ordenamiento usando cada valor objetivo
$\mathfrak{T}[1]_{distance} = \mathfrak{T}[l]_{distance} = \infty$	De modo que los puntos de contorno se seleccionen siempre
$for\ i = 2\ to\ (l-1)$	Para todos los otros puntos
$\mathfrak{T}[i]_{distance} = \mathfrak{T}[i]_{distance} + (\mathfrak{T}[i+1].m - \mathfrak{T}[i-1].m)/(f_m^{max} - f_m^{min})$	

Una vez se ha asignado a todos los miembros de la población en el conjunto $\mathfrak{T}$ la métrica de distancia, se pueden comparar dos soluciones para su grado de proximidad con otras soluciones. Una solución con un valor menor de esta medida de distancia está, en cierto sentido, más hacinada que otras soluciones. Esto es exactamente lo que se evalúa con el operador de comparación de hacinamiento propuesto, que se describe a continuación.

Operador de comparación de hacinamiento: El operador de comparación de hacinamiento ($\prec_n$) guía el proceso de selección en las diversas etapas del algoritmo hacia un frente de Pareto óptimo extendido uniformemente. Suponiendo que cada individuo de la población tiene dos atributos, el rango de no dominancia (i_{rank}) y la distancia de hacinamiento ($i_{distance}$), se define el orden parcial $\prec_n$ como:

$$i \prec_n j \qquad if(i_{rank} < j_{rank}) or\big((i_{rank} = j_{rank}) and (i_{distance} > j_{distance})\big) \qquad (2.19)$$

Es decir, entre dos soluciones con diferentes rangos de no dominancia, se prefiere la solución con el (menor) rango inferior. De lo contrario, si ambas soluciones pertenecen al mismo frente, entonces se prefiere la solución que se encuentra en una región de menor hacinamiento.

Con estas tres nuevas innovaciones: un procedimiento de clasificación de no dominados rápido, un procedimiento de estimación de la distancia de hacinamiento rápido, y un operador de comparación de hacinamiento simple, se puede describir el algoritmo NSGA-II.

2.7.3. Bucle Principal

Inicialmente, se crea una población de padres P_o de forma aleatoria. La población se clasifica en base a la no dominancia. A cada solución se le asigna un fitness (o rango) igual a su nivel de no dominancia (1 es el mejor nivel, 2 es el siguiente mejor nivel, y así sucesivamente). Al principio, la selección del torneo binario, la recombinación y los operadores de mutación habituales se utilizan para crear una población descendiente Q_o de tamaño N. Donde se introduce el elitismo mediante la comparación de la población actual con las mejores soluciones no dominadas que se han encontrado previamente, el procedimiento es diferente después de la generación inicial. En primer lugar, se describe la generación t-ésima del algoritmo propuesto, como se muestra en la Tabla 2.3.

Tabla 2.3 Algoritmo Elitista

$\boldsymbol{R_t = P_t \cup Q_t}$	Combina la población de padres y descendientes
$\boldsymbol{\mathcal{F} = \text{Clasificación\_No\_Dominada\_Rápida}(R_t)}$	$\mathcal{F} = (\mathcal{F}_1, \mathcal{F}_2, \dots)$, todos los frentes no dominados de R_t
$\boldsymbol{P_{t+1} = \emptyset \quad y \quad i = 1}$	
$\textbf{until } \lvert P_{t+1}\rvert + \lvert \mathcal{F}_i\rvert \leq N$	Hasta que la población padre esté llena
$\text{Asignación\_Distancia\_Hacinamiento}(\mathcal{F}_i)$	Calcula la distancia de hacinamiento en $\mathcal{F}_i$
$P_{t+1} = P_{t+1} \cup \mathcal{F}_i$	Incluye el i-ésimo frente no dominado en la población padre
$i = i + 1$	Verifica el próximo frente para inclusión
$\textbf{Sort}(\mathcal{F}_i, \prec_n)$	Clasifica en orden descendente usando $\prec_n$
$\boldsymbol{P_{t+1} = P_{t+1} \cup \mathcal{F}_i\,[1:(N - \lvert P_{t+1}\rvert)]}$	Elige los primeros $(N - \lvert P_{t+1}\rvert)$ elementos de $\mathcal{F}_i$
$\boldsymbol{Q_{t+1} = \text{make\_new\_pop}(P_{t+1})}$	Usa selección, cruzamiento y mutación para crear una nueva población Q_{t+1}
$\boldsymbol{t = t + 1}$	Incrementa el contador de generación

En primer lugar se forma una población combinada $R_t = P_t U Q_t$. La población R_t es de tamaño $2N$. Luego, la población R_t se ordena de acuerdo a la no dominancia. Dado que todos los miembros anteriores y actuales de población se incluyen en R_t, se asegura el elitismo. Ahora, las soluciones pertenecientes al mejor conjunto $\mathcal{F}_1$ de no dominadas, son de las mejores soluciones en la población combinada y deben hacerse hincapié más que en cualquier otra solución en la población combinada. Si el tamaño de $\mathcal{F}_1$ es menor que N, definitivamente se eligen todos los miembros del conjunto $\mathcal{F}_1$ para la nueva población P_{t+1}. Los miembros restantes de la población P_{t+1} se eligen de frentes no dominados posteriores en el orden de su ranqueo. Por lo tanto, las soluciones del conjunto $\mathcal{F}_2$ se eligen a continuación, seguidas por las soluciones del conjunto $\mathcal{F}_3$, y así sucesivamente. Este procedimiento continúa hasta que no hay más conjuntos que pueden ser acomodados. El conjunto $\mathcal{F}_{last}$ es el último conjunto de no dominados establecido más alla de cualquier otro conjunto que se pueda acomodar. En general, la cantidad de soluciones en todos los conjuntos de $\mathcal{F}_1$ a $\mathcal{F}_{last}$ sería más grande que el tamaño de la población.

Para elegir puntualmente N miembros de la población, se clasifican las soluciones del último frente de $\mathcal{F}_1$ usando el operador de comparación de hacinamiento en orden descendente y se eligen las mejores soluciones necesarias para completar población. El procedimiento NSGA-II se esquematiza en la Figura 2.9. La nueva población P_{t+1} de tamaño N se utiliza ahora, tras la selección, cruzamiento y mutación, para crear una nueva población Q_{t+1} de tamaño N. Es importante tener en cuenta que se utiliza un operador de selección de torneo binario, pero el criterio de selección se basa ahora en el operador de comparación de hacinamiento $\prec_n$. A partir de este operador se requiere tanto el rango como la distancia de hacinamiento de cada solución en la población, se calculan estas cantidades, mientras se forma la población P_{t+1}, como se muestró en el algoritmo anterior.

Considerando la complejidad de una iteración de todo el algoritmo. Las operaciones básicas y la complejidad de sus peores casos son las siguientes:

- La clasificación de no dominados es $O(M(2N)^2)$;
- La asignación de distancia de hacinamiento es $O(2NM \log(2N))$;

- La clasificación en $\prec_n$ es $O(2N \log(2N))$.

La complejidad total del algoritmo es $O(MN^2)$, que viene determinada por la parte de clasificación de no dominancia del algoritmo. Si se realiza con cuidado, la población completa de tamaño $2N$ no necesita ser ordenada de acuerdo a la no dominancia. Tan pronto como el procedimiento de clasificación ha encontrado suficiente número de frentes para tener $2N$ miembros en P_{t+1}, no hay razón para continuar con el procedimiento de clasificación.

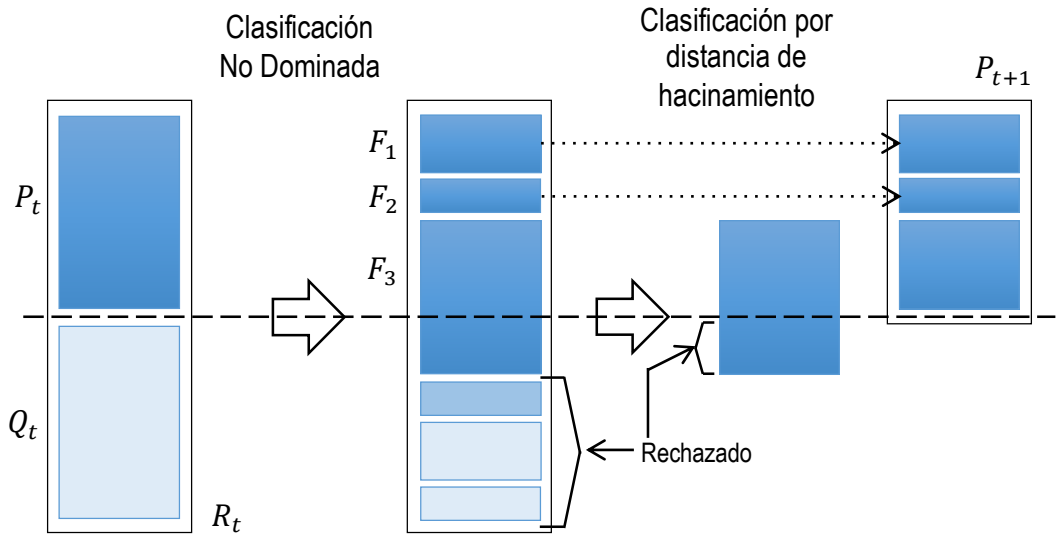

Figura 2.9. Procedimiento NSGA-II

La diversidad entre las soluciones no dominadas se introduce mediante el procedimiento de comparación de hacinamiento, que se utiliza en el torneo de selección y durante la fase de reducción de la población. Ya que las soluciones compiten a través de sus distancias de hacinamiento (una medida de la densidad de las soluciones en la vecindad), no se requiere ningún parámetro de nicho extra (como se necesitaba el $\boldsymbol{\sigma}_{\text{share}}$ en el NSGA). Aunque la distancia de hacinamiento se calcula en el espacio de la función objetivo, también se puede implementar en el espacio de parámetros.

Aquí, se propone la utilización del algoritmo NSGA-II [222] para la optimización multi-objetivo, construyendo un enfoque de selección de características *wrapper* que permite seleccionar subconjuntos de características específicas para cada etiqueta de clase. Además se utiliza el coeficiente de *Jaccard* para efectuar una medición del rendimiento del clasificador para cada clase.

2.8 CONCLUSIONES

La razón principal por la cual se ha decido seleccionar el modelo GHSOM como técnica de entrenamiento y clasificación, está en los beneficios que ofrece respecto a un SOM estándar. En [204] se han identificado algunas mejoras. En primer lugar, el tiempo total de entrenamiento se reduce en gran medida ya que se desarrolla sólo el número necesario de unidades para organizar

la colección de datos con grado de detalle deseado. En segundo lugar, el GHSOM descubre la estructura jerárquica de los datos mediante su propia arquitectura, permitiendo así al usuario entender y analizar grandes cantidades de datos. En tercer lugar, con mapas emergentes de pequeño tamaño en cada capa de la jerarquía, será mucho más fácil para el usuario mantener una visión general de los diferentes grupos. Por último, asegurando una orientación global coherente de los mapas individuales en las respectivas capas, se conservan las similitudes topológicas de mapas vecinos. Por lo tanto, se facilita la navegación a través de los mapas límites, permitiendo la exploración de las agrupaciones similares que están representadas por las ramas vecinas en la estructura GHSOM.

La computación evolutiva ha sido también aplicada a la detección de intrusos [262], [263] y [265]. En [262], la computacional evolutiva optimizada y las RBF son utilizadas en la tarea de clasificación. En [226] y [267] presentan sistemas híbridos, por ejemplo, en [226] se combinan sistemas inmunes artificiales y SOM, para detectar patrones de tráfico anormales y para categorizar estos con agrupamiento (*clustering*) SOM. Además, técnicas evolutivas para detección de intrusiones en redes móviles y *ad hoc* se han utilizado en [228]. En general, los SOMs son ampliamente utilizados en minería de datos para descubrir las similitudes en datos de alta dimensionalidad, generando un número de prototipos correspondientes al número de unidades de SOM, que generaliza el colector de datos. Además, una característica única de SOM es que las unidades están dispuestas en un espacio 2D o 3D donde los prototipos correspondientes a los más similares se mueven juntos. Por otro lado, las unidades se mueven a medida que los prototipos correspondientes son diferentes entre ellos [178].

De acuerdo con las características antes mencionadas, las SOMs pueden verse también como un método de reducción de dimensión. Sin embargo, el uso de SOM presenta algunas dificultades, sobre todo cuando la dimensión del espacio de características (entrada) es muy alta. Por otra parte, las dimensiones y el tamaño del SOM tiene que ser establecido antes del proceso de entrenamiento, y sus valores óptimos tienen que determinarse mediante ensayo y error. Sin embargo, el número de neuronas en el espacio de salida determina la calidad del mapa y, a continuación, el rendimiento del proceso de clasificación (es decir, las clases deben presentar suficientes diferencias entre ellas y deben estar lo suficientemente alejadas del resto de clases en el espacio de salida). De este modo, trabajos tales como [257], [258] y [259] y [268] muestran el uso de estructuras jerárquicas para mejorar el proceso de clasificación.

Como se ha comentado antes, una red neuronal tipo perceptrón jerárquica se presenta en [257] y el uso de estructuras SOM jerárquicas se consideran en [258] y [259]. El primer enfoque utiliza un conjunto de características para el entrenamiento de las redes perceptrón, mientras que el último utiliza un primer nivel constituido por árboles SOMs (cada SOM representa una clase de características) y un segundo nivel que resume los grupos encontrados en el primero. Aunque la propuesta descrita en [258] trata de superar algunas de las dificultades encontradas en la estructura estática SOM clásica dividiéndolo en tres mapas, los tamaños de estos mapas son todavía estáticos. Con el fin de evitar algunas de estas limitaciones, los SOM jerárquicos crecientes (GHSOM) se han propuesto en [202].

GHSOM es una estructura dinámica en la que el mapa crece durante el proceso de entrenamiento para reducir al mínimo su error de cuantización. Así, el modelo GHSOM presenta una arquitectura jerárquica compuesta de varias capas, también con varios SOM por capa. El número de SOMs por capa y el tamaño de cada SOM se determinan durante el entrenamiento del GHSOM. Hay varios

trabajos que han aplicado GHSOM al diseño de IDS [269] y han propuesto varias mejoras relativas al rendimiento de GHSOM [266] y [250].

CAPÍTULO 3. CLASIFICACIÓN DE ATAQUES CON GHSOM Y OPTIMIZACIÓN MULTIOBJETIVO

3.1 INTRODUCCIÓN

Con frecuencia, los procesos de clasificación de características existentes no son lo suficientemente discriminatorios. Por otra parte, el uso de las características correctas mejora el rendimiento de la clasificación y reduce el tiempo computacional. De allí la importancia de aplicar las técnicas más discriminatorias de selección o extracción de características, con el propósito de obtener un subconjunto de características en un espacio de menor dimensionalidad.

En este capítulo, en primera instancia se hace una descripción del coeficiente de Jaccard como medida de la asimetría de la información, el cual se integra en esta propuesta, con el algoritmo NSGA-II aplicado al proceso de selección de características (sección 3.2), luego se presenta la propuesta de optimización multiobjetivo basada en NSGA-II (sección 3.3), a continuación se detalla la propuesta de clasificación basada en GHSOM, la cual integra una novedosa técnica de re-etiquetado probabilístico (sección 3.4). El capítulo culmina con las correspondientes conclusiones.

3.2 COEFICIENTE DE JACCARD

Puesto que el rendimiento del clasificador depende del conjunto de características utilizadas, es necesario llevar a cabo una selección de características adecuada. Como se ha dicho, el problema de selección del subconjunto de características consiste en la aplicación de un algoritmo de aprendizaje para la selección de un subconjunto de las características existentes sobre las que se centra la atención, mientras se ignora el resto. Específicamente, en los enfoques wrapper, se seleccionan subconjuntos de características para maximizar el valor de un criterio específico como la precisión de la clasificación en los clasificadores supervisados [248], [249] o medidas indirectas como la calidad del agrupamiento en los no supervisados. Por lo tanto, el enfoque basado en *wrapper* se puede focalizar como un método de selección de características de clasificador específico (CSFS) [249]. De todos modos, la selección de características mediante la búsqueda exhaustiva, que finalmente resulta en la prueba de todos los posibles subconjuntos de características, no es factible en términos de tiempo de procesamiento. En consecuencia, el uso de técnicas matemáticas que evitan evaluar todos los subconjuntos de características proporcionan una manera eficaz para encontrar características relevantes. A continuación, se propone un procedimiento de selección de características basado en la optimización multiobjetivo. El problema consiste en encontrar las características que maximizan el rendimiento del clasificador, y puesto que el dataset está etiquetado, puede ser definida una medida de similitud, con el fin de evaluar simultáneamente el número de elementos que son clasificados correctamente. Nuestra propuesta en esta tesis es utilizar el *coeficiente de Jaccard* [209] y [210], que es una medida de la asimetría de la información en las variables. En otras palabras, es una medida de similitud entre *datasets*.

Si **Cg** son las etiquetas verdaderas que provee el *dataset* para identificar la clase correspondiente a cada patrón, y **Cs** es el resultado de la clasificación, podemos determinar el *coeficiente de Jaccard* para una clase específica entre las etiquetas verdaderas y las etiquetas calculadas por nuestro clasificador, a partir de (3.1).

$$J(clase) = \frac{p}{p + fn + fp} \tag{3.1}$$

donde

- $\boldsymbol{p}$ es el número de elementos clasificados correctamente (**verdaderos positivos**).
- $\boldsymbol{fn}$ es el número de elementos etiquetados como **a** ∈ **Cg** en el *dataset*, pero no etiquetado como un **a** ∈ **Cs** por el clasificador (**falso negativo**).
- $\boldsymbol{fp}$ el número de no-etiquetados como **a** ∈ **Cg** en el *dataset* y etiquetados **a** ∈ **Cs** por el clasificador (**falsos positivos**).
- $\boldsymbol{clase}$ indica si se trata de tráfico sin ataques ($clase$ = normal) o de algún tipo de ataque (*DoS, PROBE, U2R o R2L*).

Por lo tanto, el *coeficiente de Jaccard* [209] proporciona una medición del rendimiento del clasificador para la clase correspondiente. Este coeficiente se puede utilizar para seleccionar las características que se ajustan mejor a una clase específica, ya que estas características seleccionadas maximizarán el correspondiente *coeficiente de Jaccard*. Optimizando el rendimiento del clasificador lo que significa maximizar, al mismo tiempo, un número de funciones objetivo iguales al número de clases. Cada una de estas funciones objetivo corresponde al *coeficiente de similitud de Jaccard* para la clase dada.

De este modo, como se ha indicado, los cinco objetivos utilizados para la selección de características en el problema de detección de anomalías de red son J(normal), J(DoS), J(PROBE), J(R2L) y J(U2R). y p, fn y fp en la ecuación (3.1) se refieren a los positivos, falsos negativos y falsos positivos, respectivamente, para cada etiqueta de clase. Como este trabajo se centra en la selección de características, se utilizó un *índice de similaridad* como la función objetivo para cada etiqueta de clase con el fin de maximizar esta similitud entre las etiquetas de clase predichas y las reales. De esta manera, la precisión (*accuracy*) no se utiliza directamente en el método de selección de características, sino más bien en la evaluación.

3.3 PROPUESTA DE OPTIMIZACIÓN MULTIOBJETIVO BASADA EN NSGA-II

En este trabajo, como resultado de la implementación de un proceso de optimización multi-objetivo basado en el algoritmo NSGA-II [222] y [245], y descrito en el algoritmo propuesto (ver Tabla 3.1 y Figura 3.1), se obtiene el frente de Pareto que resume las soluciones no dominadas encontradas por el proceso de optimización multi-objetivo. Por lo tanto, el frente Pareto contiene suficiente información para seleccionar las características de cada tipo de ataque. El punto clave es alimentar el clasificador con las características más discriminantes para detectar los elementos pertenecientes a la clase de interés. Por lo tanto, utilizando un conjunto reducido de características aprovecha el rendimiento de la clasificación mientras que se reduce el tiempo de cálculo para el entrenamiento.

El siguiente pseudocódigo detalla el proceso de optimización multi-objetivo para la selección de características con NSGA-II [222] y [245].

Tabla 3.1 Algoritmo propuesto

Secuencia	Instrucciones
1:	Generar una población inicial
2:	**Evaluar los valores de los objetivos:**
3:	Entrenamiento del clasificador
4:	Clasificación del *dataset*
5:	Cálculo de los *coeficientes de Jaccard*
6:	**Ciclo NSGA-II**
7:	Asignación del rango basado en Dominancia de Pareto como en NSGA-II para seleccionar los individuos no dominados
8:	Aplicar operadores evolutivos para generar una nueva población
9:	Evaluar los valores objetivo
10:	Entrenamiento del clasificador
11:	Clasificación del *dataset*
12:	Cálculo de los *coeficientes de Jaccard*
13:	**Terminar el ciclo NSGA-II**

En la sección "Resultados experimentales" del capítulo 4, se compara el rendimiento de este procedimiento con los resultados obtenidos usando todo el conjunto de características y el filtrado basado en PCA para reducir la dimensión del *dataset*. La configuración de los parámetros para la ejecución de NSGA-II se detalla también en dicha sección.

