
UNIVERSIDAD NACIONAL DE LA PLATA

Facultad de INFORMÁTICA

# OPTIMIZACIÓN DE TRÁFICO EN REDES MULTISERVICIOS APLICANDO TÉCNICAS HEURÍSTICAS

Tesis presentada para obtener el grado de
Magíster en Redes de Datos

**Tesista:**
Javier Alejandro Carletto

**Directores**
Jose Luís Hernández
Francisco Javier Diaz

2022

# OPTIMIZACIÓN DE TRÁFICO EN REDES MULTISERVICIOS APLICANDO TÉCNICAS HEURÍSTICAS

Tesis presentada para obtener el grado de Magister en Redes de Datos

Tesista:

Javier Alejandro Carletto

Directores:

José Luís Hernández

Francisco Javier Diaz

Facultad de Informática - Universidad Nacional de La Plata

Diciembre 2022

# Agradecimientos

*A mis amores, Julieta y Luz, por el esfuerzo compartido.*

*A mis familiares, amigos y compañeros de trabajo por su apoyo permanente.*

*A mis directores por su tiempo, dedicación y confianza.*

*Y muy especialmente a José, por su disposición permanente, su apoyo constante, su integridad personal, su humor inteligente, y por todo lo que me enseñó en tantas horas compartidas a la distancia.*

# Resumen

El abrupto crecimiento del tráfico presente en las redes convergentes actuales, trae como consecuencia la implementación de nuevas tecnologías que permiten ofrecer a los usuarios mayores anchos de banda para lo cual es necesario realizar una distribución óptima del tráfico, tomando algún criterio de desempeño y teniendo en cuenta la elasticidad del flujo que involucra atender tráficos tan disímiles como voz, video, sonido, datos, entre otros.

Optimizar la distribución de distintos requerimientos considerando estos aspectos en redes multiservicios permite garantizar la disponibilidad de la red para los requerimientos de tráfico, cuando las demandas modernas ponen en riesgo de congestión a las redes que utilizan las técnicas tradicionales de conmutación.

MPLS (conmutación de etiquetas multiprotocolo) se ha convertido en una tecnología eficaz en la solución a estos inconvenientes, aunque el problema de la selección de la mejor ruta y de la distribución de tráfico no solo sigue existiendo, sino que exige nuevas propuestas de optimización del enrutamiento.

En muchos casos, la planificación óptima de distribución de tráfico en redes MPLS, conlleva la necesidad de resolver un problema de optimización combinatorio de características tales que, para instancias medias o grandes del problema, los métodos determinísticos no son adecuados desde el punto de vista del tiempo de ejecución necesario para obtener el óptimo. En este punto las heurísticas, constituyen una alternativa válida para proporcionar buenas soluciones en tiempos aceptables.

En esta tesis se presenta una taxonomía de estrategias heurísticas y metaheurísticas con el objetivo de distribuir los requerimientos en los enlaces disponibles de una red minimizando el costo de enrutamiento, al tiempo que se satisfacen restricciones en cuanto a demanda y capacidad de cada enlace. Se presenta el desarrollo, descripción y modelado del problema, se diseñan diferentes algoritmos bio-inspirados en el comportamiento de enjambres que brindan una solución de configuración fuera de línea, a este problema tradicional de la ingeniería de tráfico en redes con alta interconectividad.

Se implementan cinco algoritmos inspirados en bandadas de pájaros, colonias de hormigas y el comportamiento de quirópteros, que permiten determinar una solución óptima explorando el espacio de búsqueda desde diferentes estrategias. Se ejecutan los algoritmos sobre cuatro redes de ensayo de diferentes tamaños, con lo que se determina la aplicabilidad de los algoritmos, y los parámetros óptimos de funcionamiento en cada caso, se presenta el análisis comparativo de los resultados obtenidos y se dejan planteadas distintas opciones de trabajos e investigaciones a futuro.

**Palabras Claves:** MPLS, optimización, algoritmos bio-inspirados, optimización por colonia de hormigas, optimización por enjambres de partículas, optimización por algoritmo de murciélagos

# Abstract

The abrupt growth of traffic in today's convergent networks has led to the implementation of new technologies to offer users higher bandwidths, for which it is necessary to make an optimal traffic distribution, taking into account performance criteria and the elasticity of the flow that involves serving traffic as dissimilar as voice, video, sound, data, among others.

Optimizing the different requirements distribution considering these aspects in multiservice networks makes it possible to guarantee network availability for traffic requirements, when modern demands put networks using traditional switching techniques at risk of congestion.

MPLS (multiprotocol label switching) has become an effective technology in solving these problems, although the problem of selecting the best route and traffic distribution not only continues to exist, also requires new routing optimization proposals.

In many cases, the optimal planning of traffic distribution in MPLS networks involves the need to solve a combinatorial optimization problem, which, for medium or large instances of the problem, deterministic methods are not adequate from the point of view of the execution time required to obtain the optimum. At this point, heuristics are a valid alternative to provide good solutions in acceptable times.

In this thesis a taxonomy of heuristic and metaheuristic strategies is presented to distributing the requirements in the available network’s links minimizing the routing cost, while satisfying demand and links capacity. The development, description and modeling of the problem are presented, different bio-inspired algorithms are designed based on swarm behavior that provide an off-line configuration solution to this traditional networks high interconnectivity traffic engineering problema.

Five algorithms inspired by bird flocks, ant colonies and bat behavior are implemented to determine an optimal solution by exploring the search space from different strategies. The algorithms are executed on four different sizes test networks, thus determining the applicability of the algorithms and the optimal operating parameters in each case. A results obtained comparative analysis is presented and different options for future work and research are discussed.