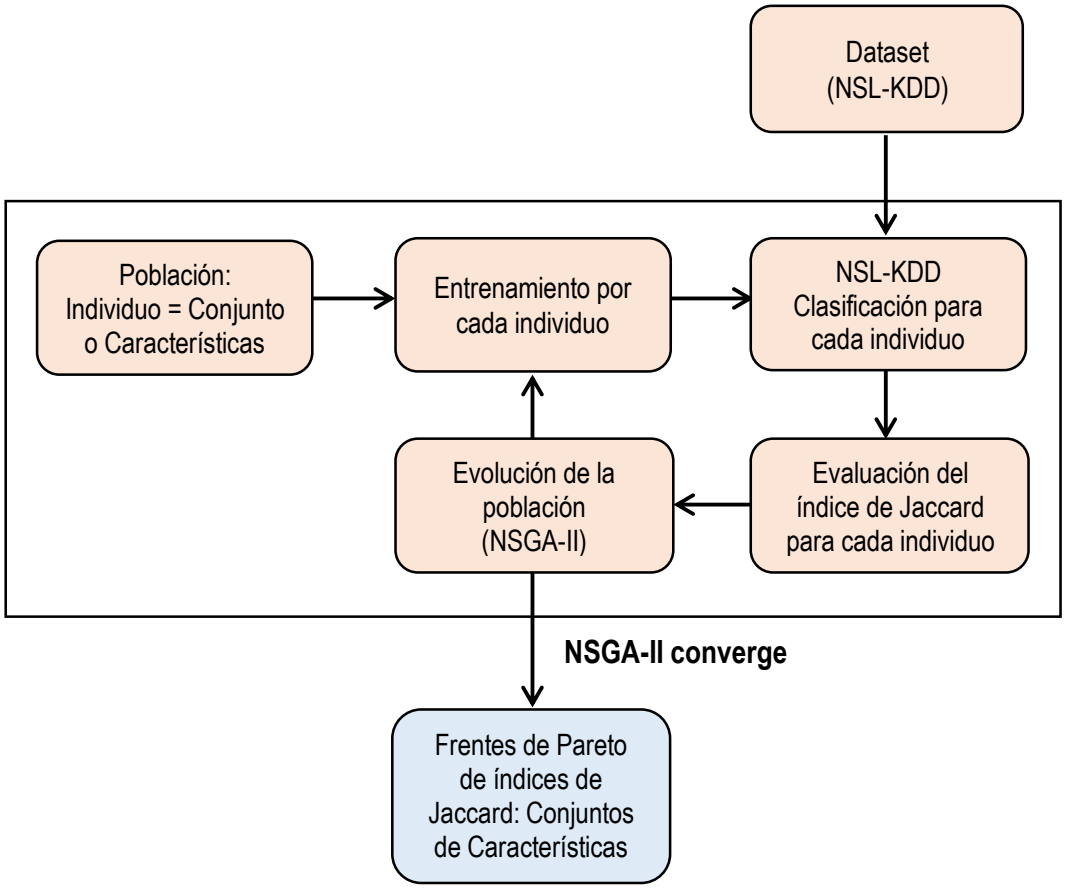

Figura 3.1. Esquema para la optimización multi-objetivo con el algoritmo NSGA-II en selección de características

3.4 PROPUESTA DE CLASIFICACIÓN CON GHSOM

En este trabajo, GHSOM se utiliza tanto para la detección de anomalías como para la clasificación de ataques. El enfoque aquí propuesto ligeramente sobredimensiona la GHSOM a lo largo del proceso de entrenamiento, dejando algunas unidades sin etiquetado (unidades que nunca ganan para cualquier patrón de entrenamiento). Sin embargo, estas unidades pueden utilizarse para propagar los grupos (clusters) aplicando una estrategia mecánica basada en probabilidad, que se describe en la sección "Clasificación con GHSOM", para etiquetar las unidades no etiquetadas en la estructura GHSOM previamente entrenada.

El procedimiento de clasificación utilizado en la técnica *wrapper* que se propone en este trabajo de tesis (el paso 10 en el algoritmo propuesto en la Tabla 3.1), se basa en un SOM de jerarquía creciente (GHSOM), anteriormente descrito. En esta sección se hace especial énfasis en algunas mejoras al modelo GHSOM, en relación a la implementación del método de etiquetado probabilístico que proponemos. Aunque los detalles sobre los algoritmos SOM y GHSOM se han descrito anteriormente en este documento, una información más detallada respecto a ellos, se pueden encontrar en [178] y [202]. La principal motivación por la cual se utilizó GHSOM, en este trabajo, está relacionada con la forma en la que se puede obtener la neurona ganadora en el mapa, también denominada Unidad de Mejor Acoplamiento (BMU).

Dada la detallada fundamentación que se hizo en un capítulo anterior sobre la clasificación basada en GHSOM, a continuación se evidencia cómo se implementa dicha técnica en la propuesta aquí planteada, describiendo las mejoras realizadas en materia de la aplicación de "GHSOM con re-etiquetado probabilístico (GHSOM-pr)", con selección de características mediante un procedimiento tipo *wrapper*.

3.4.1. GHSOM con re-etiquetado probabilístico (GHSOM-pr)

En esta sección, se describe el procedimiento que proponemos para etiquetar las unidades GHSOM que hacen posible un proceso de agrupamiento mediante comportamiento probabilístico. Los valores de los parámetros $\boldsymbol{\tau 1}$ y $\boldsymbol{\tau 2}$, previamente introducidos, han de ser cuidadosamente configurados para lograr un proceso de entrenamiento eficiente. Por ejemplo, valores muy altos para $\boldsymbol{\tau 1}$ y $\boldsymbol{\tau 2}$ hacen que el crecimiento del GHSOM, deje algunas de las unidades sin etiquetar. Por lo tanto, durante el entrenamiento de los datos se aumenta el número de neuronas en el mapa de salida que rodean la neurona ganadora. Siempre que un patrón de entrada similar a uno de los patrones de entrenamiento se presenta al GHSOM, la neurona ganadora puede estar etiquetada o sin etiquetar. La etiqueta de una unidad determina la clase de datos a la que pertenece. Si la neurona ganadora (BMU) está sin etiquetar, se aplicará un procedimiento basado en probabilidades con el fin de determinar la etiqueta de esa neurona. En este procedimiento, la neurona ganadora se vuelve a etiquetar con la etiqueta más repetida en su vecindad mediante el uso de una probabilidad determinada de acuerdo con la siguiente ecuación.

$$P_u = \frac{M_{\sigma(u,\epsilon)}(u)}{n} \tag{3.2}$$

En esta expresión (3.2), P_u es la probabilidad de que la unidad ganadora u sea exitosamente re-etiquetada, $M_{\sigma(u,\epsilon)}(u)$ es el número de unidades en el vecindario gaussiano $\sigma(u,\epsilon)$ de la unidad ganadora u, que está marcada con la etiqueta más frecuente en $\sigma(u,\epsilon)$, y n es el número de neuronas que pertenecen a $\sigma(u,\epsilon)$. En esta ecuación, el parámetro ϵ determina la anchura del vecindario de la neurona ganadora. Aunque a través de este procedimiento se le asigna una etiqueta a una unidad no marcada, hemos denominado este como procedimiento de re-etiquetado, ya que el término '"etiquetado" se aplica por lo general sólo al proceso de entrenamiento. En la Figura 3.2, se muestra un ejemplo del proceso de reetiquetado. Las unidades blancas están sin etiqueta. En dicha figura, la BMU no tiene que estar marcada inicialmente, y es reetiquetada utilizando $\epsilon = 1$ para establecer su vecindario. En este vecindario, se encontraron cuatro unidades marcadas como L1, una unidad etiquetada como L2 y una unidad etiquetada como L3. Entonces, P_u, la probabilidad a priori para re-etiquetado exitoso para esta BMU es 4/6 = 0,66 (66%) dado que:$(M_{\sigma(u,\epsilon)}(u) = 4$ y n=6.

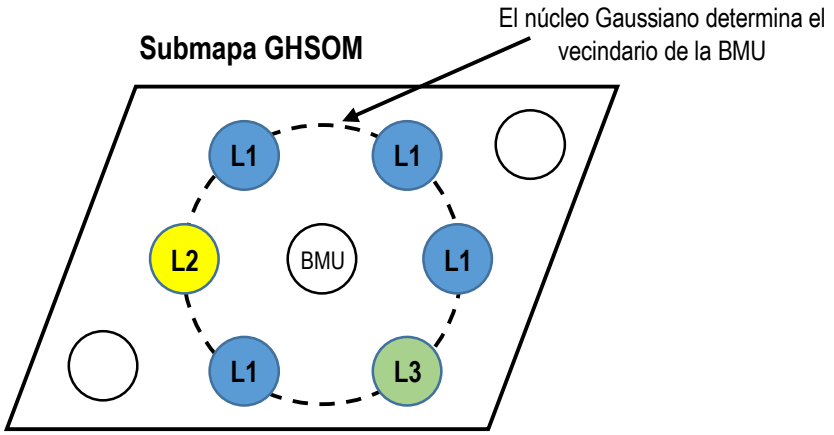

Figura 3.2. Ejemplo del proceso de re-etiquetado

La probabilidad de re-etiquetado a posteriori para la unidad k se puede calcular por medio del teorema de Bayes (3.3):

$$p(w_k|x) = \frac{p(x|w_k)P(w_k)}{p(x)} \tag{3.3}$$

En la ecuación (3.3), $p(w_k|x)$ representa la probabilidad a posteriori que un vector muestra x pertenezca a la clase w_k, mientras que $p(x|w_k)$ es la probabilidad condicional de x en la clase x, $P(w_k)$ es la probabilidad a priori y $p(x)$ es una constante de normalización, calculada como:

$$p(x) = \sum_{k=1}^{n} p(x|w_k)P(w_k) \tag{3.4}$$

La probabilidad a posteriori se puede utilizar para clasificar nuevas muestras. En este trabajo, las probabilidades a posteriori se han utilizado para re-etiquetar las unidades que permanecen sin etiqueta durante el proceso de entrenamiento del SOM. De esta forma, el re-etiquetado dinámico

de las unidades de mapeo presentado en esta sección introduce un comportamiento del proceso de clustering probabilístico que suele mejorar los modelos GHSOM. De hecho, en [119] se utiliza GMM (*Gaussian Mixture Model*) para modelar los grupos (*clusters*) en SOM. A continuación, se aplica un método de etiquetado probabilístico, pero sólo se tienen en cuenta las probabilidades a priori calculadas según la ecuación (3.2) (utiliza la etiqueta más frecuente en el vecindario). Por otra parte, [123] utiliza un esquema similar de etiquetado para las *unidades muertas* [178], sobre la base de probabilidades a priori. Sin embargo, el método propuesto en nuestro trabajo utiliza probabilidades a posteriori a través de la fórmula de *Bayes* para calcular la etiqueta más probable, de acuerdo con la ecuación (3.3). Esto constituye una mejora importante, ya que el cálculo de etiquetas de unidad no sólo depende de las etiquetas del mapa, sino también de la muestra actual, asignando la etiqueta de cada BMU de una forma adaptativa.

3.5 CONCLUSIONES

En este trabajo, se propone el uso de una estructura dinámica, GHSOM-pr, basada en la propuesta previa GHSOM, y un nuevo enfoque para la reducción de dimensionalidad a través de la selección de características basada en la optimización multi-objetivo.

Algunos trabajos sobre GHSOM para el diseño de IDS se han propuesto anteriormente [250], [266] y [269]. Después de los resultados proporcionados por [269], se han propuesto algunas mejoras en el procedimiento GHSOM en [266] y [250]. En [266], se introduce una nueva métrica que incluye datos numéricos y simbólicos junto con un procedimiento que permita un control del crecimiento automático del mapa que evite el uso del parámetro $\boldsymbol{\tau 1}$. El documento muestra que la optimización a GHSOM es mejor que un SOM usado como la configuración base del mapa y proporciona tasas de detección por debajo de 96,9%. La misma estrategia se considera en [250], donde se proponen cuatro mejoras a GHSOM. Estas mejoras están relacionadas con el entrenamiento, la normalización de entrada, la adaptación del error de cuantificación, y un mecanismo para el filtrado de confianza y el reenvío. El mapa autorganizativo jerárquico resultante, es llamado A-GHSOM y ofrece una precisión (*accuracy*) global de 99,63% y una tasa de falsos positivos del 1,8%.

En esta sección se hace una descripción de los trabajos relacionados que han generado las principales contribuciones a esta investigación. En cuanto a la selección de características en IDS basados en red no es sencillo enumerar los artículos, dado el volumen y la variedad de publicaciones en relación a este tema. Hay trabajos [275], [281] y [282] que utilizan técnicas multivariantes como PCA [275] o LDA [281], suponiendo que los demás componentes no son importantes o son sólo ruido. Aunque se obtienen buenos resultados en [275], [281] y [282], otros trabajos muestran que el uso de todo el conjunto de características es superado mediante la selección de características con PCA/LDA [254]. De esta manera, la reducción de características tiene que ser aplicada a cada vector de entrada ya que el componente más discriminativo depende del ataque específico. Así, en [223], un clasificador SOM para la detección de ataques se presenta sin reducir el espacio de características. Sin embargo, los autores utilizan una selección de 28 características de las 41 que están disponibles en el dataset KDD-NSL. En este trabajo, la influencia de cada característica se calcula a través de la U-Matrix de cada componente [178] en el espacio de entrada (característica).

En [268], se analiza la selección de características, junto con su interacción con la partición de datos de entrenamiento, y el número de capas de una SOM jerárquica. LIegaron a la conclusión de que una SOM jerarquía de dos capas basada en las 41 características del conjunto KDD es la

mejor opción. Se reportó una tasa de detección del 90,4% y una tasa de falsos positivos del 1,38%.

Producto del análisis de los temas anteriormente fundamentados, se puede concluir que la selección de características basada en técnicas multi-objetivo es adecuada siempre que:

- El desempeño de los experimentos reduzca el conjunto de características usando técnicas multivariantes y lineales (como PCA), mostrando que no es posible reducir de manera efectiva la dimensión del espacio de característica utilizando técnicas lineales, en tal caso se deberían utilizar métodos estocásticos o no lineales.
- Las características que maximizan un tipo de ataque no siempre maximizan la tasa de detección para todas las clases. En otras palabras, la maximización de la tasa de detección para todas las clases se ocupa de objetivos mutuamente exclusivos.

Como se ha dicho anteriormente, la selección de características claramente determina el rendimiento de la clasificación. Además, en [210] se indica la formulación de la selección de características como un problema de optimización multi-objetivo, dado que puede proporcionar algunas ventajas. Sin embargo, los resultados dependerán de si el procedimiento de clasificación es supervisado o no. En los procedimientos de clasificación supervisada, el objetivo ha sido por lo general maximizar el rendimiento del clasificador, mientras que el número de características se minimiza para evitar que conjuntos de características de mayor tamaño puedan producir problemas de sobreajuste y de baja generalización. De esta manera, un enfoque de optimización multi-objetivo que tiene en cuenta el rendimiento del clasificador y el número de características, permite una formulación adecuada de esta situación.

En problemas de clasificación no supervisada es diferente. En este caso, es difícil evaluar la agrupación (clustering), y las técnicas de validación aplicadas generalmente presentan un sesgo en la dimensionalidad a cualquiera de los conjuntos de características de cardinalidad, más pequeños, o más grandes. Por lo tanto, un enfoque multi-objetivo puede contrarrestar el sesgo específico del método de validación de clúster considerado.

El cálculo de una puntuación media en diferentes clases de *dataset* puede dar lugar a la eliminación de las características de la selección final, lo que podría ser especialmente relevante para una etiqueta de clase determinada. Por lo tanto, es necesario tener en cuenta métodos que evalúan el poder discriminativo de cada característica, seleccionando aquellas que mejor describan cada clase individual. En consecuencia, el método propuesto en [116] utiliza un conjunto de clasificadores SVC especializados, cada entrenamiento usando características específicas calculadas para cada tipo de anomalía. Sin embargo, los experimentos realizados en [116] revelan que ni las técnicas de proyección lineal, ni las no lineales, son capaces de encontrar características suficientemente discriminativas. Por lo tanto, se han propuesto algunos trabajos relacionados con la formulación de selección de características como un problema de optimización multi-objetivo, ya sea como clasificadores supervisados o no supervisados. Una muy buena revisión de las alternativas y las referencias previas sobre este tema se encuentran en el artículo de Handl y Knowles [210]. Con respecto a los clasificadores supervisados, en [283], se proveen procedimientos de selección de características multi-objetivos que tienen en cuenta el número de características y el rendimiento del clasificador. En [284], se aplica una variación del algoritmo NPGA [243] para la optimización multi-objetivo, mientras que en [285] utilizan NSGA y, en [238] y [283] utilizan NSGA-II. En este trabajo, también hemos utilizado NSGA-II, como en [283]. En nuestro caso, una medida de similitud entre las etiquetas predichas y las reales para cada tipo de anomalía se trata como un objetivo diferente a ser maximizado conjuntamente. Aunque el número

de características no se toma en cuenta como una función objetivo, nuestros resultados muestran que los subconjuntos que corresponden a soluciones no dominadas contienen menos, que los subconjuntos de características discriminantes para cada etiqueta de clase. Por lo tanto, el enfoque propuesto en [283], no tiene en cuenta cada etiqueta de cada clase de forma independiente en el proceso de selección, y se utiliza k-media como procedimiento clasificador basado en *wrapper* en lugar de GHSOM, como en nuestro caso. En [221], se propone un marco de selección de características que tiene en cuenta la pertinencia y la redundancia de las características seleccionadas para las diferentes etiquetas de clase. Sin embargo, este artículo no enfoca el problema como una búsqueda de frente de Pareto. Desde una perspectiva de optimización multi-objetivo, como en nuestra propuesta, los coeficientes de Jaccard para las diferentes etiquetas (utilizado como medida de rendimiento del clasificador para cada etiqueta) son los objetivos del problema considerado.

En [286] el algoritmo NSGA-II, junto con otros procedimientos propuestos, se aplica para seleccionar los subconjuntos de características no-dominadas globales para la predicción de la pérdida de clientes en telecomunicaciones. En este procedimiento de selección de características multi-objetivo, hay tres funciones de costos y la proporción del número total de predicciones correctas, la proporción de casos de deserción que se identificaron correctamente, y la proporción de casos que no son de deserción que se identificaron correctamente. Este tipo de información se tiene en cuenta en nuestro enfoque, pero a través de un coeficiente de Jaccard para cada etiqueta. En nuestro caso, los objetivos son, precisamente, los coeficientes de Jaccard y tenemos tantos objetivos como etiquetas. Por otra parte, la clasificación en [286] se hace a través del árbol de decisión C4.5 [287], por lo que corresponde a un enfoque diferente al descrito en el presente documento, que se basa en los mapas de auto-organizativos.

En el caso de clasificación no supervisada, se tienen [210], [288] y [289]. En [288], dada una selección de características, el algoritmo k-medias se utiliza para construir un agrupamiento (clustering) y se evalúa a través de cuatro objetivos (número de características, número de grupos, compactación de los grupos, y separación entre grupos). El artículo [289] también utiliza k-medias para el agrupamiento y el indice de Davies-Bouldin (DBI) [290] para el número de características. Junto con una revisión crítica de los trabajos [288] y [289], y un estudio experimental de diferentes alternativas para la selección de características sin supervisión con optimización multi-objetivo, el artículo [210] proporciona una estrategia para seleccionar (sin conocimiento externo) la solución más adecuada desde la aproximación obtenida del *frente de Pareto*.