**Keywords:** MPLS, optimization, bio-inspired algorithms, ant colony optimization, particle swarm optimization, bat algorithm optimization

# Índice general

# Índice de Tablas

# Índice de Figuras

# Índice de Algoritmos (pseudocódigos)

# Nomenclatura

## Acrónimos

**4G LTE:** *Fourth Generation Cellular Networks - Long Term Evolution*

**ACO:** *Ant Colony Optimization*
**ATM:** *Asyncronous Transfer Mode*

**BA:** *Bat Algorithm*
**BBA:** *Binary Bat Algorithm*
**BGP:** *Border Gateway Protocol*
**BPA:** *BA post-optimizado con ACO*

**CBR:** *Constraint based routing*
**CCC:** *Circuit Cross Connect*
**CoS:** *Class of Service*
**CR-LDP:** *Constraint-based Routing Label Distribution Protocol*

**DDoS:** *Distributed Denial of Service*
**DEA:** *Dolphin echolocation algorithm*
**DLCI:** *Data Link Connection Identifier*
**DSL:** *Digital Subscriber Line*

**ELS:** *Elitist learning strategy*
**ERP:** *Enterprise resource planning*
**EXP:** *Experimental (campo de Etiqueta MPLS)*

**FEC:** *Forwarding Equivalence Class*

**GA:** *Genetic Algorithms*
**GMPLS:** *Generalized Multiprotocol Label Switching*

**HDLC:** *High-Level Data Link Control,*

**LER:** *Label Edge Routers*
**LFIB:** *Label Forwarding Information Base*
**LIB:** *Label Information Base*
**LSP:** *Label Switched Path*
**LSR:** *Label Switch Routers*

**MPLS:** *Multiprotocol Label Switching*

**NP:** *nondeterministic polynomial time*

**OSI:** *Open System Interconnection*
**OSPF:** *Open Shortest Path First*

**PBIL:** *Population-based incremental learning*
**PPP:** *Point-to-Point Protocol*
**PSO:** *Particle swarm optimization*
**PVC:** *Permanent virtual circuitUI*

**QoS:** *quality of service*

**RIP:** *Routing Información Protocol*
**RFC:** *Request for Comments*
**RSVP:** *Resource Reservation Protocol*
**RSVP-TE:** *Resource Reservation Protocol - Traffic Engineering*

***SA:*** *Simulated Annealing*
**SDH:** *Synchronous Digital Hierarchy*
**SDN:** *Software-Defined Network*
**SD-WAN:** *Software-Defined Wide Area Network*
**SI:** *Swarm Intelligence*
**SONET:** *Synchronous Optical Network*
**SS:** *Scatter Search*
**TC:** *Trafic class*

**IETF:** *Internet Engineering Task Force*
**IGP:** *Interior Gateway Protocol*
**IoE:** *Internet Of Everything*
**IoT:** *Internet of Things*
**IP:** *Internet Protocol*
**IPSec:** *Internet Protocol security*
**IPv4:** *Internet Protocol version 4*
**IPv6:** *Internet Protocol version 6*
**IS-IS:** *Intermediate System to intermediate System*
**ISO:** *International standard organization*
**ISP:** *Internet Service Provider*

**L3VPN:** *Layer 3 VPNs*
**LAN:** *Local Area Network*
**LDP:** *Label Distribuion Protocol*

**TCP:** *Transmission Control Protocol*
**TDP:** *Tag Distribution Protocol*
**TE:** *Trafic Engineers*
**TS:** *Tabu Search*
**TTL:** *Time-To-Live*

**UDP:** *User Datagram Protocol*

**VCI:** *Virtual Channel Identifier*
**VoD:** *Video on demand*
**VoIP:** *Voice over Internet Protocol*
**VPI:** *Virtual Path Indenifier*
**VPN:** *virtual private network*
**VPLS:** *Virtual Private LAN Service*

**WAN:** *Wide Area Network*

# Referencias matemáticas y algorítmicas

## Relacionado al modelado

***A:*** *matriz de adyacencia*
**$Cap_e$:** *Matriz de Capacidad*
**$Costo_e$:** *Matriz de Costo de los enlaces*
***Fitness:*** *Valor de aptitud de la solución*
***$L_t$:*** *cantidad de caminos posibles del requerimiento t*
***$P_t$:*** *conjunto de los caminos posibles para el requerimiento t*
**T:** *Matriz de requerimientos*
***u/seg:*** *Unidades por segundo (referencia a requerimientos)*
***X*:** *Vector solución*

## Relacionado a PSO / Algoritmo PSO

$x_i^t$***:*** *Posiciones de la partícula i en el tiempo t*
$v_i^t$***:*** *Velocidad de la partícula i en el instante t,*
$w$***:*** *Factor de aceleración*

## Relacionado a ACO / Algoritmo ACO 1 / Algoritmo ACO 2 / Algoritmo BPA

$\alpha$**:** *Factor de peso de feromona* $\tau$
**β:** *Factor de peso del valor heurístico η*
$Cap_e$**:** *Capacidad del enlace e*
$CapU_p(p)$**:** *Capacidad útil del camino p.*
$Carga_e$**:** *Carga del enlace e*
$\mathbf{Costo(p)}$**:** *Costo del camino p.*
$\Delta\tau$**:** *Rastro de feromona que deja la hormiga.*
**η*(r,s):*** *Información heurística del enlace r-s*
$M_k$**:** *Conjunto de nodos conectados*
$\rho$**:** *Constante de evaporación.*
$\tau(p)^t$**:** *Feromona del camino p en el instante t.*
$\tau_{rs}^{\alpha}$**:** *Nivel de feromona del enlace r-s*
$x_p$**:** *Cantidad de flujo asignado al camino p*