CAPÍTULO 4. ESTUDIO EXPERIMENTAL APLICADO A LA DETECCIÓN DE INTRUSOS

4.1 INTRODUCCIÓN

En este capítulo, se presentan los resultados experimentales obtenidos mediante la aplicación del clasificador GHSOM, descrito en capítulos anteriores, para el dataset NSL-KDD [110]. Inicialmente se hace un análisis de la razón por la cual se selecciona el dataset NSL_KDD (sección 4.2) y la configuración inicial de los escenarios de experimentación (sección 4.3), la importancia del pre-procesamiento (sección 4.4) y normalización (sección 4.5) de los datos para asegurar la homogeneidad en la contribución de las características a la medida de la distancia. A continuación se detallan los resultados experimentales entrenando el GHSOM tanto con la totalidad del conjunto de características (sección 4.6) como con una reducción previa del conjunto de características (sección 4.7). A partir de los resultados obtenidos con los escenarios de experimentación, se realiza una evaluación de las prestaciones del NSGA-II considerando la convergencia del algoritmo y el uso del índice de Jaccard (sección 4.8). Posteriormente se analiza la efectividad computacional de la propuesta, en relación al nivel de discriminación de las características para todas las clases para una cantidad aceptable de tiempo de cómputo (sección 4.9). De forma complementaria, y dado que los resultados de la clasificación pueden variar entre las diferentes corridas, se efectuaron pruebas de significación estadística, con el fin de analizar si los valores medios obtenidos mediante la aplicación de varios experimentos, son diferentes. Así a partir de los resultados obtenidos se elaboró un ANOVA (sección 4.10). El capítulo termina con las conclusiones correspondientes.

4.2 SELECCIÓN DEL DATASET

Si bien en el capítulo 1 se hizo una fundamentación en detalle de los datasets, incluyendo una descripción del NSL-KDD, se considera pertinente hacer aquí un muy breve resumen del NSL-KDD, evidenciando las razones de su escogencia.

- El dataset NSL-KDD proviene del dataset KDD'99 [270], que contiene alrededor de 4 GB de datos comprimidos de capturas de tráfico tcpdump. Según se indica en [271] y [272], esos datos corresponden a capturas de alrededor de siete semanas de tráfico de red. Los datos recolectados son de tres tipos diferentes (1) funciones básicas, (2) características basadas en tráfico y (3) características basadas en contenido.
- Su predecesor, el dataset KDD'99 presenta algunos problemas inherentes, tales como características sintéticas de los datos, según se indica en [272] y [251], y en consecuencia estas características, pueden no ser representativas de los ataques reales. Por lo tanto, el Laboratorio de Seguridad en Redes para el Descubrimiento del Conocimiento y la Minería de Datos propuso el dataset NSL-KDD.
- Las mejoras que tiene el NSL-KDD respecto a su predecesor son (1) se eliminaron los registros redundantes de KDD'99 y (2) los ataques se etiquetaron y ordenaron por su nivel de dificultad de detección.
- Numerosas publicaciones de alto impacto, relativas al campo de la detección de intrusiones en red, registradas en bases especializadas, usan el NSL-KDD en sus procesos de experimentación. La Tabla 4.1 muestra las estadísticas para los últimos cinco (5) años, de la cantidad de publicaciones que citan el dataset en mención, totalizada por base de datos.

Tabla 4. 1 Estadística de artículos que utilizan el dataset NSL-KDD, por base de dato

Base de dato Indexada	2011	2012	2013	2014	2015
SCOPUS	10	16	16	32	12
Springer	7	10	9	14	11
ScienceDirect	1	4	3	7	4
IEEEXplore Digital Library	5	11	10	13	7

Las razones anteriormente expuestas validan la elección del *dataset* NSL-KDD en la investigación aquí plasmada.

4.3 CONFIGURACIÓN DE LOS ESCENARIOS DE EXPERIMENTACIÓN

Los experimentos se han realizado de dos formas, la primera corresponde a experimentos de clasificación utilizando el conjunto de características completo (41 características), mientras que la segunda consiste en la aplicación de técnicas de reducción de la dimensionalidad del conjunto de características, con el fin de evitar el uso del conjunto completo de características. Esta reducción de dimensión se ha llevado a cabo, tanto con técnicas lineales (PCA) como con el método descrito en la sección 3.3 “Propuesta de Optimización Multiobjetivo basada en NSGA-II”. De esta forma podemos comprobar distintas alternativas de selección de características.

Con el fin de demostrar que el sistema no se sobreajusta y por lo tanto, tiene un buen rendimiento para la generalización, los procesos de selección de características y entrenamiento han sido evaluados mediante la validación cruzada, de orden k = 10. Ya que con k = 10, se hacen particiones aleatorias del 90% de las muestras utilizadas para ajustar el modelo y el resto de las muestras (10%) para la prueba. Estos subconjuntos son disyuntos y no comparten ninguna muestra. Este proceso se repitió 10 veces (tenemos k-fold con k=0), asegurándose de que los datos de prueba nunca se han utilizado en la selección de características o en el entrenamiento del clasificador. Por lo tanto, los resultados proporcionados por los subconjuntos de características seleccionadas y la precisión de la clasificación se calculan como la media de las 10 evaluaciones correspondiente a las 10 repeticiones o (10-fold).

El objetivo principal de la validación cruzada es estimar el error de generalización, asegurándose de que se obtuvieran resultados similares para nuevos datos (es decir, bajo error de generalización). Este método calcula el error de predicción y evita la doble inmersión. Además, vale la pena señalar que, debido al elevado número de muestras de los dataset disponibles, ambos procesos, entrenamiento y prueba, se tratan usando un alto número de muestras. Esto proporciona una menor estimación de la varianza del error de generalización. Además, se han realizado varias pruebas con diferentes valores para $\boldsymbol{\tau 1}$ (para el control de la amplitud del mapa) y $\boldsymbol{\tau 2}$ (para controlar la profundidad).

4.4 PRE-PROCESAMIENTO DE DATOS

A pesar de que el preprocesamiento de datos juega un papel importante en el desempeño de la clasificación, pocos trabajos prestan suficiente atención a ello [276]. El preprocesamiento de datos

comprende la codificación de las variables no continuas y la normalización. Como se describe en la sección anterior, los conjuntos de datos basados de KDD'99 constan de 41 características, que deberían ser suficientes para caracterizar las conexiones anómalas. Dichas características se clasifican en tres grupos, características continuas, simbólicas y binarias. Como la mayoría de los clasificadores sólo aceptan valores numéricos, el primer problema está relacionado con la codificación de las características simbólicas. En varios trabajos, normalmente se codifican simplemente sustituyendo cada característica diferente con un número entero [276] y [277]. Aunque esto puede ser aceptable en muchas situaciones, no es la mejor solución de codificación para los clasificadores basados en la distancia euclidiana [278]. De esta manera, se adopta una solución diferente que mapea cada característica simbólica a un subespacio R_d, donde ***"d"*** es el número de posibles valores de las variables discretas. Aunque esta solución aumenta la dimensionalidad de los datos (por ejemplo, la característica de servicio puede tomar 65 valores diferentes), no es crítico para los clasificadores utilizados en este trabajo. Además, las técnicas de reducción de dimensionalidad se utilizan para comprimir la información relevante con menos características. Por lo tanto, un valor diferente en estas características contribuye a la distancia medida $\sqrt{2}$.

4.5 NORMALIZACIÓN DE DATOS

La normalización de datos permite que ninguna característica contribuya más que otra a la medida de la distancia. Hay diferentes formas de normalización de datos [102]. En este trabajo, las variables continuas son normalizadas con media cero y varianza unitaria utilizando la siguiente ecuación:

$$\hat{x} = \frac{x - \bar{x}}{\sigma} \tag{4.1}$$

Donde $\vec{x}$ y σ son, respectivamente, la media y la desviación estándar de la variable x. Esto es equivalente a expresar la variable x como la distancia entre el número de desviaciones estándar y su media. Por otra parte, todas las variables se escalan en el intérvalo [0.1]. Las características simbólicas (ya codificadas con valores binarios) y las binarias no son normalizadas.

4.6 RESULTADOS EXPERIMENTALES

En primer lugar se utiliza el dataset NSL-KDD, con todas las características, para la clasificación. El propósito de estos primeros resultados es demostrar las mejoras obtenidas por el método de re-etiquetado de GHSOM anteriormente descrito en la sección 3.4.

En la figura 4.1, se muestra la tasa de detección para cada ataque presente en el dataset NSL-KDD (las anomalías en dicha figura se pueden agrupar como pertenecientes a un tipo de ataque, como se indica en la tabla 4.2). En la figura 4.1 la tasa de detección con reetiquetado de unidades se representa mediante barras negras y sin re-etiquetado de unidades mediante barras blancas. En este caso, GHSOM es entrenado con todo el conjunto de características (1384 neuronas, cinco capas). Como muestra la figura 4.1, el proceso propuesto de re-etiquetado probabilístico realizado, aumenta la tasa de detección de la mayoría de los ataques. La mejora promedio es de alrededor de 3,5% sobre todos los ataques, y el 5,5% en los ataques en los que el re-etiquetado crea mejoras.

Tabla 4.2. Clasificación de anomalías de red en diferentes tipos de ataques

Tipo de ataque	Nombre del ataque
DoS	apache2, back, land, mailbomb, neptune, pod, processtable, smurf, teardrop, udpstorm
Probe	ipsweep, mscan, nmap, portsweep, saint, satan
R2L	ftp_write, guess_passwd, httptunnel, imap, multihop, named, phf, sendmail, snmpgetattack, snmpguess, spy, warezclient, warezmaster, worm, xclock, xsnoop
U2R	buffer_overflow, loadmodule, perl, ps, rootkit, sqlattack, xterm

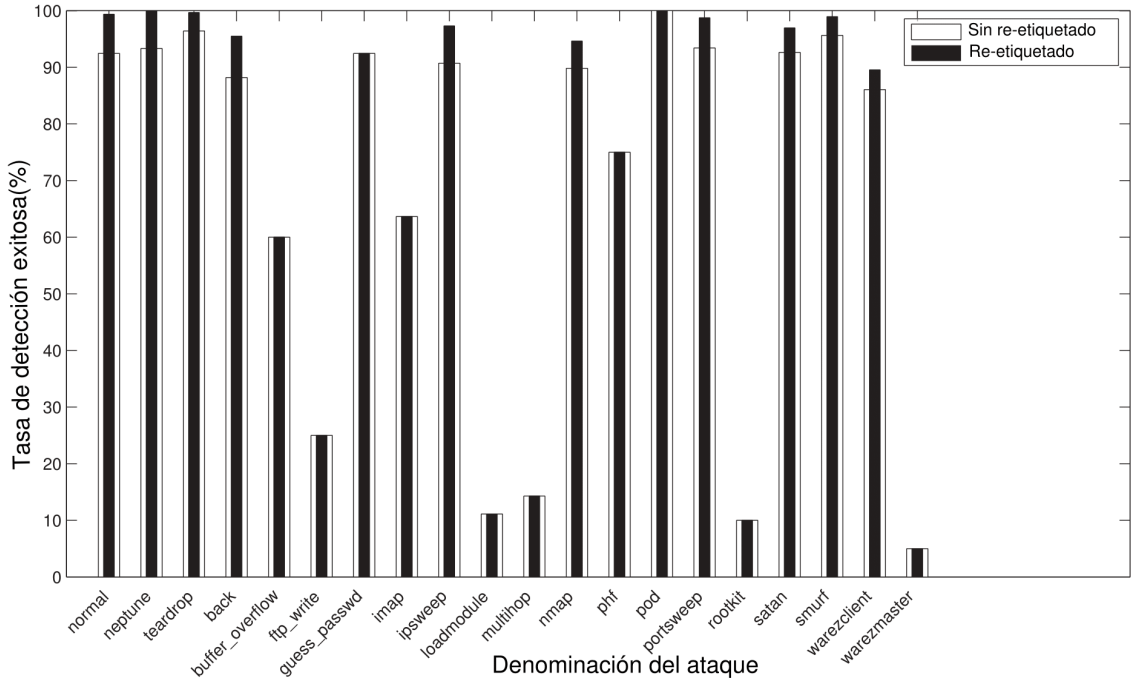

Figura 4.1. Tasa de detección con y sin reetiquetado de unidades

La tasa de detección de la Figura 4.1, refleja el porcentaje de anomalías correctamente identificadas (es decir, verdaderos positivos). Los falsos positivos (FP), verdaderos negativos (VN) y falsos negativos (FN) se indican por medio de las curvas ROC y los valores de la precisión (*accuracy*) que también proporcionamos en esta sección. Así, para mostrar la eficacia del proceso de re-etiquetado anteriormente descrito, las curvas ROC se dan en la Figura 4.2.

La Figura 4.2(a) corresponde a la curva ROC (tasa de falsos positivos) y la Figura 4.2(b) es el reflejo de la curva ROC (tasa de verdaderos negativos) obtenidos conlos patrones de prueba del *dataset* (no utilizadas durante el entrenamiento). En cuanto al rendimiento derivado de estas curvas, se ha calculado el área bajo la curva ROC (AUC). El uso del AUC hace la interpretación de los resultados de la curva de ROC más fácil. Por lo tanto, un clasificador perfecto proporcionará un AUC = 1,0 y un AUC = 0,5 correspondería a un clasificador aleatorio. En las curvas ROC, el punto de corte determina el mejor rendimiento que provee el clasificador. Los resultados visualizados en las figuras 4.2(a) y (b) se generaron a partir de un GHSOM entrenado con el conjunto total de características, 1384 neuronas, 5 capas.

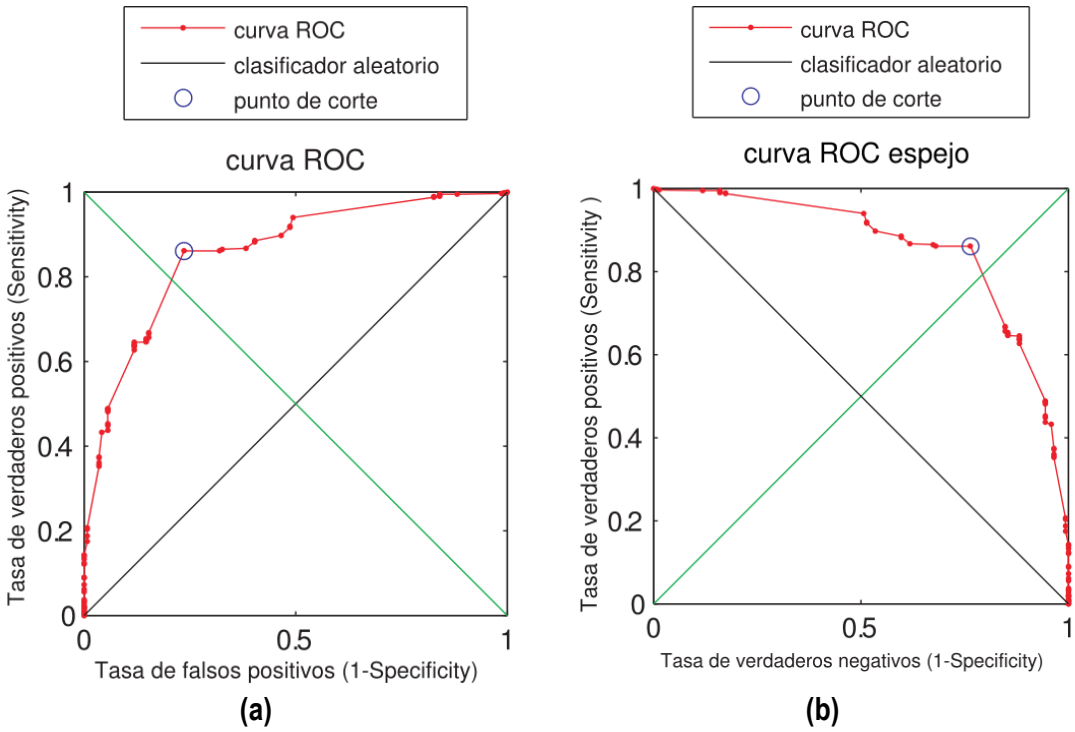

Figura 4.2. Curvas ROC para el proceso de reetiquetado

La Figura 4.3 presenta la tasa de detección según el tipo de ataque. Como puede verse, el método de reetiquetado aumenta el rendimiento de la clasificación en la mayoría de los casos, así como la tasa de detección. Vale la pena mencionar que el rendimiento es peor para U2R que para otros tipos de ataque, debido al menor número de patrones de entrenamiento U2R en el dataset NSL-KDD [275].

Para este caso también se utiliza un GHSOM de 1384 neuronas. Los resultados detallados sobre el tipo de ataque evaluado en la Figura 4.3, se resumen en la Tabla 4.3, donde se presentan la detección y tasas de falsos positivos. Como se muestra en esta tabla, el 99,8% de los patrones corresponde a patrones de tráfico normal y el 99,5% corresponde a patrones de diferentes ataques clasificados correctamente. Como se muestra en la Figura 4.3 se obtuvieron altas tasas de detección, aunque la mayoría de los ataques U2R no se clasificaron correctamente, ya que el dataset KDD está bastante desequilibrado con relación a este tipo de ataque. De hecho, las muestras U2R y R2L representan sólo el 0,001% y el 0,02% de las muestras de entrenamiento, respectivamente [268].

Tabla 4.3. Tasas de detección de tráfico Normal/Ataques

Conexión	Tasa de detección (%)	Tasa de falsos positivos (%)
Normal	99.8 ± 0.10	1.10 ± 0.92
Ataque	99.5 ± 0.33	4.33 ± 0.56

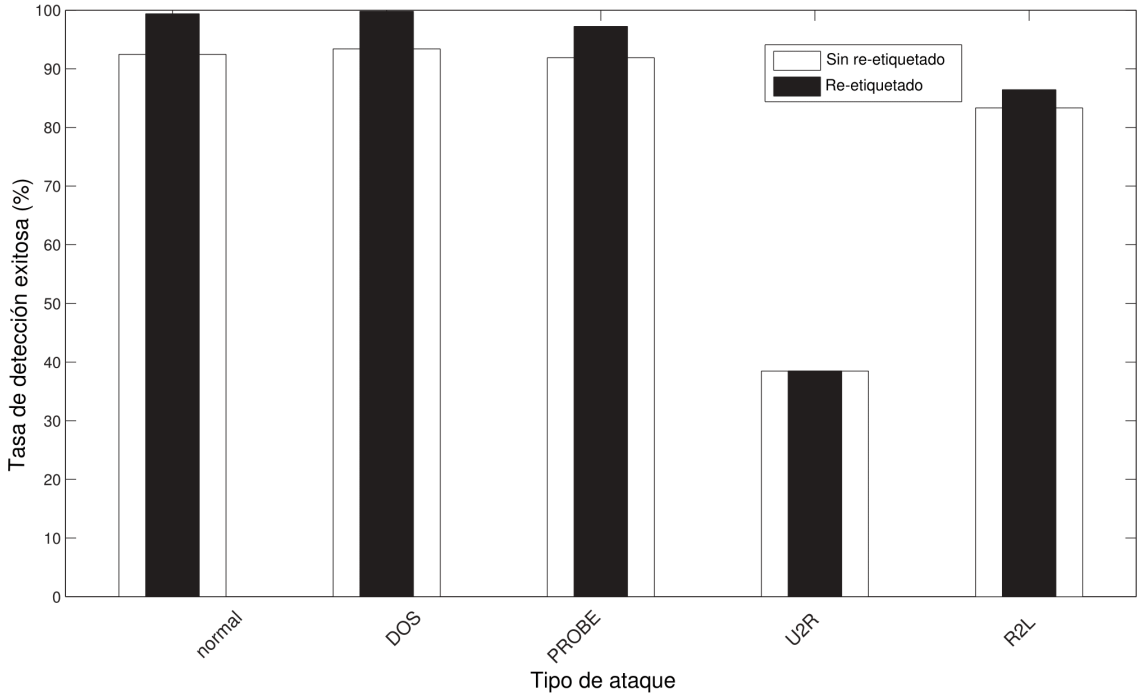

Figura 4.3. Tasa de detección para differentes tipos de ataque

Es importante resaltar que cuando se utiliza el conjunto total de características, consecuentemente, se requiere de un gran número de neuronas GHSOM para alcanzar tasas de detección más altas. La Figura 4.4 muestra el rendimiento proporcionado por el algoritmo de clasificación como una función del número de neuronas GHSOM. Al mismo tiempo, la Figura 4.4 muestra el incremento del rendimiento alcanzado por el proceso de reetiquetado anteriormente descrito. La probabilidad de unidades estructurales *muertas* aumenta a medida que crece la GHSOM. Este efecto se refleja en la exactitud de la clasificación como se muestra en la Figura 4.4. Sin embargo, las unidades que permanecen sin etiquetar durante el proceso de entrenamiento (es decir, una unidad muerta), pueden ser la BMU para nuevas instancias de datos. En este caso, el re-etiquetado despierta las unidades *muertas* en función de la etiqueta previamente asignada a las unidades vecinas durante el entrenamiento. Por otra parte, la Figura 4.4 muestra el efecto del método re-etiquetado y su ventaja cuando el número de unidades crece.

El GHSOM ha sido entrenado con el conjunto total de características y su máximo tamaño es 1384 unidades y cinco capas, tal y como se ha dicho anteriormente. Cuando se aplica el re-etiquetado, la tasa de detección depende menos de la cantidad de neuronas GHSOM. Por otra parte, el rendimiento del sistema sin re-etiquetado disminuye a medida que el número de neuronas GHSOM crece. Según esto, dado que el uso de conjuntos de características no optimizadas permite un GHSOM más amplio y más profundo, con más neuronas, el proceso de re-etiquetado tiene un efecto de desempeño positivo. Esta es precisamente la situación en los casos de ataques desconocidos, donde las caracteristicas no pueden ser seleccionadas de acuerdo a sus propiedades discriminativas.

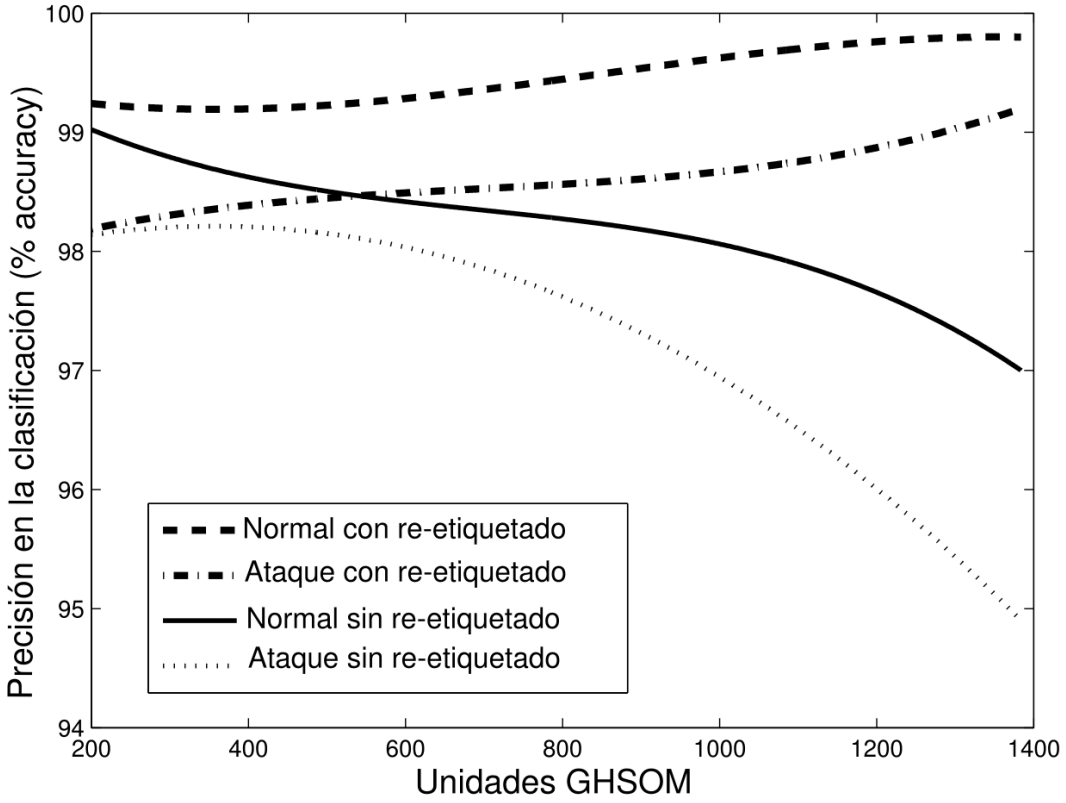

Figura 4.4. Tasa de detección con o sin reetiquetado utilizando el conjunto total de características

4.7 RESULTADOS EXPERIMENTALES CON CONJUNTOS DE CARACTERÍSTICAS REDUCIDAS

Como se ha indicado anteriormente, aunque GHSOM con re-etiquetado permite altas tasas de detección utilizando mapas grandes GHSOM, es necesario proporcionar tasas de detección mayores al 98%. Dado que el objetivo de los algoritmos propuestos en este trabajo es implementar el IDS/IPS en un módulo en tiempo real que pueda detectar comportamientos anómalos y decidir si bloquea o no una conexión todo el proceso tiene que ser lo suficientemente rápido, y para acelerar la detección, se hace necesario reducir la complejidad tanto de la selección de características como de la clasificación.

En consecuencia, la reducción de la dimensión del espacio de características es conveniente con el fin de optimizar tanto la tasa de clasificación como el tiempo de computacional, preservando al mismo tiempo el rendimiento del proceso de etiquetado. El entrenamiento del clasificador con un conjunto reducido de características podría mejorar el rendimiento, mientras que la determinación del GHSOM se hace más rápida y efectiva, si las características se seleccionan adecuadamente para discriminar entre los patrones de entrada. Así, puesto que el clasificador se entrena usando vectores con menos dimensiones, el proceso de entrenamiento, así como el cálculo de la BMU,

será más rápido. De hecho, la Figura 4.5 muestra que el tiempo de entrenamiento del GHSOM crece exponencialmente cuando aumenta el número de unidades.

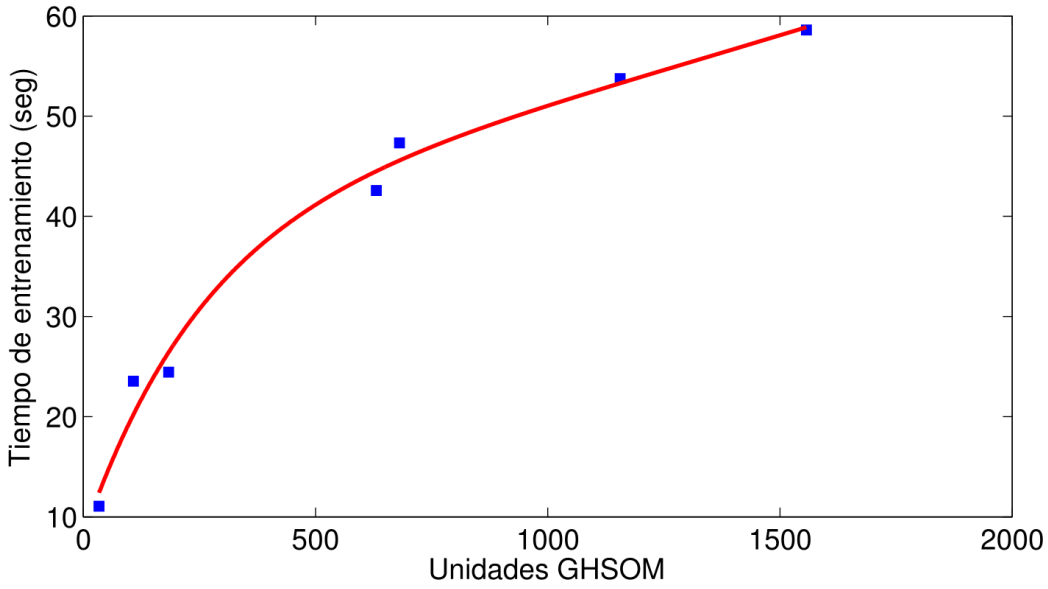

Figura 4.5. Tiempo de entrenamiento dependiendo del número de unidades GHSOMs

Sin embargo, encontrar un conjunto lo suficientemente reducido y discriminativo de características no es sencillo. En esta sección, se muestran los resultados obtenidos con conjuntos de características seleccionadas para cada tipo de ataque mediante el uso de un procedimiento propio, de optimización multi-objetivo, el cual ha sido anteriormente descrito en la sección 3.3. Cada conjunto de características se utiliza para entrenar un GHSOM y clasificar los patrones de prueba.

Es importante señalar que los *frentes de Pareto* en la Figura 4.6 han sido calculados utilizando los datos de entrenamiento, ya que las etiquetas son necesarias para calcular el *índice de Jaccard* utilizado en funciones *fitness*. Como los *frentes de Pareto* son de cinco dimensiones, en este caso se consideran cinco funciones objetivo en el proceso de optimización. La Figura 4.6 muestra cuatro proyecciones de los *frentes de Pareto* obtenidas en los planos correspondientes a cuatro pares de objetivos. Apartir del *frente de Pareto* obtenido mediante el procedimiento de optimización multi-objetivo, se pueden seleccionar varios conjuntos de características diferentes. En los experimentos mostrados a continuación, se han utilizado los conjuntos de características S1, S2, S3, S4 y S5, como se muestra en la figura 4.6.

Los resultados mostrados en la Figura 4.6 corresponden a un GHSOM entrenado con conjuntos de características no dominadas S1 (340 neuronas, cinco capas), S2 (308 neuronas, seis capas), S3 (340 neuronas, cinco capas), S4 (420 neuronas, cinco capas) y S5 (225 neuronas, cinco capas).

Los conjuntos no-dominados de características seleccionados, han sido utilizados para el entrenamiento del GHSOM. A continuación, se muestran los resultados de la clasificación obtenida cuando la GHSOM está entrenada y probada con los conjuntos de características optimizadas obtenidos a partir del *frente Pareto*, como se indica en la Figura 4.6.

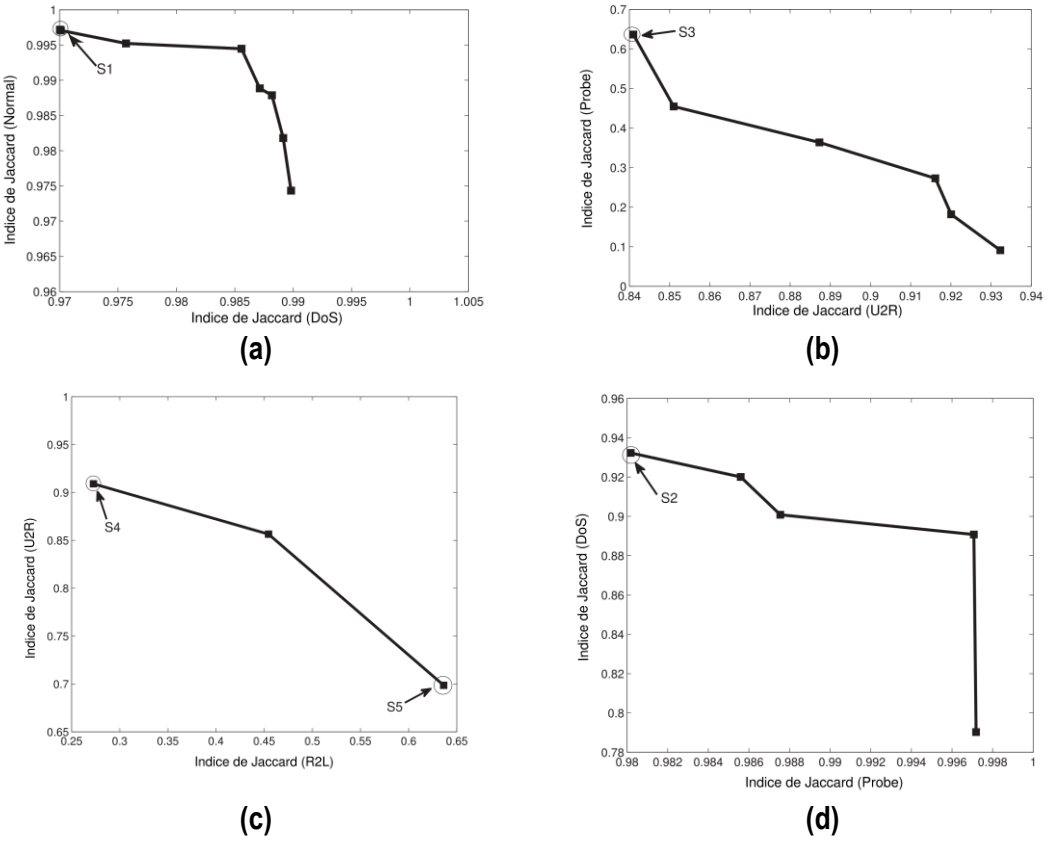

Figura 4.6. Frentes de Pareto para (a) Normal/DoS, (b) Probe/U2R, (c) U2R/R2L y (d) DoS/Probe

En la Figura 4.7, se muestran las tasas de detección obtenidas utilizando conjuntos de características no dominadas, seleccionadas mediante la optimización multiobjetivo. En cuanto a la detección de cada tipo de ataque, se aborda mediante distintos conjuntos de características y la estructura GHSOM desarrollada durante el entrenamiento es diferente. Como vemos en la Figura 4.7, conjuntos de características no-dominadas proporcionan valores de tasa de detección más altos para diferentes ataques que la alternativa que usa todas las características. Además, en el caso del ataque U2R, el conjunto de características S4, en la que el *índice de Jaccard* para U2R es máximo, proporciona una mayor precisión (*accuracy*) de clasificación que el conjunto completo de características. Es importante señalar que, mientras que los conjuntos de características no dominadas se han calculado a partir de las muestras de entrenamiento, los resultados de la clasificación que se muestra en la Figura 4.7 se obtienen con el *dataset* de prueba. El uso de todas las características proporciona altos niveles de precisión (*accuracy*) para las clases más probables. Este es el caso de las clases normal, DoS o Probe. Sin embargo, las características seleccionadas proporcionan claramente un mayor rendimiento para las clases menos probables, tales como U2R y R2L, como se puede comprobar en la Figura 4.7. Además, el propósito de la selección de características no es sólo aprovechar el rendimiento de la clasificación, sino también tener un conjunto reducido de características discriminantes que proporcionen valores de alta precisión (*accuracy*). Por lo tanto, el uso de un menor número de características reduce la carga computacional asociada con el entrenamiento y clasificación de nuevas muestras. En otras palabras, como las características seleccionadas proporcionan al menos la misma precisión

(*accuracy*) que el conjunto completo de características, nuestro método efectivamente descarta las características no informativas o redundantes para cada etiqueta de clase.

En la Figura 4.7 se muestran los resultados obtenidos para el *índice de Jaccard* al utilizar un GHSOM entrenado con todas las características (1384 neuronas y cinco capas), los conjuntos de características óptimos para tráfico normal (340 neuronas y cinco capas), ataques de DoS (308 neuronas y seis capas), ataques PROBE (340 neuronas y cinco capas), ataques U2R (420 neuronas y cinco capas) y ataques R2L (225 neuronas y cinco capas). En la Figura 4.8 se muestran los resultados de la precisión (*accuracy*) obtenidos al utilizar un GHSOM entrenado con las características anteriores (ver Figura 4.7).

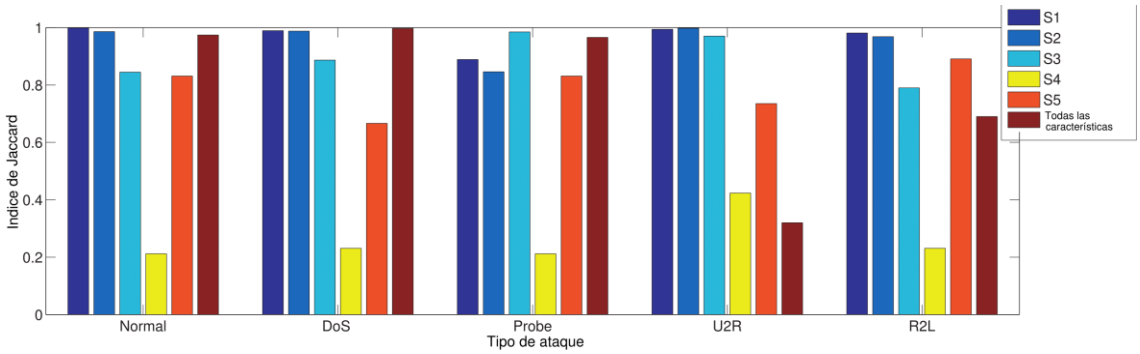

Figura 4.7. Tasa de detección utilizando conjuntos de características no dominadas seleccionadas desde optimización multi-objetivo

En la Figura 4.8, es evidente que el conjunto de características S4 proporciona mejores resultados de clasificación de ataques que el conjunto completo de características. Además, se requiere un menor número de unidades GHSOM; en consecuencia, los procesos de entrenamiento y de clasificación son menos costoso en términos de procesamiento computacional.

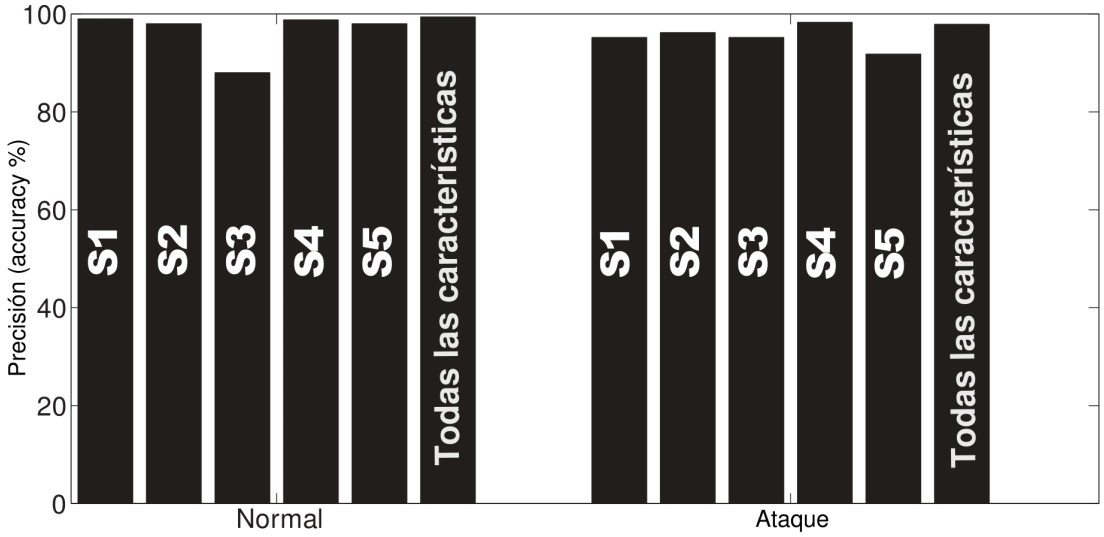

Figura 4.8. Comparación de tasas de detección para diferente número de PCs

NSGA-II utiliza individuos que codifican cadenas binarias que definen el subconjunto de características que deben seleccionarse en cada iteración. Se utilizan operadores de cruzamiento en un solo punto (*single-point*) y mutación a nivel de bit (*bitwise*) y la población inicial de NSGA-II consta de 30 individuos. Por otra parte, la probabilidad de cruzamiento es de p_c=0.9 y la probabilidad de mutación es de p_m=1/l donde l es la longitud de los genes, se utiliza (p_m=1/41). Estos valores dan buenos resultados en diferentes experimentos realizados en [245] en diferentes *dataset*. También proporcionaron una buena dispersión de la solución, así como una convergencia adecuada del algoritmo, como se muestra en la sección siguiente.

La Figura 4.9 muestra las curvas ROC para la clasificación de tráfico del tipo normal/ataque utilizando re-etiquetado y con el correspondiente conjunto de características no dominado. A partir de estas curvas, está claro que los conjuntos de características no-dominadas mejoran los resultados obtenidos con el conjunto completo de características en términos de sensibilidad y especificidad.