$c_1$: *Factor de aprendizaje individual*

$c_2$: *Factor de aprendizaje social*

$r_1^t$ *y* $r_2^t$: *Valores aleatorios en el rango [0,1] para el tiempo t.*

$p_i^{best}$: *Mejor posición conocida de la partícula i*

$G_{Best}$: *Mejor posición conocida por el enjambre (Óptimo Global)*

### Relacionado a PSO / Algoritmo PSO

$x_i^t$: *Posiciones de la partícula i en el tiempo t*

$v_i^t$: *Velocidad de la partícula i en el instante t,*

$w$: *Factor de aceleración*

$c_1$: *Factor de aprendizaje individual*

$c_2$: *Factor de aprendizaje social*

$r_1^t$ *y* $r_2^t$: *Valores aleatorios en el rango [0,1] para el tiempo t.*

$p_i^{best}$: *Mejor posición conocida de la partícula i*

$G_{Best}$: *Mejor posición conocida por el enjambre (Óptimo Global)*

### Relacionado a BA / Algoritmo BA / Algoritmo BPA

$x_i$: *Posición del murciélago*

$v_i$: *Velocidad del murciélago*

$f_i$: *Frecuencia de los pulsos emitidos*

$x^*$: *Mejor solución obtenida hasta ese momento*

β: *Valor aleatorio*

$w$: *Factor de aceleración.*

$\epsilon$: *Valor aleatorio*

$A^t$: *Promedio del volumen de todos los murciélagos en el tiempo t*

$A_0$: *Volumen inicial*

$A_{min}$: *Volumen Mínimo*

$r_i^0$: *Tasa de pulsos inicial.*

$r_i$: *Tasa de pulsos.*

$\alpha$ y $\gamma$: *Parámetros de configuración*

$f_{min}$: *Frecuencia mínima emitida por los murciélagos*

$f_{max}$: *Frecuencia máxima emitida por los murciélagos*

# Terminología propia

**Algoritmo ACO 1:** *Algoritmo desarrollado inspirado en colonia de hormigas que distribuye los flujos.*

**Algoritmo ACO 2:** *Algoritmo desarrollado inspirado en colonia de hormigas que redistribuye los flujos.*

**Algoritmo BA:** *Algoritmo desarrollado inspirado en murciélagos*

**Algoritmo BPA:** *Algoritmo híbrido desarrollado inspirado en murciélagos y hormigas*

***Algoritmo PSO:*** *algoritmo desarrollado inspirado en enjambres de partículas*

**Post-Optimizador:** *Algoritmo de post-optimización*

**Post-Optimización:** *Proceso aplicado en forma posterior a un algoritmo de optimización con el fin de mejorar las soluciones.*

# Contribuciones

En el marco del desarrollo de la presente tesis, se han realizado las siguientes publicaciones:

- Javier Carletto, Mercedes Carnero, Javier Díaz, José Hernández (2022) *"Algoritmo Bioinspirado en Murciélagos Aplicado a Optimización en redes MPLS"* 6º Congreso Latinoamericano de Ingeniería y Ciencias Aplicadas "CLICAP 2022" Facultad de Ciencias Aplicadas a la Industria, Universidad Nacional de Cuyo. San Rafael Mendoza 6, 7 y 8 de abril de 2022.
- Javier Carletto, José Hernández, Javier Díaz, (2022) *"Algoritmos bioinspirados aplicados en optimización de redes de datos"* Semana de la Ingeniería – Semana FICA. Revista IdentiFICA ISSN 2953-383X. Facultad de Ingeniería y Ciencias Agropecuarias, Universidad Nacional de San Luis. Villa Mercedes, San Luis, 6 al 10 de junio 2022.
- Javier Carletto (2022) *"Optimización de distribución de tráfico en redes MPLS mediante heurísticas bioinspiradas"* Disertación en el Seminario de Investigación, Actualización y Vinculación en Ciencias de la Ingeniería 2022 (SIAVCI 2022). Facultad de Ingeniería y Ciencias Agropecuarias, Universidad Nacional de San Luis. Villa Mercedes, San Luis, 16 de septiembre 2022.
- Javier Carletto, José Hernández, Javier Díaz, (2022) *"Estudio comparativo de heurísticas bioinspiradas para optimización de redes MPLS"* 51 JAIIO Jornadas Argentinas de Informática SIIIO 2022 – Simposio Argentino de Informática Industrial e Investigación Operativa, 17 al 27 de octubre 2022.

# Capítulo 1
# Introducción

## 1.1 Introducción

Las redes de todo el mundo que forman Internet han sufrido, y sufren en la actualidad, un crecimiento vertiginoso fundamentalmente en el número de usuarios que la utilizan, y por consiguiente un aumento del tráfico de datos debido a los servicios que se van agregando a las redes convergentes, tales como VoIP, TV sobre Internet, radio sobre Internet, video *streaming* multi-punto, entre otros. Esto trae como consecuencia la demanda cada vez mayor de nuevos y mejores servicios que Internet debe ofrecer, lo que obliga a plantearse la implementación de nuevas tecnologías que permitan disponer mayores anchos de banda.

El algoritmo típico de enrutamiento que determina la “ruta más corta” puede producir el congestionamiento de ciertos enlaces, mientras existen otras rutas disponibles que no son utilizadas. Este tipo de enrutamiento puede provocar demoras impredecibles y pérdida de datos. Si bien esto no ha representado un problema para las aplicaciones tradicionales de Internet, tales como web, correo electrónico, transferencia de archivos y similares, la nueva generación de aplicaciones, exige alto rendimiento, mayor ancho de banda y baja latencia (Hernández Camacho, 2015). Tradicionalmente estos algoritmos de enrutamiento se han basado en minimizar su función objetivo, sin considerar el tráfico en sí, es decir sin optimizar el rendimiento de la red. Como ejemplos de estos, podemos mencionar a RIP (Routing Información Protocol), basado en el método denominado vector de distancias y el OSPF (Open Shortest Path First), que es el de mayor utilización basado en el método conocido como link-state method. Ambos métodos escogen el camino con el menor costo entre pares de nodos, lo cual puede llevar a "cuellos de botella" o congestión, ya que siempre se escoge el mejor camino para enviar todo el tráfico mientras otros caminos permanecen sin uso. (Gonzales Cos, 2012)

Estos algoritmos tradicionales de enrutamiento no tienen la suficiente flexibilidad para satisfacer las demandas crecientes de tráfico en las redes modernas. Los requerimientos de tráfico y calidad de servicio en las redes actuales, requieren optimizar el rendimiento de modo de satisfacer adecuadamente la demanda de los servicios en la red.