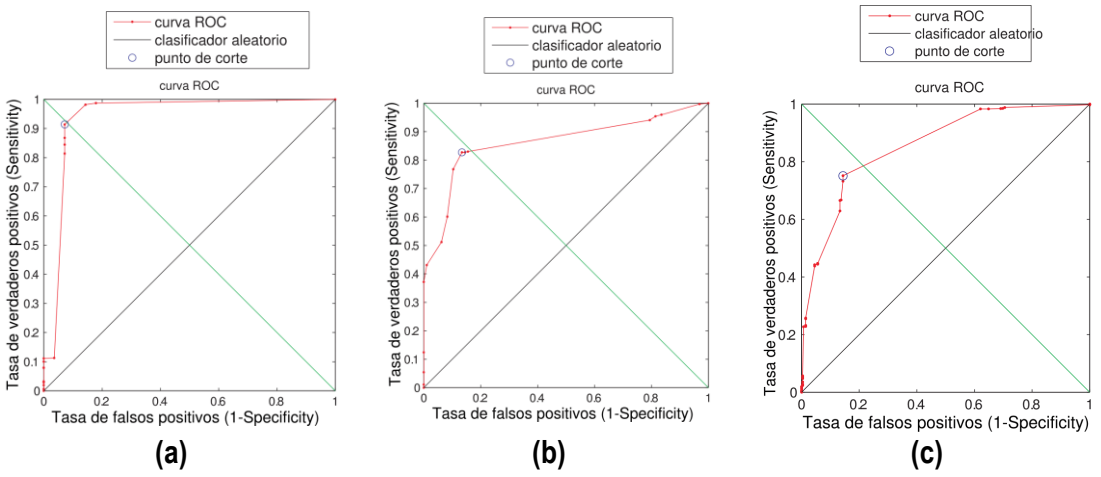

Figura 4.9.(a)(b)(c) Curvas ROC para el proceso de reetiquetado con el conjunto de características no dominado

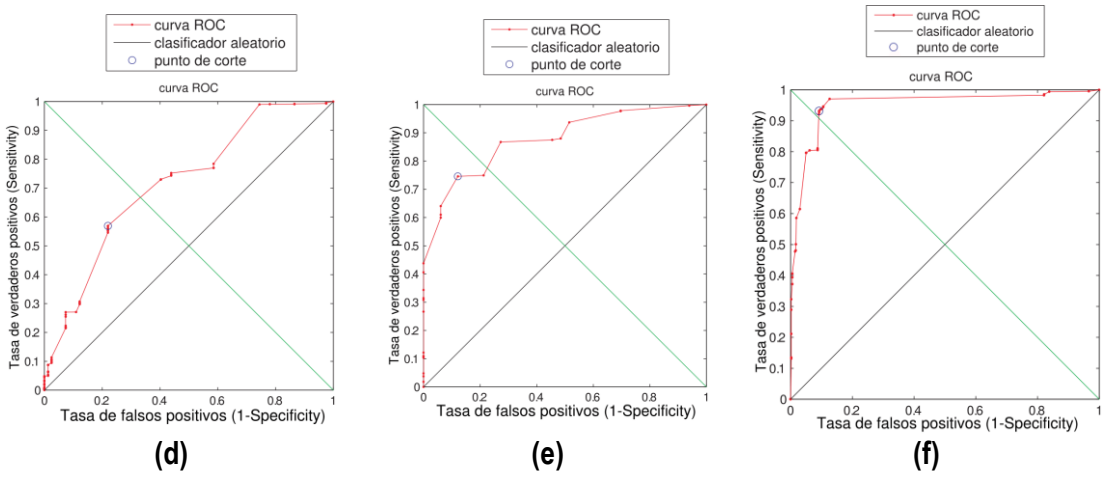

Figura 4.9. (d)(e)(f) Curvas ROC para el proceso de reetiquetado con el conjunto de

características no dominado

Los resultados obtenidos para el proceso de reetiquetado con el conjunto de características no dominado (que se visualizan en la Figura 4.9) son (a) S1 (340 neuronas y cinco capas), (b) S2 (308 neuronas y seis capas), (c) S3 (340 neuronas y cinco capas), (d) S4 (225 neuronas y cinco capas), (e) S5 (420 neuronas y cinco capas) como en las figuras anteriores y (f) PCA con 30 PCs (1654 neuronas y cinco capas).

La Tabla 4.4 proporciona la precisión (*accuracy*) del proceso de clasificación, y los diferentes conjuntos de características determinadas por el procedimiento de optimización multi-objetivo aquí propuesto. La última fila de la tabla 4.4 proporciona la precisión (accuracy) obtenida con el mejor conjunto de características determinados por el Análisis de Componentes Principales (PCA). Se logra una precisión (*accuracy*) de 99.12% para los cinco diferentes tipos de ataques aquí considerados cuando se utiliza el conjunto de características no dominado con un índice de Jaccard máximo para S2.

Tabla 4.4 Precisión (accuracy) de la clasificación para los diferentes conjuntos de características

Conjunto de Características	Número de Características	Precisión o Accuracy (%)	Falsos Positivos (%)
Todas	41	99.60 ±0.33	4.32 ±0.80
S1	22	98.12 ±1.10	3.10 ±0.76
S2	29	99.12 ±0.61	2.24 ±0.41
S3	25	98.27 ±1.19	2.13 ±0.59
S4	25	98.10 ±1.71	3.15 ±0.50
S5	29	97.18 ±1.43	4.16 ±0.52
PCA	20	82.1 ±5.13	6.50 ±1.00

En las secciones anteriores, se analizó el desempeño de GHSOM al detectar diferentes tipos de ataques. Sin embargo, la detección de tráfico normal y ataques con la máxima precisión (*accuracy*) es esencial para la aplicación práctica, es decir, la detección de alta precisión (*accuracy*) no es útil si no se detecta de manera similar a otros tipos de ataques.

4.8 EVALUACIÓN DE LAS PRESTACIONES DE NSGA-II

En esta sección se proporcionan los resultados de la convergencia de NSGA-II, que muestran el número de generaciones necesarias para proporcionar soluciones suficientemente buenas. Aunque la determinación de la convergencia en algoritmos multi-objetivo no es sencilla [279], en este trabajo, se ha utilizado la conocida métrica de hiper-volumen [280]. De esta manera, cuanto mayor es el hiper-volumen, mayor es el número de soluciones no dominadas, lo que indica un mejor rendimiento. En los algoritmos evolutivos, el tamaño de la población juega un papel importante, ya que está directamente relacionada con su diversidad. Poblaciones con baja diversidad pueden necesitar un mayor número de generaciones para evolucionar a soluciones aceptables. Por el contrario, en poblaciones con alta diversidad aumenta la complejidad y

finalmente el tiempo de procesamiento. Los experimentos que se han realizado ponen de manifiesto que las poblaciones de 30 individuos proporcionan los mejores resultados, como se muestra en la Figura 4.10, dado que proporcionan un buen compromiso (*trade-off*) entre el tiempo de procesamiento y la precisión (*accuracy*). Por otra parte, la Tabla 4.5 muestra la media del *índice de Jaccard* para todas las clases dado el enfoque multi-objetivo pretende maximizar para todas las clases de forma simultánea, así como el hiper-volumen para los correspondientes frentes no dominados. Como se muestra en la Tabla 4.5 y la Figura 4.10(a), las poblaciones de 30, 40 y 50 individuos (N) proporcionan el mismo desempeño teniendo en cuenta los valores de la desviación estándar. Sin embargo, la complejidad computacional es considerablemente mayor para N = 40 como se muestra en la Figura 4.10(b). Del mismo modo, mientras que N = 20 proporciona valores de hiper-volumen más bajos, los proporcionados por N=30, N=40 y N=50 no son significativamente diferentes, lo que indica niveles de convergencia similares.

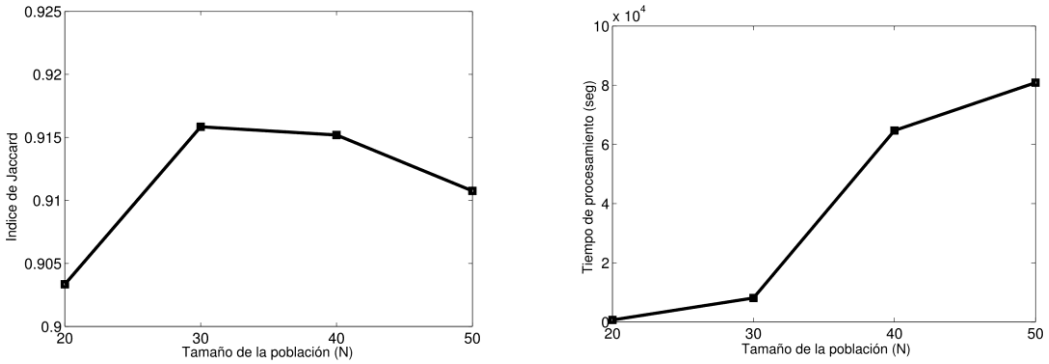

Figura 4.10. Promedio del índice de Jaccard (a) y tiempo de procesamiento para J = 0. 88 (b)

Los resultados mostrados en la Figura 4.10 corresponden a diferentes tamaños de población con los correspondientes valores de desviación estándar. Téngase en cuenta que los valores de desviación estándar no son visibles en (b) ya que son pequeños en comparación con los valores medios que proporciona la figura.

Tabla 4.5. Métrica de Hyper-volumen para NSGA-II

Tamaño de la población	Hiper-volumen	Índice de Jaccard
20	0.95 ± 0.35	0.880 ± 0.010
30	1.60 ± 0.15	0.920 ± 0.010
40	1.57 ± 0.12	0.910 ± 0.005
50	1.50 ± 0.12	0.910 ± 0.005

Los resultados visualizados en la Tabla 4.5 corresponden a ejecuciones con 50 generaciones.

4.9 EFECTIVIDAD COMPUTACIONAL

El objetivo del método de selección de características es calcular un conjunto de características lo suficientemente discriminativas para todas las clases, dentro de un límite de tiempo aceptable. La complejidad exponencial del tiempo de procesamiento es un problema que limita la utilidad de algunos enfoques para los problemas de clasificación por encima de una dimensión dada. En cualquier caso, en aplicaciones reales, es mejor tener un procedimiento que sea capaz de proporcionar una buena solución del problema en un tiempo aceptable que un procedimiento que obtiene una solución no tan buena, aunque muy rápidamente. Por otro lado, la cuestión es determinar las soluciones no dominadas correspondientes a selecciones de características de alta calidad a pesar de que esto podría requerir más tiempo de cálculo que la obtención de una selección de características no tan buena. Para completar la caracterización del procedimiento de selección *wrapper* aquí propuesto, se han obtenido datos acerca de sus requisitos de tiempo de computación, comparándolos con otros métodos de selección tipo *filter* y *wrapper*. Por lo tanto, la Tabla 4.6 muestra el tiempo de CPU requerido para seleccionar el mejor conjunto de características que proporciona cada técnica y el uso de GHSOM-pr como clasificador. Procedimiento que proponemos para etiquetar las unidades GHSOM que hacen posible un proceso de agrupamiento mediante comportamiento probabilístico (ya detallado en la sección 3.4). Por otra parte, el conjunto de entrenamiento se ha utilizado para determinar el número de características que proporcionan los mejores resultados de la clasificación en términos del *índice de Jaccard*.

Vale la pena señalar que se evaluó el tiempo de procesamiento necesario para calcular las soluciones aceptables en términos de la media del *índice Jaccard* calculado para todas las clases, ya que representa el rendimiento de la clasificación multiclase. La Figura 4.10 muestra que una población de 30 individuos puede proporcionar un buen equilibrio entre rendimiento y complejidad, y que aumentar el número de individuos (es decir, N=40) implica un tiempo de procesamiento considerablemente más alto.

Tabla 4.6. Complejidad computacional y precisión (accuracy) para diferentes métodos de selección de características

Método de Selección de Características	Tiempo de Procesamiento (s)	Índice de Jaccard
Métodos basados en filtros		
PCA	796 ± 20	0.83 ± 0.04
FDR	597 ± 3	0.74 ± 0.03
ReliefF	32112 ± 32	0.92 ± 0.04
Métodos basados en envoltorios		
BackwardFS	17194 ± 121	0.82 ± 0.07
Selección de características multiobjetivo	8084 ± 180	0.92 ± 0.01

4.10 SIGNIFICACIÓN ESTADÍSTICA

Debido a la variabilidad impuesta tanto por el proceso de inicialización pseudo-aleatorio del GHSOM como por el algoritmo evolutivo utilizado para la optimización multi-objetivo, los resultados

de la clasificación pueden variar entre las diferentes ejecuciones. De hecho, ya en la Tabla 4.4, se han mostrado, los resultados medios correspondientes a la reducción de características mediante PCA y al método de optimización multi-objetivo. Se proporcionan con la desviación estándar calculada sobre 50 ejecuciones del algoritmo de entrenamiento/clasificación. En cualquier caso, las pruebas de significación estadística son necesarias con el fin de analizar hasta que punto los valores medios obtenidos mediante diferentes experimentos son estadísticamente diferentes. Esto se ha abordado, a través de la prueba de la hipótesis que supone que los resultados de los diferentes experimentos se han extraído de una distribución con el mismo promedio (hipótesis nula). Mientras que ANOVA [215] se utiliza para encontrar diferencias significativas entre las medias de grupo, mediante pruebas de comparación múltiple se trata de identificar los grupos específicos cuyos medios son significativamente diferentes.

Se ha aplicado ANOVA a los resultados obtenidos utilizando la optimización multi-objetivo para reducir la dimensión del espacio de características. Estos resultados se presentan en la Figura 4.11. El p-valor que ha proporcionado el análisis de ANOVA ($p < 10^{-7}$) indica que todos las medias son significativamente diferentes. Sin embargo, una prueba de múltiples comparaciones revela que los resultados obtenidos utilizando conjuntos de características no dominadas S1, S2, S3, S4 y S5 no son estadísticamente diferentes para la detección de tráfico normal/ataque, proporcionando la misma precisión (*accuracy*).

En la Figura 4.11(a) se visualiza la prueba ANOVA para los resultados experimentales utiliza la optimización multi-objetivo y en la figura 4.11(b) se visualizan las pruebas producto de las comparaciones múltiples para identificar el origen de las diferencias en las medias.

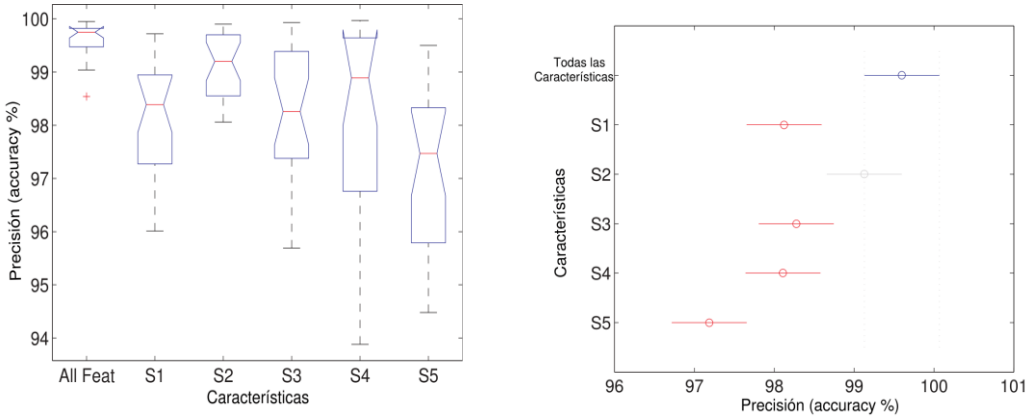

Figura 4.11. Prueba ANOVA (a) y pruebas de comparaciones múltiples (b)

4.11 CONCLUSIONES

A partir de los distintos experimentos realizados con el *dataset* completo, se ha llegado a las siguientes conclusiones:

- El proceso de re-etiquetado probabilístico, aquí propuesto, aumenta la tasa de detección de la mayoría de los ataques (anomalías correctamente identificadas, es decir, verdaderos

positivos). Obteniendo una mejora promedio del 3,5% sobre todos los ataques, y del 5,5% en los ataques en los que el re-etiquetado produce mejoras.

- En cuanto a la tasa de detección según el tipo de ataque (tráfico normal, ataque DoS, ataque Probe, ataque U2R y ataque R2L), el método de re-etiquetado propuesto aumenta el rendimiento de la clasificación y la tasa de detección, en la mayoría de los casos. Exceptuando la categoría de ataque U2R, dado que este tipo de ataque tiene un reducido número de patrones de entrenamiento (las muestras U2R representan sólo el 0,001% de las muestras de entrenamiento) en el *dataset* NSL-KDD.

- El rendimiento del sistema sin re-etiquetado disminuye a medida que el número de neuronas GHSOM crece. Cuando se aplica el re-etiquetado, mediante el método aquí propuesto, la tasa de detección depende menos de la cantidad de neuronas en el GHSOM.

Dado que se pretende conseguir un IDS/IPS que funcione en tiempo real y pueda detectar comportamientos anómalos y decidir si bloquea o no una conexión, es necesario reducir la complejidad tanto de la selección de características como de la clasificación. Por ello se ha intentado reducir la dimensión del espacio de características optimizando tanto la tasa de clasificación como el tiempo de cómputo y preservar el rendimiento del proceso de etiquetado. Así el clasificador ha sido entrenado usando vectores con menos dimensiones, permitiendo un cálculo más rápido de las BMU. Por lo tanto, el uso de un menor número de características reduce la carga computacional asociada con el entrenamiento y clasificación de nuevas muestras.

A partir de los experimentos realizados con un conjunto reducido de características, se obtuvo una precisión (accuracy) de 99.12% para los cinco diferentes tipos de ataques aquí considerados, cuando se utiliza el conjunto de características no dominado (29 características) con un índice de Jaccard máximo. Este resultado pese a no ser mejor que la precisión (99.60%) obtenida con el uso de todas las características, representa una interesante alternativa, dado que utiliza mucho menos características y por tanto el tiempo de procesamiento computacional disminuye.

De forma complementaria, mediante un procedimiento de optimización multi-objetivo basado en el algoritmo NSGA-II se obtuvo el *frente de Pareto* del conjunto de características, a partir del cual se calculó la tasa de detección de tráfico normal/ataque. Para la convergencia del algoritmo NSGA-II se utilizó la métrica hiper-volumen (cuanto mayor es, mejor es el rendimiento del algoritmo multiobjetivo). Por otra parte, en algoritmos evolutivos el tamaño de la población está directamente relacionada con su diversidad, de forma que poblaciones con baja diversidad pueden necesitar un mayor número de generaciones para evolucionar a soluciones aceptables, en contraposición, poblaciones con alta diversidad aumenta la complejidad y finalmente el tiempo de procesamiento. En nuestro caso hemos comprobado que poblaciones de 30 individuos proporcionan resultados aceptables.

Los métodos que se muestran en la Tabla 4.7 se han clasificado en (1) métodos que no utilizan selección de características, (2) métodos basados en *filter*, y (3) métodos basados en *wrapper*. En la Tabla 4.7, se muestra la comparación con otros métodos existentes que utilizan el dataset NSL-KDD. Los valores de la desviación estándar se muestran siempre que estén disponibles.

La Tabla 4.7, pone de manifiesto que el enfoque propuesto alcanza una alta tasa de ataques detectados en el rango de los enfoques existentes que mejor prestaciones proporciona. No

obstante, hay que señalar que con los resultados de la Tabla 4.7 sólo damos una idea del rendimiento de nuestro procedimiento en comparación con los resultados proporcionados por otros autores. Aunque, en los casos que se muestran en la Tabla 4.7, nuestra propuesta supera a otros procedimientos anteriores, no pretendemos indicar que nuestro método es mejor que los otros en todos los casos y problemas de clasificación. Tal comparación exhaustiva no es posible porque los únicos resultados de rendimiento disponibles se refieren a la tasa de detección, la desviación estándar no está aún disponible en todos los casos, y no tenemos las implementaciones específicas de los diferentes procedimientos para llevar a cabo el conjunto requerido de ejecuciones.

Tabla 4.7. Comparación de la precisión (accuracy) de la detección para diferentes métodos de clasificación

Implementación en IDS	Características usadas para la clasificación	Exactitud de la detección (Accuracy %)	Falsos positivos (%)
Sin selección de características			
Naïve Bayes [291]	41	76.56	No proporcionado
Random forest [291]	41	80.67	No proporcionado
Árboles de decisión (J48) [291]	41	81.05	No proporcionado
AdaBoost [273]	41	90.31	3.38
GHSOM-pr	41	99.59 ± 2.25	4.32 ± 1.93
GHSOM [292]	41	96.02	4.92
A-GHSOM [250]	41	96.63	1.80
Kayacik y otros [268]	41	90.40	1.38
Métodos basados en *filter*			
Naïve Bayes + N2B [273]	41	96.50	3.00
PCA	30	82.1 ± 5.13	6.50 ± 1.00
FDR + kernel PCA [116]	23	90.0	8.00
Métodos basados en *wrapper*			
Basados en Árboles de decisión [293]	16	98.38 ± 1.62	No proporcionado
GHSOM + Selección de caract. multiobjetivo (S4)	25	99.12 ± 0.61	2.24 ± 0.41

CAPÍTULO 4.
ESTUDIO EXPERIMENTAL APLICADO A LA DETECCIÓN DE INTRUSOS

CAPÍTULO 5. CONCLUSIONES Y PRINCIPALES APORTACIONES

En este capítulo se describen las conclusiones y aportaciones a las cuales se llega producto del análisis de los resultados de los experimentos descritos en el capítulo anterior. El capítulo finaliza con la identificación de algunos trabajos futuros, coherentes con la propuesta aquí enunciada.

5.1 CONCLUSIONES

En este trabajo se presenta un enfoque de detección de intrusos que se aprovecha de las propiedades discriminantes de los mapas auto-organizativos. Más específicamente, se ha considerado GHSOM donde hemos introducido un procedimiento de reetiquetado que mejora el rendimiento de la detección y prevención de intrusiones. Este procedimiento de agrupamiento permite un mejor etiquetado de los datos de entrada, teniendo en cuenta los grupos (*cluster*) encontrados por el GHSOM previamente entrenado. También incluye un procedimiento multi-objetivo basado en el algoritmo NSGA-II para la selección de características con el fin de reducir la complejidad de la GHSOM y para mejorar el rendimiento de clasificación. Los resultados obtenidos superan los proporcionados por todos los enfoques considerados, excepto los de tasas de falsos positivos que se muestra en [250] en este caso, sólo hay un 0,44% de diferencia en la tasa de falsos positivos.