En redes dinámicas, el flujo de paquetes no es estático y sigue un comportamiento estocástico, lo cual lo hace muy difícil de modelar. El algoritmo de enrutamiento debe manejar un conjunto de funcionalidades básicas para manejar la congestión, control, encolamiento y tráfico generado por los usuarios. Existen muchos posibles problemas de ruteo, de acuerdo a las características de la red y del tráfico. Las condiciones del tráfico cambian constantemente y la estructura de la red también puede cambiar debido a fallas en routers y/o enlaces, o nuevos nodos que se agregan. La tarea principal del algoritmo de enrutamiento es direccionar los paquetes de datos desde su nodo de origen a su nodo de destino maximizando el rendimiento de la red y minimizando los costos (demora de los paquetes). (Gonzales Cos, 2012)

Existen varios mecanismos para optimizar el rendimiento que incluyen el modelado, medición, caracterización y el control de tráfico en una red para obtener objetivos específicos de rendimiento y ofrecer servicios competitivos a los clientes. Uno de estos mecanismos de transporte

de datos estándar, creado por la IETF y definido en el RFC 3031, se la conoce como la conmutación de etiquetas multiprotocolo comúnmente denominada MPLS por sus siglas en inglés, MultiProtocol Label Switching (Rosen *et al.,* 2001)

MPLS constituye un elemento clave en el despliegue de técnicas de Ingeniería de Tráfico ya que su idea básica es separar completamente el plano de control (enrutamiento) del plano de datos (reenvío de paquetes) mientras mantiene la compatibilidad con las infraestructuras de red IP existentes (Hernández Camacho, 2015). Esta separación de los planos de control y de envío de datos, constituye una de las principales características conceptuales de MPLS. Mientras al plano de control le conciernen las funciones de coordinación a nivel de red, en el plano de datos se realizan operaciones simples de conmutación de etiquetas. De esta manera es posible dar tratamientos diferenciales a distintos tipos de tráficos al tiempo que se mantiene la sencillez y agilidad en los enrutadores de núcleo. (Cruz *et al.,* 2013)

Una red MPLS consta de un grupo de enrutadores interconectados controlados y gerenciados por una administración global en la que se distinguen dos zonas: el núcleo y la zona de borde. El núcleo consiste en el conjunto de nodos vecinos con capacidades MPLS solamente, mientras que el borde está conformado por los enrutadores que reciben el paquete IP desde otras redes no necesariamente MPLS y lo envían, etiquetado, al núcleo. (Cruz *et al.,* 2013)

Los enrutadores de borde son conocidos como Label Edge Routers (LER) y son los enrutadores de entrada, próximo a la fuente, y de salida, en el lado de destino del dominio MPLS. Los enrutadores internos al dominio MPLS son conocidos como Label Switch Routers (LSR). Para la transmisión, se genera una ruta entre los LERs correspondientes para reenviar paquetes etiquetados a través de los LSRs, conocida como Label Switched Path (LSP), a través de la cual fluyen los paquetes sobre la base de sus etiquetas en lugar de utilizar la estrategia tradicional de utilizar la información provista en los encabezados de los paquetes IP.

MPLS habilita a realizar Ingeniería de tráfico sobre redes IP (Awduche y Malcolm 1999). La característica principal que lo hace posible, es la de ruteo explícito. El ruteo explicito permite establecer los caminos (LSP) predefinidos para los paquetes. Esto se realiza desde los routers de la frontera de la red. MPLS retoma en este sentido las bases sobre las que se diseñó ATM, al establecer caminos virtuales para los flujos. (Belzarena, 2003) La ruta, por la cual el paquete se envía es asignada sólo una vez. Los enrutadores, a través del camino completo desde un origen a un destino, no toman decisiones de ruteo para ningún paquete, sino que utilizan una etiqueta insertada en el paquete como un índice dentro de una tabla que le indica cual es el próximo salto. Antes de enviar el paquete, el enrutador cambia la etiqueta que recibió en el paquete por la que será usada por el próximo vecino en el camino. (Cruz *et al.,* 2013)

MPLS se integra dentro de la tecnología IP, no requiriendo el despliegue, la operación y la gestión de una tecnología diferente como era el caso de IP sobre ATM. Al realizar ingeniería de tráfico en MPLS se genera la posibilidad de usar esta arquitectura para asegurar Calidad de Servicio. Las funciones de ruteo explícito de MPLS permiten enviar los paquetes por una ruta preestablecida o que se obtenga la misma analizando la carga de la red. De esta forma se puede, por ejemplo, enviar los agregados de flujo con fuertes requerimientos de calidad de servicio por rutas específicas descongestionadas. (Belzarena, 2003)

En los últimos años, MPLS ha obtenido una reputación notable debido a su soporte para técnicas de conmutación mixtas, ofreciendo varios parámetros de Calidad de Servicio (QoS) sin comprometer los recursos de la red. El enrutamiento eficiente es el componente clave de la gestión

de la ingeniería de tráfico en redes MPLS, que satisface la QoS con una gestión eficaz de los recursos. Las redes MPLS en lugar de usar datagramas IP tradicionales, usan paquetes etiquetados dentro del dominio de la red evitando la búsqueda compleja en las tablas de enrutamiento (Masood *et al.,* 2018).

MPLS puede optimizar los recursos de la red y proporcionar un tratamiento de calidad de servicio al tráfico, por lo que se ha convertido en el estándar de facto para la infraestructura de red central. Es escalable, está orientado a la conexión e independiente de cualquier tecnología de transporte de reenvío de paquetes. También reduce la búsqueda de direcciones IP en cada enrutador y disminuye la latencia de la red. Mejora el reenvío de paquetes en una red y permite determinar la ruta específica y el tiempo que tomará un paquete de origen a destino, lo que no era posible en el reenvío IP tradicional (PremKumar y Saminadan, 2017). Una red MPLS puede decidir la mejor ruta para cada flujo, asignar múltiples servicios en la misma red y tratar cada tráfico según los requisitos de QoS (Ridwan *et al.,* 2019).

El mecanismo MPLS puede canalizar múltiples tipos de tráfico a través de la red central. El túnel es la ruta por donde fluye el tráfico en la red central MPLS. La tunelización es una herramienta poderosa porque solo los enrutadores de entrada y salida necesitan conocer el contenido del tráfico transportado a través del túnel. Los detalles están ocultos a los enrutadores en el núcleo. Con el uso de túneles MPLS, el tráfico se puede enrutar explícitamente siguiendo las políticas de tráfico especificas (Ridwan *et al.,* 2019)

Uno de los principales tópicos de investigación y desarrollo en el área de Ingeniería de Trafico en MPLS es ruteo basado en restricciones y el reparto de carga. El ruteo basado en restricciones (Constraint based routing, CBR) busca caminos entre puntos de la red que satisfagan un conjunto de restricciones explícitas. Sin embargo, se ha observado, que el ruteo basado en restricciones para casi cualquier problema real es un problema NP-completo. Por esta razón se han propuesto múltiples algoritmos heurísticos sub-óptimos para realizar CBR (Kuipers, *et al.*, 2002). Por su parte, el balanceo de carga (load balancing) plantea el problema de dividir el tráfico de un agregado de flujos entre diversos caminos basados en algún criterio de optimalidad de la red. (Belzarena, 2003)

La distribución de la demanda de tráfico sobre la topología de la red puede ser eficientemente controlada en una forma que optimice la utilización y el desempeño a la vez que se satisfacen requerimientos de calidad de servicio, topología de la red y restricciones administrativas o de recursos (Cruz *et al.,* 2013). Sin embargo, el problema de la asignación de recursos, balanceo de cargas y cumplimientos de restricciones, no son sencillos de abordar, y no existen métodos determinísticos que aporten una solución en tiempos razonables debido a su extrema complejidad.

Las heurísticas son algoritmos de optimización estocástica que se utilizan para resolver problemas de alta complejidad computacional. En este tipo de problemas, más fáciles de enunciar que de resolver, los algoritmos exactos pueden tener bajo rendimiento, cuando no incapacidad, para proporcionar soluciones en tiempos aceptables. Los algoritmos que implementan heurísticas proveen soluciones de alta calidad, aunque no necesariamente la óptima, con un costo computacional razonable. Las estrategias generales que se utilizan para guiar heurísticas, son denominadas metaheurísticas. “Los procedimientos metaheurísticos son una clase de métodos aproximados que están diseñados para resolver problemas de difícil optimización combinatoria, en los que los heurísticos clásicos no son efectivos” (Osman *et al.,* 1996).

En esta tesis se propone, mediante estas técnicas no determinísticas, resolver el problema de asignación de recursos frente a situaciones de múltiples requerimientos en una red multicaminos MPLS, donde el objetivo es minimizar el costo de ruteo, cumpliendo con restricciones de capacidad de los enlaces y cumpliendo con demandas específicas para una planificación de la red fuera de línea. Para ello se diseñan e implementan distintos algoritmos heurísticos bioinspirados los que se prueban sobre topologías de red de distintos tamaños.

## 1.2 Objetivos:

El objetivo general de esta tesis es resolver el problema de optimización de asignación de recursos en redes multiservicios mediante la aplicación de técnicas heurísticas y meta-heurísticas puras o combinadas.

Los objetivos particulares son los siguientes:

- Contribuir a completar una taxonomía sobre la base del análisis de diferentes estrategias heurísticas puestas a resolver problemas combinatorios en el ámbito de las redes de comunicaciones.

- Diseñar e implementar una batería de algoritmos que permitan resolver casos diferentes que respondan a modelado similar.

- Estudiar sobre la combinación de heurísticas y determinar un criterio de selección para cada tipo de problema considerado.

## 1.3 Motivación - Estado del Arte

En la actualidad, el tráfico de datos, especialmente voz y multimedia imponen la necesidad de un ancho de banda importante y con garantía de calidad de servicio (QoS). Estos requerimientos causan problemas de congestión en las redes, que sólo puede paliarse a través de un diseño cuidadoso y mediante la mejor utilización posible de los recursos disponibles.

Este diseño de administración de recursos conlleva a problemas generalmente NP-Duros, que si bien son simples de plantear en general no hay métodos directos para su resolución (Donoso, 2004).

Este tipo de problemas, no tienen algoritmos que puedan dar una solución en tiempos razonables, dado que el espacio de búsqueda asociado resulta excesivamente grande. Las heurísticas son técnicas que permiten la exploración parcial del espacio de búsqueda con algunos criterios que permiten intensificar dicha búsqueda en regiones más promisorias. Además, deben estar diseñadas para sortear uno de los obstáculos mayores en estos problemas de optimización: la trampa de los subóptimos y la incertidumbre sobre el óptimo global.