En cuanto a los *dataset* utilizados en procesos de selección, se identificó que existen muchas entradas de datos redundantes en KDD'99 y por lo tanto las técnicas de aprendizaje automático están sesgadas hacia la mayoría de los eventos que ocurren. Esta propiedad conduce a los algoritmos a ignorar eventos menos frecuentes, que pueden ser más perjudiciales que la mayoría de los eventos que ocurren. Los falsos positivos son otro inconveniente importante en el *dataset* KDD'99. Por ello en este trabajo se adoptó el *dataset* NSL-KDD (una versión mejorada de KDD'99 conjunto de datos). El enfoque que se presenta en este trabajo lleva a una selección de las características de acuerdo a su relevancia y por ello se beneficia la comprensión de las funciones de red y sus influencias a ataques.

5.2 APORTACIONES Y RESULTADOS

- Una técnica para la clasificación de ataques basa en mapas GHSOM y el método de re-etiquetado propuesto, con el que se alcanza una tasa de detección de 99,4% para los patrones normales, y 99,2% para las conexiones de ataque.
- Un enfoque para seleccionar el conjunto más adecuado de características en lugar de utilizar todo el conjunto de características, utilizando el índice de Jaccard para cada tipo de ataque, evaluado después del entrenamiento del GHSOM.
- Se ha implantado un procedimiento multi-objetivo basado en el NSGA-II, en nuestros experimentos, hemos considerado cinco soluciones no dominantes diferentes (conjuntos de características seleccionadas) que pertenecen al *frente de Pareto* obtenido. Estas cinco soluciones se han elegido de tal manera que optimizan el *índice de Jaccard* para las situaciones de tráfico normal y uno de los tipos de ataque (Normal, DoS, Probe, U2R, y R2L). Los resultados obtenidos en el KDD-NSL muestran que el conjunto de características que maximiza el *índice de Jaccard* para el ataque U2R, junto con el procedimiento de re-etiquetado aplicado en el procedimiento de entrenamiento del GHSOM, proporciona tasas de detección de hasta un 99,8% para el tráfico normal, y hasta 99,4 % para el tráfico anómalo con cinco niveles y 225 neuronas.

- La precisión (accuracy) de la detección alcanzada por el procedimiento propuesto es 99,12%, mejorando así los resultados obtenidos por otros procedimientos propuestos considerados en la tabla 4.7.
- Producto de la investigación que sustenta esta tesis se efectuaron cuatro (4) publicaciones, como autor principal, de las cuales tres (3) fueron en revistas indexadas (Knowledge-Based Systems - Q1, Lecture Notes in Computer Science – Q2, y JATIT – Q4). El detalle de las publicaciones puede ser consultado en la presentación de esta tesis.
- Se participó como coautor en cuatro (4) publicaciones, de las cuales tres (3) fueron en revistas indexadas (Neurocomputing – Q1, Lecture Notes in Computer Science – Q2, y JATIT – Q4). El detalle de las publicaciones puede ser consultado en la presentación de esta tesis.

5.3 TRABAJOS FUTUROS

En trabajos futuros, tenemos la intención de analizar cómo el GHSOM actual podría mejorarse mediante hibridación con otras técnicas de clustering mejoradas, tales como el Modelo de Mezcla Gaussiana [252] o el uso de Máquinas de Soporte Vectorial (SVM). Por otro lado, también se analizarán alternativas para implementar eficientemente los sistemas de prevención de intrusos mediante el uso de módulos del núcleo optimizadas en ordenadores con varios procesadores y/o tarjetas de interfaz de red programable (por ejemplo, incluyendo procesadores de red). Consideramos que aprovechando el paralelismo presente en los nodos de procesamiento actuales mejorará el rendimiento del IPS, permitiendo así que los sistemas de prevención de intrusión actúen más eficientemente.

REFERENCIAS

[1] J. Mieres, «Ataques Informáticos - Debilidades de seguridad comúnmente explotadas,» *Evil fingers,* pp. 1-17, 2009.

[2] A. Lazarevic, J. Srivastava y V. Kumar, A survey of intrusion detection techniques. Book "Managing Cyber Threats: Issues, approaches and challenges", Kluwer in spring, 2004.

[3] K. Kendall, *A Database of Computer Attacks for the Evaluation of Intrusion Detection Systems,* 1998.

[4] J. D. Howard, *An analysis of security incidents on the internet. PhD dissertation,* Pittsburgh: Carnegie Mellon University, 1997.

[5] J. Christy, *Cyber threat & legal issues,* Dahlgren, 1999.

[6] N. Freed, *Behavior of and Requirements for Internet Firewalls,* Network Working Group, 2000.

[7] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach y T. Berners-Lee, *Hypertext Transfer Protocol -- HTTP/1.1,* Network Working Group, 1999.

[8] CISCO, «CISCO - América Latina,» Abril 2015. [En línea]. Available: http://www.cisco.com/web/LA/soluciones/la/vpn/index.html.

[9] E. Rosen y Y. Rekhter, *BGP/MPLS IP Virtual Private Networks (VPNs),* Network Working Group, 2006.

[10] A. Melnikov, *IMAP4 Access Control List (ACL) Extension,* Network Working Group, 2005.

[11] L. Hung-Jen, R. L. Chun-Hung, L. Ying-Chih y T. Kuang-Yuan, «Intrusion detection system: A comprehensive review,» *Journal of Network and Computer Applications,* pp. 16-24, 2013.

[12] R. Bace y P. Mell, «Intrusion Detection Systems,» National Institute of Standards and Technology (NIST), 2001.

[13] P. Stavroulakis y M. Stamp, Handbook of information and communication security, New York: Springer-Verlag, 2010.

[14] ISI - GOST, «Information Sciences Institute - Global Operating Systems Technology Group,» 10 Septiembre 1999. [En línea]. Available: http://gost.isi.edu/cidf/.

[15] C. Kahn, D. Bolinger y D. Schnackenberg, *Communication in the Common Intrusion Detection Framework v 0.7,* CIDF Working Group, 1998.

[16] AusCERT, «Australian Computer Emergency Response Team,» 20 Abril 2015. [En línea]. Available: https://www.auscert.org.au/.

[17] IETF, «Intrusion Detection Exchange Format (IDWG),» 20 Abril 2015. [En línea]. Available: http://datatracker.ietf.org/wg/idwg/documents/.

[18] IETF - Network Working Group, «The TUNNEL Profile - RFC 3620,» Octubre 2003. [En línea]. Available: http://datatracker.ietf.org/doc/rfc3620/.

[19] IETF - Network Working Group, «The Intrusion Detection Message Exchange Format (IDMEF) - RFC 4765,» Marzo 2007. [En línea]. Available: http://datatracker.ietf.org/doc/rfc4765/.

[20] M. Wood y M. Erlinger, «Intrusion Detection Message Exchange Requirements - RFC 4766,» Marzo 2007. [En línea]. Available: http://datatracker.ietf.org/doc/rfc4766/.

[21] B. Feinstein y G. Matthews, «The Intrusion Detection Exchange Protocol (IDXP) - RFC 4767,» Marzo 2007. [En línea]. Available: http://datatracker.ietf.org/doc/rfc4767/.

[22] MITRE, «Common Vulnerabilities and Exposures,» Abril 2015. [En línea]. Available: http://cve.mitre.org/about/index.html.

[23] B. Bazara y C. H. Anthony, «Chapter 10. Intrusion Detection Systems.,» de *Handbook of Information and Communication Security*, New York, Springer-Verlag, 2010, pp. 193-215.

[24] P. García-Teodoro, J. Díaz-Verdejo, G. Maciá-Fernández y E. Vázquez, «Anomaly-based network intrusion detection: Techniques, systems and challenges,» *Computers & Security,* pp. 18-28, 2009.

[25] Y. Liao y V. R. Vemuri, «Use of K-Nearest Neighbor classifier for intrusion detection,» *Computers & Security,* pp. 439-448, 2002.

[26] S. X. Wu y W. Banzhaf, «The use of computational intelligence in intrusion detection systems: A review,» *Applied Soft Computing,* pp. 1-35, 2010.

[27] I. Corona, G. Giacinto y F. Roli, «Adversarial attacks against intrusion detection systems: Taxonomy, solutions and open issues,» *Inf ormation Sciences,* pp. 201-225, 2013.

[28] C. Guo, Y.-J. Zhou, Y. Ping, S.-S. Luo, Y.-P. Lai y Z.-K. Zhang, «Efficient intrusion detection using representative instances,» *Computers & Security,* pp. 255-267, 2013.

[29] H. A. Madni Uppal, M. Javed y M. Arshad, «An Overview of Intrusion Detection System (IDS) along with its Commonly Used Techniques and Classifications,» *International Journal of Computer Science and Telecommunications,* pp. 20-24, 2014.

[30] K. Scarfone y P. Mell, «Guide to Intrusion Detection and Prevention Systems (IDPS),» de *Special Publication 800-94*, Gaithersburg, 2007.

[31] M. Xie, S. Han, B. Tian y S. Parvin, «Anomaly detectioninwirelesssensornetworks:Asurvey,» *Journal ofNetworkandComputerApplications,* pp. 1302-1325, 2011.

[32] M. Tavallaee, E. Bagheri, W. Lu y A. Ghorbani, «A Detailed Analysis of the KDD CUP 99 dataset,» de *IEEE Symposium on Computational Intelligence for Security and Defense Applications - CISDA*, 2009.

[33] SNORT, «Snort,» 2015. [En línea]. Available: https://www.snort.org/.

[34] Check Point Software Technologies Ltd, «Checkpoint,» 2015. [En línea]. Available: http://www.checkpoint.com/corporate/nfr/index.html.

[35] L. Heberlein, «Network Security Monitor - Final Report,» Lawrence Livermore National Laboratory (LLNL) and the University of California - Davis (UCD), California, 2015.

[36] CISCO Security, «Cisco Intrusion Prevention System,» 2015. [En línea]. Available: http://www.cisco.com/c/en/us/support/security/intrusion-prevention-system/tsd-products-support-series-home.html.

[37] IBM, «IBM Software support lifecycle,» 2015. [En línea]. Available: http://www-01.ibm.com/software/support/lifecycleapp/PLCDetail.wss?synkey=L708592I24274G03-H021998L72016P05-Z585552G00815B27.

[38] M. Shyu, S. Chen, K. Sarinnapakorn y L. Chang, «A novel anomaly detection scheme based on principal component classifier,» de *Proceedings of the IEEE foundations and new directions of data mining workshop, in conjunction with the Third IEEE International Conference on Data Mining (ICDM03)*, 2003.

[39] Lazarevic, A; Kumar, V; Srivastava, J, «Intrusion detection a survey: managing cyber threats, issues, approaches, and challenges,» *Springer Verlag,* p. 330, 2005.

[40] D. Denning y P. Neumann, *Requirements and model for IDES – a real-time intrusion detection system,* Computer Science Laboratory, SRI International, 1985.

[41] N. Ye, S. Emran, Q. Chen y S. Vilbert, «Multivariate statistical analysis of audit trails for host-based intrusion detection,» *IEEE Transactions on Computers,* 2002.

[42] K. Surrey y R. Pyke, «Detecting Hackers (Analyzing Network Traffic) by Poisson Model Measure,» 2015. [En línea]. Available: http://www2.ensc.sfu.ca/people/grad/pwangf/IPSW_report.pdf.

[43] D. Anderson, T. Lunt, H. Javitz, A. Tamaru y A. Valdes, «Detecting unusual program behaviour using the statistical component of the next-generation intrusion detection expert system (NIDES),» Computer Science Laboratory, SRI International, Menlo Park, CA, USA, 1995.

[44] D. Heckerman, *A tutorial on learning with Bayesian networks,* Microsoft Research, 1995.

[45] C. Kruegel, D. Mutz, W. Robertson y F. Valeur, «Bayesian event classification for intrusion detection,» de *Proceedings of the 19th Annual Computer Security Applications Conference*, 2003.

[46] D. Yeung y Y. Ding, «Host-based intrusion detection using dynamic and static behavioral models,» *Pattern Recognition,* pp. 229-243, 2003.

[47] M. Mahoney y P. Chan, «Learning nonstationary models of normal network traffic for detecting novel attacks,» de *Proceedings of the Eighth ACM SIGKDD*, 2002.

[48] J. Estévez-Tapiador, P. García-Teodoro y J. Díaz-Verdejo, «Detection of web-based attacks through Markovian protocol parsing,» de *Proc. ISCC05*, 2005.

[49] K. Fox, R. Henning, J. Reed y R. Simonian, «A neural network approach towards intrusion detection,» de *13th National Computer Security Conference*, 1990.

[50] H. Debar, M. Becker y D. Siboni, «A neural network component for an intrusion detection system,» de *IEEE Symposium on Research in Computer Security and Privacy*, 1992.

[51] A. Cansian, E. Moreira, A. Carvalho y J. Bonifacio, «Network intrusion detection using neural networks,» de *International Conference on Computational Intelligence and Multimedia Applications (ICCMA'97)*, 1997.

[52] M. Ramadas, S. Ostermann y B. Tjaden, «Detecting anomalous network traffic with self-organizing maps,» de *Recent advances in intrusion detection, RAID*, 2003.

[53] J. Dickerson, «Fuzzy network profiling for intrusion detection,» de *Proceedings of the 19th International Conference of the North American Fuzzy Information Processing Society (NAFIPS)*, 2000.

[54] S. Bridges y R. Vaughn, «Fuzzy data mining and genetic algorithms applied to intrusion detection,» de *Proceedings of the National Information Systems Security Conference*, 2000.

[55] W. Li, «Using genetic algorithm for network intrusion detection,» *C.S.G. Department of Energy,* pp. 1-8, 2004.

[56] L. Portnoy, E. Eskin y S. Stolfo, «Intrusion detection with unlabeled data using clustering,» de *Proceedings of The ACM Workshop on Data Mining Applied to Security*, 2001.

[57] V. Barnett y T. Lewis, «Outliers in statistical data,» *Wiley,* 1994.

[58] K. Sequeira y M. Zaki, «ADMIT: anomaly-based data mining for intrusions,» *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining,* pp. 386-395, 2002.

[59] S. Noel, D. Wijesekera y C. Youman, «Modern Intrusion Detection, Data Mining and Degrees of attack guilt. In applications of data mining in computer security,» *Kluwer Academic Publisher,* 2002.

[60] K. Scarfone y P. Mell, «Intrusion Detection and Prevention Systems,» de *Handbook of Information and Communication Security*, Springer, 2010, pp. 177-192.

[61] IEEE Standards Association, «IEEE 802.11™: Wireless LANs,» 2015. [En línea]. Available: http://standards.ieee.org/about/get/802/802.11.html.

[62] PCI Security Standards Council, «Information Supplement: PCI DSS Wireless Guideline,» Julio 2009. [En línea]. Available: https://www.pcisecuritystandards.org/pdfs/PCI_DSS_Wireless_Guidelines.pdf.

[63] D. Marchette, «Computer Intrusion Detection and Network Monitoring: A Statistical Viewpoint,» *Springer,* 2001.

[64] Prelude, «Prelude A CS Product,» 2015. [En línea]. Available: https://www.prelude-ids.com/index.php/uk/.

[65] A. Lazarevic, V. Kumar y J. Srivastava, «Intrusion Detection: A survey,» de *Managing Cyber Threats*, Minnesota, Springer, 2005, pp. 19-78.

[66] A. Ghorbani, W. Lu y M. Tavallaee, «Evaluation Criteria. Network Intrusion Detection and Prevention. Concepts and Techniques. Advances in Information Security,» *Springer US,* pp. 161-183, 2010.

[67] J. Winkler, «A Unix Prototype for Intrusion and Anomaly Detection in Secure Networks,» de *Proceedings of the 13th National Computer Security Conference*, Baltimore, 1990.

[68] J. Winkler y L. Landry, «Intrusion adn Anomaly Detection, ISOA Update,» de *Proceedings of the 15th National Computer Security Conference*, Baltimore, 1992.

[69] SRI International, «Computer Science Laboratory,» 28 February 1992. [En línea]. Available: http://www.csl.sri.com/papers/9sri/9sri.pdf.

[70] E. Spafford y D. Zamboni, «Intrusion Detection Usign Autonomous Agents,» *Computer Network,* vol. 34, pp. 547-570, 2000.

[71] ArcSight, «Enterprise Security Management Software,» 1999. [En línea]. Available: http://www.arcsight.com/.

[72] NetForensics , «Security Information Management,» 1998. [En línea]. Available: http://www.nai.com/products/security/cybercopsvr/index.asp .

[73] S. Snapp, G. Dias, T. Goan, T. Heberlein, J. Brentano, C. Ho, B. Levitt, S. Mukherjee, T. Smaha, D. Grance y D. Mansur, «DIDS (Distributed Intrusion Detection System) Motivation, Architecture and an Early Prototype,» de *Proceedings of the 14th National Computer Security Conference*, Washington D.C., 1991.

[74] W. Jansen y P. Mell, «Mobile Agents in Intrusion Detection and Response,» de *Proceedings of the 12th Annual Canadian Information Technology Security Symposium*, Ottawa - Canada, 2000.

[75] P. Chan y V. Wei, «Preemptive Distributed Intrusion Detection Using Mobile Agents,» de *Proceedings of the Eleventh IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises - WET ICE*, Pittsburgh, 2002.

[76] C. Kruegel y T. Toth, «Distributed Pattern Detection For Intrusion Detection,» de *Proceedings of the Network and Distributed System Security Symposium Conference*, Los Angeles - CA, 2002.

[77] N. Einwechter, «An Introduction to Distributed Intrusion Detection Systems,» *Security Focus,* January 2002.

[78] S. Singh y S. Kandula, «Argus: A Distributed Network Intrusion Detection System,» 2001.

[79] M. Asaka, S. Okazawa, A. Taguchi y S. Goto, «A Method of Tracing Intruders by use of Mobile Agents,» de *Proceedings of the 9th Annual Conference of the Internet Society (INET99)*, San Jose - CA, 1999.

[80] J. Quiroz y L. Carmo, «Micael: An Autonomous Mobile Agent System to Protect New Generation Networked Applications,» de *Proceedings of the 2nd Annual Workshop in Recent Advances in Intrusion Detection,* , Rio de Janeiro - Brasil, 1992.

[81] Arbor Networks, «Intelligent Network Management,» June 2015. [En línea]. Available: http://es.arbornetworks.com/.

[82] Riverbed, «http://www.riverbed.com/products/performance-management-control/network-performance-management/,» June 2015. [En línea]. Available: http://www.mazunetworks.com/solutions/white_papers/download/Mazu_Profiler.pdf.

[83] N. Habra, B. LeCharlier, A. Mounji y I. Mathieu, «ASAX: Software Architecture and Rule-Based Language for Universal Audit Trail Analysis,» de *Proccedings of the Second European Symposium on Research in Computer Security (ESORICS)*, Toulouse - France, 1992.

[84] A. Mounji, *Languages and Tools for Rule-Based Distributed Intrusion Detection,* Namur: Facult es Universitaries Notre-Dame de la Paix, 1997.

[85] D. Farmer, «Trouble,» May 1993. [En línea]. Available: http://www.trouble.org/cops/overview.html.

[86] D. Safford, D. Schales y D. Hess, «Proceedings of the fourth USENIX Security Symposium,» de *The Tamu Security Package: An Ongoing Response to Internet Intruders in an Academic Environment*, Santa Clara - CA, 1993.

[87] D. Farmer y W. Venema, «Improving The Security Of Your Site By Breaking Into It,» [En línea]. Available: http://www.trouble.org/security/admin-guide-to-cracking.html.

[88] Network Associates Inc., «Cybercop server,» 1998. [En línea]. Available: http://www.nai.com/products/security/ cybercopsvr/index.asp.

[89] Secure Network Inc, «Ballista Security Auditing System,» 1997. [En línea]. Available: http:// www.securenetworks.com/ballista/ballista.html.

[90] H. Debar, M. Dacier y A. Wespi, «Towards a Taxonomy of Intrusion Detection Systems,» *Computer Networks,* vol. 31, nº 8, pp. 805-822, 1999.

[91] G. Kim y E. Spafford, «Proceedings of the ACM Conference on Computer and Communications Security - COAST,» de *The Design and Implementation of Tripwire: A File System Integrity Checker*, 1994.

[92] D. Vincenzetti y M. Cotrozzi, «Proceedings of the Fourth USENIX Security Symposium,» de *ATP - Anti Tampering Program*, Santa Clara - CA, 1993.

[93] I. Bazara, A. Barry y H. Chan, «Chapter 10. Intrusion Detection Systems,» de *Handbook of information and communication security*, New York, Springer, 2010, pp. 193-205.

[94] K. Ilgun, R. Kemmerer y P. Porras, «State Transition Analysis: A Rule-Based Intrusion Detection Approach,» *IEEE Trans. Soft. Eng,* vol. 21, nº 3, pp. 181-199, 1995.