En la mayoría de los casos pueden obtenerse buenas soluciones, no obstante, existen otros en los que la solución óptima es desconocida. Los requerimientos concretos exigen la entrega de soluciones al problema. Incluso aunque no sean las mejores. Entonces las soluciones sub-óptimas tendrán su utilidad y serán entronizadas como un límite alcanzado y al que debe superarse. Este es el motor de algunas técnicas y también una manera de contrastar algoritmos cuando la verdadera solución es desconocida.

En las últimas décadas se ha incrementado el uso de heurísticas para resolver problemas de optimización en general y de optimización combinatoria en particular, en muchos ámbitos de aplicación.

Los Algoritmos Genéticos fueron presentados en por Holland (Holland, 1975) y consisten en estrategias que toman ideas de los mecanismos de la evolución natural e incluso utiliza alguna terminología proveniente de la biología. Así, una población evoluciona a través de operadores de cruzamiento y mutación y mecanismos de selección de los individuos más aptos como padres de la siguiente generación. El proceso termina cuando un criterio de parada es alcanzado.

El recocido simulado (*Simulated Annealing, SA*), cuyo nombre está inspirado en el proceso metalúrgico de recocido, es una técnica que simula calentamiento y enfriamiento controlado de un material para maximizar el tamaño de los cristales y minimizar sus defectos (Kirkpatrick, 1983)

La busqueda Tabú (Glover, 1989) es una estrategia que parte de una solución inicial y define una vecindad de dicha solución en función de un operador llamado movimiento (cuya definición es específica del problema a resolver). Posteriormente se modifica la vecindad como el resultado de la mantención de una historia selectiva de los estados encontrados durante la búsqueda. De ese conjunto se elige la mejor solución la cual es tomada como punto de arranque de la nueva iteración. El término tabú proviene de prohibir, durante un cierto número de operaciones, un movimiento que haya alterado la solución. De esta manera se evita la oscilación.

Los algoritmos de optimización basados en colonia de hormigas (*Ant Colony Optimization, ACO*) propuestos por Dorigo (1992) tienen como objetivo la búsqueda del mejor camino en un grafo basándose en el comportamiento de las hormigas cuando buscan un buen camino entre el hormiguero y una fuente de alimento.

Se han desarrollado y evolucionado otras heurísticas, Búsqueda Dispersa (SS del inglés Scatter Search), Algoritmos de aprendizaje incremental basado en poblaciones (PBIL del inglés population-based incremental learning), algoritmos de vecindad variable, Algoritmos Diferenciales, etc. En los últimos años, los investigadores han hecho notar un aspecto importante: las heurísticas trabajan bien para encontrar buenas soluciones en tiempos razonables, pero la combinación de dos o más de ellas resultan en un mejor desempeño del algoritmo. Por esta razón han surgido, y recibida cada vez más atención, una nueva clase de métodos, llamados metaheurísticas híbridas las cuales son más eficientes y flexibles (Chen, 2010).

El término metaheurística se utiliza para definir métodos que pueden ser aplicados a un conjunto de problemas diferentes. Existe una concepción de verla como un marco algorítmico general que puede ser aplicado a varios tipos de problema diferentes conservando su estructura y con relativamente pocas modificaciones para cada uno de ellos. Estas modificaciones no necesariamente son fáciles de implementar, lo cierto es que los algoritmos evolutivos, la búsqueda tabú, simulado recocido, búsqueda local, o colonia de hormigas son ejemplos claros de lo antedicho.

Uno de los más clásicos esquemas de hibridación fue el uso de mecanismos de búsqueda local dentro de métodos basados en poblaciones. De hecho, ya no se concibe un algoritmo evolutivo sin alguna forma de mejora por búsqueda local de determinados tipos de individuos, para aumentar la calidad de las soluciones encontradas. Esto no es casual: la fortaleza de los métodos basados en poblaciones es la capacidad de exploración, mientras que la importancia de los mecanismos de búsqueda local, reside en su capacidad de encontrar rápidamente una mejor

solución dentro de una vecindad de la misma definida con algún criterio. En definitiva, el primer método explora las regiones más promisorias del espacio de búsqueda, mientras que el segundo explota una vecindad para obtener las mejores soluciones. El balance entre explotación y exploración favorece al éxito de este tipo de hibridación. En el campo de los algoritmos evolutivos el mecanismo explicado corresponde a lo que se denomina Algoritmos Meméticos (Krasnogor 2005).

Walshaw (2004) aplica la técnica de Refinamiento Multinivel para resolver problemas de optimización combinatorios. EL paradigma multinivel es un esquema de optimización propuesto para resolver el problema de repartición de grafos y posteriormente adaptado a otros problemas de optimización. Por otra parte, Montemanni y Smith (2009) proponen para resolver el problema de asignación de canales en redes celulares, un mecanismo basado en búsqueda tabú a la cual se le incorpora una estrategia de manipulación heurística que relaciona la reducción del espacio de búsqueda con las restricciones incorporadas al problema.

La aparición del vídeo-bajo-demanda (VoD) ha despertado un gran interés, aunque inevitablemente, la comparación con el vídeo convencional hace que las expectativas de los clientes sean muy altas. Todo esto, unido al gran consumo de recursos computacionales y de red que caracteriza a esta tecnología, obliga a los proveedores a prestar especial atención en la configuración de sus servicios (Melendi, *et al.,* 2003). Por su parte, la necesidad de contar con redes de Voz sobre IP (VoIP) capaces de cursar llamadas entre usuarios (extremo a extremo) con el mismo nivel de calidad de servicio que las redes de telefonía actuales, provoca también la necesidad de poner atención en la planificación eficiente de una red de datos. Estos son solo algunos ejemplos donde la optimización de redes multiservicios se torna importante.