[95] A. Pagnoni y A. Visconti, «An innate immune system for the protection of computer networks,» de *Proceeding 4th International Symposium on Information and Communication Technologies*, Cape Town, 2005.

[96] K. Srinivasa, «Application of Genetic Algorithms for Detecting Anomaly in Network Intrusion Detection Systems,» *Lecture Notes of the Institute for Computer Sciences, Social*

Informatics and Telecommunications Engineering, vol. 84, pp. 582-591, 2012.

[97] K. Wang y S. Stolfo, «Anomalous Payload-Based Network Intrusion Detection,» de *RAID*, vol. 3224, Heidelberg, Springer, 2004, pp. 203-222.

[98] R. Chen, J. Chen, T. Chen, C. Hsieh, T. Chen y K. Wu, «Building an Intrusion Detection System Based on Support Vector Machine and Genetic Algorithm,» *Advances in Neural Networks,* vol. 3498, pp. 409-414, 2005.

[99] S. Theodoridis y K. Koutroumbas, Pattern Recognition, Burlington : Academic Press - Elsevier , 2009, pp. 1-967.

[100] W. G. 2. o. t. J. C. f. G. i. M. (. 2. l. v. o. m. —. B. a. g. c. a. a. t. VIM, «Bureau International des Poids et Mesures,» 2008. [En línea]. Available: http://www.bipm.org/utils/common/documents/jcgm/JCGM_200_2008.pdf. [Último acceso: 26 Junio 2015].

[101] A. Alazab, M. Hobbs, J. Abawajy y M. Alazab, «Using Feature selection for intrusion detection system,» de *International Symposium on Communications and Information Technologies (ISCIT)* , 2012.

[102] M. Tavallaee, E. Bagheri, W. Lu y A. Ghorbani, «The NSL-KDD Data Set,» 2009. [En línea]. Available: http://nsl.cs.unb.ca/NSL-KDD/. [Último acceso: 26 June 2015].

[103] E. De la Hoz, E. De la Hoz, J. Ortega y A. Ortiz, «Modelo de detección de intrusiones en Sistemas de Red, realizando selección de características con FDR y entrenamiento y clasificación con SOM,» *IngeCUC,* vol. 8, pp. 85-116, 2012.

[104] LL-MIT, «Publications,» 2014. [En línea]. Available: http://www.ll.mit.edu/publications/index.html. [Último acceso: 26 June 2015].

[105] UCI, «KDD Cup 1999 Data,» 28 October 1999. [En línea]. Available: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html. [Último acceso: 26 June 2015].

[106] The University Of Waikato, «WEKA,» 2015. [En línea]. Available: https://weka.wikispaces.com/. [Último acceso: 2015].

[107] S. Zargari y D. Voorhis, «Feature Selection in the Corrected KDD-dataset,» de *EIDWT - Proceedings - 3rd International Conference on Emerging Itelligent Data and Web Technologies*, Bucharest, 2012.

[108] E. Vergard, «Machine Learning For Network Based Intrusion Detection,» 2010. [En línea]. Available: http://www.vengen.co.uk/files/Engen2010-PhD.pdf.

[109] R. Kaur, G. Kumar y K. Kumar, «A Comparative Study of Feature Selection Techniques for Intrusion Detection,» de *2nd International Conference on Computing for Sustainable Global Development*, 2015.

[110] E. De la Hoz, A. Ortiz, E. De la Hoz y J. Ortega, «Network Anomaly Classification by Support Vector Classifiers Ensemble and Non-linear Projection Techniques,» de *Hybrid Artificial Intelligent Systems. 8th International Conference, HAIS*, Salamanca, 2013.

[111] R. Mitra, S. Mazumder, T. Sharma, N. Sengupta y J. Sil, «Dynamic Network Traffic Data Classification for Intrusion Detection Using Genetic Algorithm,» de *Swarm, Evolutionary, and Memetic Computing. Third International Conference, SEMCCO* , Bhubaneswar - India, 2012.

[112] N. Sengupta y J. Sil, «Comparison of Supervised Learning and Reinforcement Learning in Intrusion Domain,» de *Wireless Networks and Computational Intelligence. 6th International Conference on Information Processing, ICIP*, Bangalore - India, 2012.

[113] E. De la Hoz, E. De la Hoz, A. Ortiz, J. Ortega y A. Prieto, «Network Anomaly Detection with Bayesian Self-Organizing Maps,» de *Advances in Computational Intelligence. 12th International Work-Conference on Artificial Neural Networks, IWANN*, Puerto de la Cruz, Tenerife, Spain, 2013.

[114] H. Chauhan, V. Kumar, S. Pundir y E. Pilli, «Comparative Analysis and Research Issues in Classification Techniques for Intrusion Detection,» de *Intelligent Computing, Networking, and Informatics. Proceedings of the International Conference on Advanced Computing, Networking, and Informatics*, India, 2014.

[115] H. Eid, M. Salama, A. Ella y T. Kim, «Bi-Layer Behavioral-Based Feature Selection Approach for Network Intrusion Classification,» de *Security Technology. International Conference, SecTech*, 2011.

[116] H. Eid, A. Ella, T. Kim y S. Banerjee, «Linear Correlation-Based Feature Selection for Network Intrusion Detection Model,» de *Advances in Security of Information and Communication Networks. First International Conference, SecNet*, Cairo - Egypt, 2013.

[117] A. Ortiz, J. Ortega, A. Díaz y A. Prieto, «Network Intrusion Prevention by Using Hierarchical Self-Organizing Maps and Probability-Based Labeling,» de *Advances in Computational Intelligence. 11th International Work-Conference on Artificial Neural Networks, IWANN*, Torremolinos-Málaga, Spain, 2011.

[118] M. Turk y A. Pentland, «Eigenfaces for Recognition,» *Journal of Cognittive Neuroscience,* vol. 3, nº 1, pp. 71-86, 1991.

[119] P. Belhumeur, J. Hespanha y D. Kriegman, «Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection,» *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 19, nº 7, pp. 711-720, 1997.

[120] A. Hyvärinen y E. Oja, «Independent Component Analysis: Algorithms and Applications,» *Neural Networks,* vol. 13, nº (4-5), pp. 411-430, 2000.

[121] V. Bolón-Canedo, N. Sánchez-Maroño y A. Alonso-Betanzos, «A review of feature selection methods on synthetic data,» *Knowledge and Information System,* pp. 483-519, 2012.

[122] J. Handl y J. Knowles, «Feature Subset Selection in Unsupervised Learning via Multiobjective Optimization,» *International Journal of Computational Intelligence Research,* pp. 217-238, 2006.

[123] S. Amari y M. Arbib, «Competition and Cooperation in Neural Nets,» *Springer Verlag,* 1982.

[124] J. Feldman y D. Ballard, «Connectionist Models and Their Properties,» *Cognitive Science,* pp. 205-254, March 1982.

[125] R. Lippmann, «An Introduction to Computing with Neural Nets,» *IEEE ASSP Magazine,* pp. 4-22, April 1987.

[126] P. Treleaven, «Neurocomputers,» *International Journal of Neurocomputing,* vol. 1, pp. 4-31, 1989.

[127] M. Minsky y S. Papert, «Perceptrons,» *MIT Press - Cambridge Mass,* 1969.

[128] J. Hopfield, «Neural Networks and physical systems with emergent collective computational abilities,» de *Proccedings National Academic Science*, 1982.

[129] B. Soucek y M. Soucek, Neural and Massively Parallel Computers, John Wiley & Sons, 1988.

[130] G. Hinton, «Connectionist Learning Procedures,» 1987.

[131] D. Rumelhart y J. McClelland, Parallel distributed processing: exploration in the micro

structure of cognition, Vols. %1 de %21-2, MIT Press, 1986.

[132] D. Ackley, G. Hinton y T. Sejnowski, «A learning algorithm for boltzmann machine,» *Cognitive Science,* vol. 9, pp. 147-169, 1985.

[133] K. Fukushima, «A Neural Network for Visual Pattern Recognition,» *IEEE Computer,* pp. 65-75, March 1988.

[134] T. Kohonen, Self-Organizing Maps, Springer, 2001, p. 502.

[135] D. Phuc y M. X. Hung, «Using SOM based Graph Clustering for Extracting Main Ideas from Documents,» de *IEEE International Conference on Research, Innovation and Vision for the Future*, 2008.

[136] I. Nakaoka, J.-i. Kushida y K. Kamei, «Proposal of Group Decision Support System Using "SOM" for Purchase of Automobiles,» de *The 3rd International Conference on Innovative Computing Information and Control (ICICIC'08)*, 2008.

[137] M. O. Afolabi y O. Olude, «Predicting Stock Prices Using a Hybrid Kohonen Self Organizing Map (SOM),» de *Proceedings of the 40th Annual Hawaii International Conference on System Sciences*, 2007.

[138] I. Manolakos y E. Logaras, «High throughput systolic SOM IP core for FPGAs,» de *IEEE International Conference on Acoustics, Speech and Signal Processing - ICASSP'07*, Honolulu, 2007.

[139] K. Yin y L. Gang, «Fault Pattern Recognition of Thermodynamic System Based on SOM,» de *International Conference on Electrical and Control Engineering (ICECE'10)*, Wuhan, 2010.

[140] H. Ying, W. Li-Qiang y Z. Xi'an, «Automatic Roads Extraction From High-resolution Remote Sensing Images Based on SOM,» de *Sixth International Conference on Natural Computation (ICNC'10)*, Yantai - Shandong, 2010.

[141] H. Tokutaka, K. Yoshihara, K. Fujimura, K. Iwamoto, T. Watanabe y S. Kishida, «Applications of Self-organizing Maps (SOM) to the Composition Determination of Chemical Products,» de *Neural Networks Proceedings, 1998. IEEE World Congress on Computational Intelligence. The 1998 IEEE International Joint Conference on*, Anchorake - AK, 1998.

[142] L. Min y W. Dongliang, «Anormaly Intrusion Detection Based on SOM,» de *WASE International Conference on Information Engineering (ICIE'09)*, Taiyuan - Shanxi, 2009.

[143] J. Patra, J. Abraham, P. Meher y G. Chakraborty, «An Improved SOM-based Visualization Technique for DNA Microarray Data Analysis,» de *The 2010 International Joint Conference on Neural Networks (IJCNN'10)* , Barcelona, 2010.

[144] Y. Venkatesh, S. K. Raja y N. Ramya, «A Novel SOM-based Approach for Active Contour Modeling,» de *Proceedings of the Intelligent Sensors, Sensor Networks and Information Processing Conference*, 2004.

[145] A. Ultsch, «Self-organizing neural networks for visualization and classification,» de *Information and classification: Concepts, Methods and Applications.*, Germany, Springer-Verlag, 1992, pp. 307-313.

[146] D. Merkl y A. Rauber, «Alternative ways for cluster visualization in self-organizing maps,» de *Proceedings Workshop Self-Organizing Maps (WSOM97)*, Finland, 1997.

[147] A. Rauber, «LabelSOM: On the labeling of self-organizing maps,» de *Proceeding International Joint Conference Neural Networks (IJCNN'99)*, Washington DC, 1999.

[148] D. Merkl y A. Rauber, «Automatic labeling of self-organizing maps for information retrieval,»

de *Proceeding 6th International Conference Neural Information (ICONIP'99)*, Perth - Australia, 1999.

[149] S. Santini, «The self-organizing field,» *IEEE Transaction Neural Networks,* vol. 7, pp. 1415-1423, Nov 1996.

[150] C. Bishop, M. Svensen y C. Williams, «GTM: The generative topographic mapping,» *Neural Computing,* vol. 10, nº 1, pp. 215-253, 1998.

[151] R. Miikkulainen, «Script recognition with hierarchical feature maps,» *Connection Science,* vol. 2, pp. 83-101, 1990.

[152] P. Koikkalainen y E. Oja, «Self-organizing hierarchical feature maps,» de *Proceeding International Conference Artificial Neural Networks*, Paris - France, 1995.

[153] P. Koikkalainen, «Fast deterministic self-organizing maps,» de *Proceeding International Conference Artificial Neuronal Networks*, Paris - France, 1995.

[154] J. Blackmore y R. Miikkulainen, «Incremental grid growing: Encoding high-dimensional structure into a two-dimensional feature map,» de *Proceeding IEEE International Conference Neural Networks (ICNN'93)*, San Francisco - CA, 1993.

[155] B. Fritzke, «Growing grid—A self-organizing network with constant neighborhood range and adaption strength,» *Neural Processing Lett,* vol. 2, nº 5, pp. 1-5, 1995.

[156] D. Alahakoon, S. Halgamuge y B. Srinivasan, «Dynamic self-organizing maps with controlled growth for knowledge discovery,» *IEEE Transaction Neural Networks,* vol. 11, pp. 601-614, May 2000.

[157] H. Bauer y T. Villmann, «Growing a hypercubical output space in a self-organizing feature map,» *IEEE Transaction Neural Networks,* vol. 8, pp. 226-233, Mar 1997.

[158] A. Rauber, D. Merkl y M. Dittenbach, «The growing hierarchical self-organizing map: exploratory analysis of high-dimensional data,» *IEEE Transac. Neural Networks,* vol. 13, nº 6, pp. 1331-1341, 2002.

[159] M. Dittenbach, D. Merkl y A. Rauber, «The growing hierarchical self-organizing map,» de *Proceding International Conference Neural Networks (IJCNN 2000)*, 2000.

[160] A. Rauber, D. Merkl y M. Dittenbach, «The Growing Hierarchical Self-Organizing Map: Exploratory Analysis of High-Dimensional Data,» *IEEE Transactions on Neural Networks,* pp. 1331-1341, 2002.

[161] IFS, «The Growing Hierarchical Self-Organizing Map,» 1999. [En línea]. Available: http://www.ifs.tuwien.ac.at/~andi/ghsom/. [Último acceso: 18 Sept 2015].

[162] B. Fritzke, «Growing grid—A self-organizing network with constant neighborhood range and adaption strength,» *Neural Processing Lett,* vol. 2, nº 5, pp. 1-5, 1995.

[163] M. Dittenbach, A. Rauber y D. Merkl, «Recent advances with the growing hierarchical self-organizing map,» de *Proceeding 3rd Workshop Self-Organizing Maps - Advances in Self-Organizing Maps*, U.K., 2001.

[164] T. Kohonen, S. Kaski, K. Lagus, J. Salojärvi, J. Honkela, V. Paatero y A. Saarela, «Self-organization of a massive document collection,» *IEEE Transaction Neural Networks,* vol. 11, pp. 574-85, May 2000.

[165] R. Miikkulainen, «Script recognition with hierarchical feature maps,» *Connection Science,* vol. 2, pp. 83-101, 1990.

[166] M. Grond, N. Loung, J. Morren y J. Slootweg, «Multi-Objective Optimization Techniques and Applications in Electric Power Systems,» *Universities Power Engineering Conference*

(UPEC), 2012 47th International, pp. 1-6, 2012.

[167] M. Irving y Y. Song, «Optimisation techniques for electrical power systems, part 1: mathematical optimisation methods,» *Power Engineering Journal,* vol. 14, nº 5, pp. 245-254, October 2000.

[168] Y. Song y M. Irving, «Optimisation techniques for electrical power systems, part 2: heuristic optimisation methods,» *Power Engineering Journal,* vol. 15, nº 3, pp. 151-160, June 2001.

[169] H. Seifi y M. Sepasian, «Electric Power System Planning: Issues, Algorithms and Solutions,» *Springer,* 2011.

[170] L. Ochoa, *Performance of Distributions Networks with Distributed Generation,* U. E. P. ". d. M. Filhio", Ed., 2006.

[171] R. Romero, R. Gallego y A. Monticelli, «Transmission system expansion planning by simulated annealing,» *IEEE Transactions on Power Systems,* vol. 11, nº 1, pp. 364-369, February 1996.

[172] M. AlRashidi y M. El-Hawary, «A survey of particle swarm optimization applications in electric power systems,» *IEEE Transactions on Evolutionary Computation,* vol. 13, nº 4, pp. 913-918, August 2009.

[173] E. Da Silva, J. Ortiz, G. De Oliveira y S. Binato, «Transmission network expansion planning under a tabu search approach,» *IEEE Transactions on Power Systems,* vol. 16, nº 1, pp. 62-68, February 2001.

[174] K. Lee y J. Vlachogiannis, «Optimization of power systems based on ant colony system algorithms: an overview,» de *Proceedings of the 13th International Conference on Intelligent Systems Application to Power Systems*, 2005.

[175] K. Lee y M. El-Sharkawi, Modern Heuristic Optimization Techniques: Theory and Application to Power Systems, Wiley, 2008.

[176] K. Deb, Multiobjective Optimization Using Evolutionary Algorithms, 2001, p. 518.

[177] C. Fonseca y P. Fleming, «Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization,» de *Proceedings of the Fifth International Conference on Genetic Algorithms*, San Mateo - CA, 1993.

[178] J. Horn, N. Nafploitis y D. Goldberg, «A niched Pareto genetic algorithm for multiobjective optimization,» de *Proceedings of the First IEEE Conference on Evolutionary Computation*, Piscataway - NJ, 1994.

[179] N. Srinivas y K. Deb, «Multiobjective function optimization using nondominated sorting in genetic algorithms,» *Evolutionary Computation,* vol. 2, nº 3, pp. 221-248, 1995.

[180] E. Zitzler y L. Thiele, «Multiobjective optimization using evolutionary algorithms—A comparative case study,» *Parallel Problem Solving From Nature,* pp. 292-301, 1998.

[181] J. Horn, N. Nafpliotis y D. E. Goldberg, «A niched pareto genetic algorithm for multiobjective optimization,» de *Proceedings of the First IEEE Conference on Evolutionary Computation* , 1994.

[182] E. Zitzler, K. Deb y L. Thiele, «Comparison of multiobjective evolutionary algorithms: Empirical results,» *Evolutionary Computational,* vol. 8, nº 2, pp. 173-195, 2000.

[183] J. Knowles y D. Corne, «The Pareto archived evolution strategy: A new baseline algorithm for multiobjective optimization,» de *Proceedings of the 1999 Congress on Evolutionary Computation* , 1999.

[184] K. Deb, A. Pratap, S. Agarwal y T. Meyarivan, «A fast and elitist multiobjective genetic

algorithm: NSGA-II,» *IEEE Transactions on Evolutionary Computation,* vol. 6, nº 2, pp. 182-197, April 2002.

[185] E. Zitzler, «Evolutionary algorithms for multiobjective optimization: Methods and applications,» Zurich, Switzerland, 1999.

[186] K. Deb y D. Goldberg, «An investigation of niche and species for- mation in genetic function optimization,» de *Proceedings of the Third International Conference on Genetic Algorithms*, 1989.

[187] K. Deb, S. Agrawal, A. Pratap y T. Meyarivan, «A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II,» de *Proceedings of the 6th International Conference on Parallel Problem Solving from Nature - PPSN*, London - UK, 2000.

[188] M. Bauer, O. Buchtala, T. Horeis, R. Kern, B. Sick y R. Wagner, «Technical data mining with evolutionary radial basis function classifiers,» *Applied Soft Computing,* vol. 9, nº 2, pp. 765-774, March 2009.

[189] S. T. Powers y J. He, «A hybrid artificial immune system and Self Organising Map for network intrusion detection,» *Information Sciences,* pp. 3024-3042, 1 August 2008.

[190] S. Sen y J. A. Clark, «Evolutionary computation techniques for intrusion detection in mobile ad hoc networks,» *Computer Networks,* vol. 55, nº 15, pp. 3441-3457, October 2011.

[191] M. Govindarajan y R. Chandrasekaran, «Intrusion detection using neural based hybrid classification methods,» *Computer Networks,* vol. 55, nº 8, pp. 1662-1671, 1 June 2011.

[192] A. Hofmann, C. Schmitz y B. Sick, «Intrusion detection in computer networks with neural and fuzzy classifiers,» *Artificial Neural Networks and Neural Information Processing — ICANN/ICONIP 2003,* vol. 2714, pp. 316-324, 18 June 2003.

[193] P. Lichodzijewski, A. Nur Zincir-Heywood y M. Heywood, «Host-based intrusion detection using self-organizing maps,» *Proceedings of the 2002 International Joint Conference on Neural Networks,* vol. 2, pp. 1714-1719, 2002.

[194] C. Zhang, J. Jiang y M. Kamel, «Intrusion detection using hierarchical neural networks,» *Pattern Recognition Letters,* vol. 26, nº 6, pp. 779-791, 2005.

[195] H. Gunes Kayacik, A. Nur Zincir-Heywood y . M. I. Heywood, «A hierarchical SOM-based intrusion detection system,» *Engineering Applications of Artificial Intelligence,* vol. 20, nº 4, pp. 439-451, June 2007.

[196] E. Palomo, E. Domínguez, R. Luque y J. Muñoz, «Network Security Using Growing Hierarchical Self-Organizing Maps,» de *Proceedings of the 9th International Conference on Adaptive and Natural Computing Algorithms, ICANNGA'09*, 2009.

[197] Y. Yang, D. Jiang y M. Xia, «Using improved GHSOM for intrusion detection,» *Journal of Information Assurance and Security,* pp. 232-239, 20 November 2009.

[198] D. Ippoliti y X. Zhou, «A-GHSOM: An adaptive growing hierarchical self organizing map for network anomaly detection,» *Journal of Parallel and Distributed Computing,* vol. 72, nº 12, pp. 1576-1590, December 2012.

[199] R. Kohavi y G. H. John, «Wrappers for feature subset selection,» *Artificial Intelligence,* vol. 97, nº 1-2, pp. 273-324, 1997.

[200] N. Abe, M. Kudo, J. Toyama y M. Shimbo, «Classifier-independent feature selection on the basis of divergence criterion,» *Pattern Analysis and Applications,* vol. 9, nº 2-3, pp. 127-137, October 2006.

[201] P. Jaccard, «Novelles recgerches sur la distribution florale,» *Bulletin de la Socie`te` Vaudoise des Sciences Naturelles,* pp. 223-270, 1908.

[202] M. Anderberg, «Cluster Analysis for Applications,» *Academic Press,* 1973.

[203] L. Shilpa, J. Sini y V. Bhupendra, «Feature Reduction using Principal Component Analysis for Effective Anomaly–Based Intrusion Detection on NSL-KDD,» *International Journal of Engineering Science and Technology,* pp. 1790-1799, 2010.

[204] R. Datti y B. Verma, «Feature Reduction for Intrusion Detection Using Linear Discriminant Analysis,» *International Journal of Advanced Trends in Computer Science and Engineering,* pp. 1072-1078, 2 December 2009.

[205] G. R. Zargar y P. Kabiri, «Selection of effective network parameters in attacks for intrusion detection,» de *Proceedings of the 10th Industrial Conference on Advances in Data Mining. Applications and Theoretical Aspects - ICDM'10*, Berlin - Heidelberg, 2010.

[206] S. Mukkamala y A. H. Sung, «Feature Ranking and Selection for Intrusion Detection Systems Using Support Vector Machines,» de *Proceedings of the Second Digital Forensic Research Workshop*, 2002.

[207] H. Oh, I. Doh y K. Chae, «Attack classification based on data mining technique and its application for reliable medical sensor communication,» *International Journal of Computer Science and Applications,* vol. 6, nº 3, pp. 20-32, 2009.

[208] T. M. Hamdani, J.-M. Won, A. M. Alimi y F. Karray, «Multi-objective feature selection with NSGA II,» de *Proceedings of the 8th International Conference on Adaptive and Natural Computing Algorithms - ICCANGA - LNCS*, 2007.

[209] C. Emmanouilidis, A. Hunter y J. MacIntyre, «A multiobjective evolutionary setting for feature selection and a commonality-based crossover operator,» de *Proceedings of the 2000 Congress on Evolutionary Computation* , New York, 2000.

[210] L. Oliveira, R. Sabourin, F. Bortolozzi y C. Suen, «A methodology for feature selection using multiobjective genetic algorithms for handwritten digit string recognition,» *International Journal of Pattern Recognition and Artificial Intelligence,* vol. 17, nº 6, pp. 903-929, 2003.

[211] Y. Zhang, S. Li, T. Wang y Z. Zhang, «Divergence-based feature selection for separate classes,» *Neurocomputing,* pp. 32-42, 2013.

[212] B. Huang, B. Buckley y T. Kechadi, «Multi-objective feature selection by using NSGA-II for customer churn prediction in telecommunications,» *Expert Systems with Applications,* vol. 37, nº 5, pp. 3638-3646, May 2010.

[213] J. R. Quinlan, «C4.5: Programs for Machine Learning,» *Morgan Kaufmann Publishers Inc,* 1993.

[214] Y. Kim, W. N. Street y F. Menczer, «Evolutionary model selection in unsupervised learning,» *Intelligent Data Analysis,* vol. 6, pp. 531-556, 2002.

[215] M. Morita, R. Sabourin, F. Bortolozzi y C. Suen, «Unsupervised feature selection using multi-objective genetic algorithms for handwritten word recognition,» de *Proceedings of the seventh International Conference on Document Analysis and Recognition*, New York, 2003.

[216] D. L. Davies y D. W. Bouldin, «A cluster separation measure,» *IEEE Transactions on Pattern Analysis and Machine Intelligence ,* pp. 224-227, April 1979.

[217] University of California - Irvine, «KDD Cup 1999 Data,» 28 October 1999. [En línea]. Available: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html. [Último acceso: 15 Agost 2015].

[218] L. Richard, D. Fried, I. Graf, J. Haines, K. Kendall, D. McClung, D. Weber, S. Webster, D. Wyschogrod, R. Cunningham y M. Zissman, «Evaluating intrusion detection systems: the 1998 DARPA off-line intrusion detection evaluation,» de *Proceedings in DARPA Information Survivability Conference and Exposition*, 2000.

[219] J. McHugh, «Testing Intrusion Detection Systems: A Critique of the 1998 and 1999 DARPA Intrusion Detection System Evaluations as Performed by Lincoln Laboratory,» *ACM Transactions on Information and System Security,* vol. 3, nº 4, pp. 262-294, November 2000.

[220] E. Uriarte y F. Martín, «Topology Preservation in SOM,» *International Journal of Applied Mathematics and Computer Sciences,* pp. 19-22, 2005.

[221] M. Tavallaee, N. Stakhanova y A. A. Ghorbani, «Toward credible evaluation of anomaly-based intrusion-detection methods,» *IEEE Transactions on Systems, man and Cybernetics,* vol. 40, nº 5, pp. 516-524, September 2010.

[222] H. G. Kayacik, A. N. Zincir-Heywood y M. Heywood, «A hierarchical SOM-based intrusion detection system,» *Engineering Applications of Artificial Intelligence,* vol. 20, pp. 439-451, 2007.

[223] E. Eskin, A. Arnold, M. Prerau, L. Portnoy y S. Stolfo, «A geometric framework for unsupervised anomaly detection: detecting intrusions in unlabeled data,» *Applications of Data Mining in Computer Security,* pp. 77-101, 2002.

[224] T. Hanne, «On the convergence of multiobjective evolutionary algorithms,» *European Journal of Operational Research,* vol. 177, pp. 553-564, 1999.

[225] E. Zitzler y L. Thiele, «Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach,» *IEEE Transactions on Evolutionary Computation,* vol. 3, nº 4, pp. 257-271, november 1999.

[226] W. Navidi, Statistics for Engineers and Scientists, Third ed., McGraw-Hill, 2010.

[227] M. Tavallaee, E. Bagheri, W. Lu y A. Ghorbani, «A Detailed Analysis of the KDD CUP 99 Data Set,» de *Proceedings of the 2009 IEEE Symposium on Computational Intelligence in Security and Defense Applications (CISDA 2009)*, 2009.

[228] M. Panda, A. Abraham y M. Patra, «Discriminative multinomial Naïve Bayes for network intrusion detection,» de *Sixth International Conference on Information Assurance and Security*, 2010.

[229] Z. Yu, J. Tsai y T. Weigert, «An adaptive automatically tuning intrusion detection system,» *ACM Transaction Auton. Adaptive System,* vol. 3, nº 3, pp. 10-25, 2008.

[230] S. Sivatha Sindhu, S. Geetha y A. Kannan, «Decision tree based light weight intrusion detection using a wrapper approach,» *Expert Systems Application,* vol. 39, nº 1, pp. 129-141, 2012.

[231] M. Bahrololum y M. Khaleghi, «Anomaly Intrusion Detection System Using Gaussian Mixture Model,» *Third International Conference Convergence and Hybrid Information Technology, 2008. ICCIT '08.,* vol. 1, pp. 1162-1167, 2008.

[232] M. Leech, M. Ganis, Y. Lee, R. Kuris, D. Koblas y L. Jones, *SOCKS Protocol Version 5,* 1996.

[233] R. Sekar, Y. Guang, S. Verma y T. Shanbhag, «Proceeding 6th ACM Conference on Computer and Communication Security,» de *A high- performance network intrusion detection system,*, Singapure, 1999.

[234] R. Sekar, A. Gupta, J. Frullo, T. Shanbhag, A. Ti-wari, H. Yang y S. Zhou, «ACM Computer and Communication Security Conference (CCS),» de *Specification-based anomaly*

detection: A new approach for detecting network intrusions, Whashington DC, 2002.

[235] K. Kendall, *A Database of Computer Attacks for the Evaluation of Intrusion Detection Systems,* Massachusetts Institute of Technology, 1998.

[236] J. Howard, *An Analysis of Security Incidents on the Internet,* Pittsburgh: Carnegie Mellon University, 1997.

[237] J. Christy, «Cyber Threat & Legal Issues,» de *Proceedings of the ShadowCon'99*, Dahlgren - Virginia, 1999.

[238] R. Singh, H. Kumar y R. Singla, «Analyzing Statistical Effect of Sampling on Network Traffic Dataset,» de *ICT and Critical Infrastructure: Proceedings of the 48th Annual Convention of Computer Society*, Switzerland, 2014.

[239] A. Saved, A. Ella y S. Taher, «Machine Learning Techniques for Anomalies Detection and Classification. Advances in Security of Information and Communication Networks,» de *First International Conference, SecNet.* , Cairo, 2013.

[240] S. Mazumder, T. Sharma, R. Mitra, N. Sengupta y J. Sil, «Generation of Sufficient Cut Points to Discretize Network Traffic Data Sets,» de *Third International Conference, SEMCCO*, Bhubaneswar - India, 2012.

[241] Y. Hou y X. Feng, « SVM Based MLP Neural Network Algorithm and Application in Intrusion Detection,» de *Artificial Intelligence and Computational Intelligence. Third International Conference, AICI*, Taiyuan - China, 2011.

[242] A. Webb y K. D. Copsey, Statistical Pattern Recognition, Third Edition ed., J. W. &. S. Ltd, Ed., WILEY, 2011, p. 514.

[243] K. Fukunaga, Introduction to Statistical Pattern Recognition, Second Edition ed., A. P. Inc, Ed., 1990.

[244] S. Suthaharan y T. Panchagnula, «Relevance Feature Selection with Data Cleaning for Intrusion Detection System,» de *Conference Proceedings - IEEE SOUTHEASTCON*, Orlando - United States, 2012.

[245] S. Zargari y D. Voorhis, «Feature Selection in the Corrected KDD-dataset,» de *Third International Conference on Emerging Intelligent Data and Web Technologies*, Bucharest - Romania, 2012.

[246] F. Zhang y D. Wang, «An Effective Feature Selection Approach for Network Intrusion Detection,» de *Eighth International Conference on Networking, Architecture and Storage*, Xi'an - Shaanxi - China, 2013.

[247] M.-L. Shyu, S.-C. Chen, K. Sarinnapakorn y L. Chang, «Principal Component-based Anomaly Detection Scheme,» *Studies in Computational Intelligence,* vol. 9, pp. 311-329, 2006.

[248] M. Abdi y H. Szu, «Independent component analysis (ICA) and self-organizing map (SOM) approach to multidetection system for network intruders,» de *Proceedings Independent Component Analyses, Wavelets, and Neural Networks*, Orlando - Florida, 2003.

[249] Y. Gu, B. Zhou y J. Zhao, «PCA-ICA ensembled intrusion detection system by Pareto-Optimal optimization,» *Information Technology Journal,* vol. 7, nº 3, pp. 510-515, 2008.

[250] N. Sun y Y. Li, «Intrusion detection based on back-propagation neural network and feature selection mechanism,» de *1st International International Mega-Conference on Future Generation Information Technology*, Jeju Island - South Korea, 2009.

[251] Ł. Saganowski, M. Goncerzewicz y T. Andrysiak, «Anomaly Detection Preprocessor for

SNORT IDS System,» de *Image Processing and Communications Challenges*, R. S. Chorás, Ed., Springer-Verlag Berlin Heidelberg, 2013, pp. 225-232.

[252] A. Namik y Z. Othman, «Reducing Network Intrusion Detection Association Rules Using Chi-Squared Pruning Technique,» de *3rd Conference on Data Mining and Optimization*, Selangor - Malaysia, 2011.

[253] N. Muraleedharan, A. Parmar y M. Kumar, «A flow based anomaly detection system using chi-square technique,» de *2nd International Advance Computing Conference - IACC*, Patiala - India, 2010.

[254] Y. Gong, Y. Fang, L. Liu y J. Li, «Multi-Agent Intrusion Detection System Using Feature Selection Approach,» de *Tenth International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, Kitakyushu - Japan, 2014.

[255] T. Hastie, R. Tibshirani y J. Friedman, The Elements of Statistical Learning Data Mining, Inference, and Prediction, Second ed., Springer, 2009.

[256] J. Quinlan, «Induction of Decision Trees,» *Machine Learning,* pp. 81-106, 1986.

[257] F. Fleuret, «Fast Binary Feature Selection with Conditional Mutual Information,» *Journal of Machine Learning Research,* pp. 1531-1555, 2004.

[258] G. Wang y F. Lochovsky, «Feature selection with conditional mutual information maximin in text categorization,» de *Proceedings of the Thirteenth ACM International Conference on Information and Knowledge Management - CIKM*, New York - USA, 2004.

[259] M. Hall, «Correlation-based feature selection for discrete and numeric class machine learning,» de *Proceedings of the Seventeenth International Conference on Machine Learning - ICML*, San Francisco CA - USA, 2000.

[260] L. Yu y H. Liu, «Efficient feature selection via analysis of relevance and redundancy,» *Journal of Machine Learning Research,* pp. 1205-1224, 2004.

[261] A. Patcha y J.-M. Park, «An overview of anomaly detection techniques: Existing solutions and latest technological trends,» *Computer Networks,* pp. 3448-3470, 2007.

[262] W. Khreich, E. Granger, A. Miri y R. Sabourin, «Adaptive ROC-based ensembles of HMMs applied to anomaly detection,» *Pattern Recognition,* vol. 45, nº 1, pp. 208-230, January 2012.

[263] «Feature Ranking and Selection for Intrusion Detection Systems Using Support Vector Machines».

[264] J. Z. Lei y A. A. Ghorbani, «Improved competitive learning neural networks for network intrusion and fraud detection,» de *Neurocomputing - Brazilian Symposium on Neural Networks (SBRN 2010) International Conference on Hybrid Artificial Intelligence Systems (HAIS 2010)*, 2012.

[265] T. Wang, S. Mabu, L. Nannan y K. Hirasawa, «A novel intrusion detection system based on the 2-dimensional space distribution of average matching degree,» de *Proceedings of SICE Annual Conference (SICE)*, 2011.

[266] D. Fisch, A. Hofmann y B. Sick, «On the versatility of radial basis function neural networks: A case study in the field of intrusion detection,» *Information Sciences,* vol. 180, nº 12, 15 June 2010.

[267] D. Fisch y B. Sick, «Training of radial basis function classifiers with resilient propagation and variational Bayesian inference,» *International Joint Conference on Neural Networks,* pp. 838-847, 14-19 June 2009.

[268] A. Ortiz, J. Ortega, A. F. Diaz y A. Prieto, «Network Intrusion Prevention by Using Hierarchical Self-Organizing Maps and Probability-Based Labeling,» *Advances in Computational Intelligence - Lecture Notes in Computer Science,* vol. 6691, pp. 232-239, 2011.

[269] J.-P. Nziga, «Minimal Dataset for Network Intrusion Detection Systems via Dimensionality Reduction,» de *Six International Conference on Digital Information Management (ICDIM)*, 2011.

[270] P. Wasserman y T. Schwartz, «Neural Networks, Part 2,» *IEEE Expert,* vol. 3, nº 1, pp. 10-15, 1988.

[271] W. McCulloch y W. Pitts, «A logical calculus of the ideas immanent in nervous activity,» *The bulletin of Mathematical Biophysics,* pp. 115-133, 1943.

[272] R. Rosenblatt, «The Perceptron: A probabilistic model for information storage and organization in the brain,» *Psychological Review ,* vol. 65, pp. 386-408, 1958.

[273] P. Rowcliffe, J. Feng y H. Buxton, «Spiking Perceptrons,» *IEEE Transactions on Neural Networks,* vol. 17, nº 3, pp. 803-807, 2006.

[274] J. Sima, «Training a single sigmoidal neuron Is hard,» *Neural Computation,* vol. 14, pp. 2709-2728, 2002.

[275] R. Rosenblatt, Principles of neurodynamics, Spartan Books, 1962.

[276] B. Widrow y M. Hoff, «Adaptive switching circuits,» de *IRE Eastern Elec- tronic Show & Convention (WESCON)* , 1960.

[277] M. Gori y M. Maggini, «Optimal convergence of on-line backpropagation,» de *IEEE Transactions on Neural Networks*, 1996.

[278] S. Gallant, «IEEE Transactions on Neural Networks,» de *Perceptron-based learning algorithms* , 1990.

[279] M. Muselli, «On convergence properties of pocket algorithm,» de *IEEE Transactions on Neural Networks*, 1997.

[280] B. Widrow y M. Lehr, «30 years of adaptive neural networks,» de *Proceedings of the IEEE*, 1990.

[281] P. Bouboulis y S. Theodoridis, «Extension of Wirtinger's calculus to reproducing kernel Hilbert spaces and the complex kernel LMS,» de *IEEE Transactions on Signal Processing*, 2011.

[282] D. Bolle y G. Shim, «Nonlinear Hebbian training of the perceptron,» *Network,* vol. 6, pp. 619-633, 1995.

[283] C. Mays, *Adaptive threshold logic,* S. University, Ed., 1963.

[284] R. Duda y P. Hart, Pattern classification and scene analysis, New York: Wiley, 1973.

[285] Y. Ho y R. Kashyap, «An algorithm for linear inequalities and its applications,» *IEEE Transactions of Electronic Computers,* vol. 14, pp. 683-688, 1965.

[286] F. Vallet, «The Hebb rule for learning linearly separable Boolean functions: Learning and generalisation.,» *Europhysics Letters,* vol. 8, pp. 747-751, 1989.

[287] M. Amin y K. Murase, «Single-layered complex-valued neural network for real-valued classification problems,» *Neurocomputing,* vol. 72, pp. 945-955, 2009.

[288] C. Eitzinger y H. Plach, «A new approach to perceptron training,» *IEEE Transaction on Neural Networks,* vol. 14, pp. 216-221, 2003.

[289] A. Mansfield, «Training perceptrons by linear programming,» 1991.

[290] S. Perantonis y V. Virvilis, «Efficient perceptron learning using constrained steepest descent,» *Neural Networks,* vol. 13, pp. 351-364, 2000.

[291] J. Chen y J. Chang, «Fuzzy perceptron neural networks for classifiers with numerical data and linguistic rules as inputs.,» *IEEE Transactions on Fuzzy Systems,* vol. 8, pp. 730-745, 2000.

[292] G. Nagaraja y R. Bose, «Adaptive conjugate gradient algorithm for perceptron training,» *Neurocomputing,* vol. 69, pp. 368-386, 2006.

[293] M. Hassoun y J. Song, «Adaptive Ho-Kashyap rules for perceptron training,» *IEEE Transactions on Neural Networks,* vol. 3, pp. 51-61, 1992.

[294] D. Rumelhart, G. Hinton y R. Williams, «Learning internal representations by error propagation,» de *Parallel Distributed Processing: Explorations in the Microstructure of Cognition* , Cambridge: MIT Press, 1986, pp. 318-362.

[295] P. Werbos, *Beyond Regressions: New Tools for Prediction and Analysis in the Behavioral Sciences,* 1974.

[296] V. Maiorov y A. Pinkus, «Lower bounds for approximation by MLP neural networks,» *Neurocomputing,* vol. 25, pp. 81-91, 1999.

[297] M. Manry, S. Apollo, L. Allen, W. Lyle, W. Gong y M. Dawson, «Fast training of neural networks for remote sensing,» *Remote Sensing Reviews,* vol. 9, pp. 77-96, 1994.

[298] S. Narayan, «The generalized sigmoid activation function: Competitive supervised learning,» *Information Sciences,* vol. 99, pp. 69-82, 1997.

[299] D. Rumelhart, R. Durbin, R. Golden y Y. Chauvin, «Backpropagation: the basic theory,» de *Backpropagation: Theory, Architecture, and Applications* , 1995, pp. 1-34.

[300] G. Rudolph, «Evolutionary search under partially ordered sets,» Dortmund - Germany, 1999.

[301] C. Fonseca y P. Fleming, «Multiobjective optimization and multiple constraint handling with evolutionary algorithms - Part II: Application example,» *IEEE Transaction on Systems, Man and Cybernetics,* vol. 28, nº 1, pp. 38-47, 1998.