En el ámbito de las redes MPLS, se han propuesto soluciones a problemas de optimización que resultan de interés. Algunas soluciones consideraron el monitoreo de la red, mediante la toma de muestras del tráfico en determinados puntos de la misma y, con esas muestras alimentar un bloque de optimización cuya salida afecta la configuración de los enrutadores (Cassetti, 2003). Los algoritmos genéticos han sido empleados para buscar una topología óptima en redes MPLS, empleando un modelo de capas y descomponiendo el problema en un conjunto de subproblemas, cada uno de los cuales se resuelven con una heurística basada en algoritmos genéticos (El-Alfy 2006).

En Dale (2007) se presenta la utilización de técnicas basadas en redes neuronales para resolver un problema de optimización de flujos de tráfico, utilizando MPLS. Intenta atacar el problema de encontrar la mejor forma en la cual distribuir el tráfico sobre los elementos disponibles de la red de manera tal que se mejore su desempeño.

En cuanto a la planificación de los LSPs, existen dos tipos de diseño de esquemas posible para realizar un planeamiento óptimo de una red MPLS: en línea y fuera de línea. La ventaja del diseño fuera de línea es que es posible planificar una red global óptima. Se han reportado trabajos en ambos tipos de diseño. En Sarsembagieva, (2012) se utiliza un método de planificación de rutas con la posibilidad de uso en línea y con resultados satisfactorios. Para el caso de planeamientos óptimos fuera de línea han sido utilizados algoritmos evolutivos como motores de optimización para alojar ancho de banda en una red MPLS, (Sylwester, 2008). Lemeshko, (2011) ha propuesto un modelo matemático para la administración del tráfico en una red MPLS con ingeniería de tráfico permitiendo la escalabiliad de las soluciones a través de un método predictivo. Casellas, *et al.*, (2002). utilizo modelos basados en el ancho de banda efectivo para selección de caminos y se han propuesto algoritmos para balancear carga en línea en una red MPLS (Elwalid

*et al.*, 2001) (Belzarena,2003). En Cruz *et al.,* (2013) se aplicó una estrategia basada en Búsqueda Tabú para resolver el problema de un multicamino restringido de mínimo costo en redes MPLS con buenos resultados en cuanto a la calidad de las soluciones. El-Sayed *et al.,* (2013) aplicaron técnicas evolutivas para minimizar los costos de enrutamiento y optimizar el equilibrio de carga considerando varias restricciones, incluidas las capacidades de enlace, la longitud de la ruta, el número de divisiones de tráfico para cada demanda y el número total de rutas. Onety *et al.,* (2013) presentaron un algoritmo genético para la optimización de múltiples índices de calidad de servicio de redes MPLS. Masood *et al.,* (2018), propusieron un modelo de optimización para redes MPLS e implementaron un algoritmo inspirado en la ecolocalización de delfines (DEA) para el cálculo de la ruta óptima. Un año después, el mismo equipo propuso una versión novedosa, con modificaciones en la estrategia de aprendizaje elitista (ELS), en optimización por enjambre de partículas (PSO) que no sólo resuelve el problema de exploración existente en PSO, sino que también produce soluciones óptimas con tasas de convergencia eficientes para diferentes escalas de red MPLS / GMPLS (Masood *et al.* 2019).

Esta tesis se centrará en la planificación fuera de línea del reparto de cargas entre diversos caminos (LSPs) en una red MPLS, para cuya optimización se utilizarán distintos algoritmos bioinspirados, los cuales han sido utilizados para resolver diversos problemas en el ámbito de las redes de datos, lo que motivó su elección. Mas detalles del estado del arte de estas técnicas se citarán oportunamente en el capítulo 4, al momento de analizar cada estrategia de manera particular.

## 1.4 Temas de Investigación:

Se analizarán e implementarán diferentes estrategias inspiradas en la naturaleza para para dar solución al problema de distribución de flujos sobre la red multicamino con el fin de lograr la optimización del tráfico sobre distintas topologías de redes.

Estas estrategias considerarán la selección, utilización, adaptación y/o desarrollo de modelos, métodos y algoritmos que permitan mejorar la calidad de las soluciones y minimizar los tiempos de cómputo asociados. En el desarrollo propuesto se seleccionarán y combinarán diferentes metaheurísticas. La hipótesis de trabajo es que es posible mejorar la calidad de las soluciones y/o los tiempos de cómputo asociados a través de la selección y aplicación de estrategias puras o combinadas.

Parte de la investigación asociada al trabajo, consiste en determinar la aplicabilidad de las diferentes técnicas a emplear, el diseño de la solución algorítmica y la determinación de sus parámetros de configuración para obtener soluciones en tiempos razonables. Esto significa que deberá analizarse la complejidad del problema y su tamaño, el modelado algorítmico del mismo y el diseño de la estrategia de la solución que posea una escalabilidad aceptable y entregue soluciones en un tiempo acotado.

## 1.5. Alcances y límites

Este trabajo se enmarca en el contexto académico como requisito para alcanzar el grado de Magister en Redes de Datos, realizando un aporte teórico sobre distintas técnicas algorítmicas bioinspiradas aplicada a las redes de datos.

Se realiza el diseño e implementación sobre un intérprete de cinco prototipos de algoritmos propuestos y la experimentación son su aplicación numérica sobre topologías de red de cuatro niveles de dificultad. Para cada caso se determinarán los parámetros de configuración para el buen desempeño de cada uno de los algoritmos. Con estas cuatro topologías, se obtendrán los resultados, se evaluará la eficiencia y resultados estadísticos de los algoritmos, a modo de determinar las mejores opciones frente a escenarios similares.

Se excluye de esta obra y se propondrán como trabajos futuros, la evaluación de otras técnicas, la codificación de los algoritmos en un lenguaje compilable y paralelizable con el fin de lograr mejores tiempos de ejecución, como así también el establecimiento de los mecanismos e interfaces necesarias para aplicar las soluciones entregadas por los algoritmos a dispositivos intermediarios reales o simulados.

## 1.6 Estructura del documento

Además de este capítulo introductorio, el documento cuenta con cinco capítulos que exponen el desarrollo de la tesis.

En el capítulo 2 se dedica al entendimiento del mecanismo de transporte de datos estándar creado por la IETF, la conmutación de etiquetas multiprotocolo o MPLS. Se describe su historia evolución y usos actuales, sus características y concepto funcional, la operación y aplicación como así también las ventajas y desventajas.

En el capítulo 3 se realiza la definición del problema y su formulación matemática. Se describe la representación de la red, la representación de la solución, y un análisis de la complejidad de determinar buenas soluciones.

El capítulo 4 recopila la investigación realizada en torno a las técnicas heurísticas, la definición de las heurísticas a utilizar y el diseño de los algoritmos propuestos como contribución a la taxonomía de estrategias a utilizar para la resolución del problema de asignación de demandas.

El capítulo 5 expone la experimentación y resultados de los distintos algoritmos, los cuales se ensayan para 4 instancias del problema de diferentes tamaños. Se expone el funcionamiento de cada algoritmo para cada caso y una comparación que aporta a la selección del mejor diseño.

Finalmente, el capítulo 6 expone los trabajos a futuro y las conclusiones de la investigación y desarrollo.

# Capítulo 2
# Multiprotocol Label Switching (MPLS)

## 2.1 Introducción

Como sabemos las redes de todo el mundo que forman Internet han sufrido, y sufren en la actualidad, un crecimiento vertiginoso fundamentalmente en el número de usuarios que la utilizan, y por consiguiente un aumento del tráfico de datos debido a los servicios que se van agregando a las redes convergentes, como ser VoIP, TV sobre Internet, radio sobre Internet, Video Streamming multi-punto, etc. Esto trae como como consecuencia la demanda cada vez mayor de nuevos y mejores servicios que Internet tiene que ofrecer, lo que obliga a plantearse la implementación de nuevas tecnologías que permiten ofrecer a los usuarios mayores anchos de banda.

Internet como red pública debió incorporar los mecanismos necesarios para asegurar, no únicamente el transporte de datos mediante la filosofía actual del best-effort, sino también el soporte y transporte eficiente de los datos de las distintas aplicaciones y servicios críticos que muchos usuarios necesitan y, por lo tanto, surge la necesidad de los ISPs de poseer sofisticadas herramientas de gestión de redes para garantizar el correcto funcionamiento, y lograr un uso óptimo de los recursos y la determinación correcta de por dónde dirigir el tráfico de cada demanda de manera de no comprometer ninguna porción de la red.

El típico algoritmo de enrutamiento utilizado por los IGPs que determina la “ruta más corta” produce el congestionamiento de ciertos enlaces, mientras existen otras rutas disponibles que no son utilizadas. Este tipo de enrutamiento provoca demoras impredecibles y pérdida de datos. Sin embargo, no ha sido un problema para las aplicaciones tradicionales de Internet como web, correo electrónico, transferencia de archivos y similares, pero la nueva generación de aplicaciones que incluyen audio y video streaming, exigen alto rendimiento, ancho de banda y baja latencia. (Hernández Camacho, 2015)

En definitiva, necesitamos incrementar la eficiencia de la utilización de los recursos de la red, al mismo tiempo que debemos minimizar la posibilidad de congestión de los enlaces.

Claramente, la congestión es un fenómeno nada deseable y es causada por ejemplo por la insuficiencia de recursos en la red. En casos de congestión de algunos enlaces, el problema se resolvía añadiendo capacidad a los enlaces. La otra causa de congestión es la utilización ineficiente de los recursos debido al mapeado del tráfico. El objetivo básico de la Ingeniería de Tráfico es adaptar los flujos de tráfico a los recursos físicos de la red. La idea es equilibrar de forma óptima la utilización de esos recursos, de manera que no haya algunos que estén sobreutilizados, creando cuellos de botella, mientras otros puedan estar subutilizados. (Delfino, *et al.,* 2006)

Existen varios mecanismos para optimizar el rendimiento, modelado, medición, caracterización y el control de tráfico en una red para obtener objetivos específicos de rendimiento y ofrecer servicios competitivos a los clientes. Uno de estos mecanismos de transporte de datos estándar, creado por la IETF y definido en el RFC 3031, se la conoce como la conmutación de

etiquetas multiprotocolo comúnmente denominada MPLS por la sigla en inglés de Multiprotocol Label Switching.

MPLS constituye un elemento clave en el despliegue de técnicas de Ingeniería de Trafico ya que la idea básica de MPLS es separar completamente el plano de control (enrutamiento) del plano de datos (reenvío de paquetes) mientras mantiene la compatibilidad con las infraestructuras de red IP existentes. (Hernández Camacho, 2015)

Podría decirse que es un método de reenvío de tráfico de alta performance basado en el enrutamiento IP. En la implementación de este mecanismo, los dispositivos de borde de la red aplican al paquete IP, una "etiqueta" sobre la base de la cual los dispositivos realizarán el reenvío de tráfico a través de la ruta correspondiente con una carga mínima de procesamiento.

Esta "etiqueta" no es más que una cabecera adicional que contienen la información necesaria para la toma de decisiones de envío. En consecuencia, en los routers habilitados con MPLS el tráfico no se reenvía basado en el algoritmo tradicional de la "ruta más corta" sino que los paquetes son agrupados por clases o FEC (Forwarding Equivalence Class) en el momento en que entran en el dominio MPLS y basado en esta clasificación son asignados a un LSP (Label Switched Path) y reenviados al router de salida. De esta manera el administrador de red tiene un control total sobre la clasificación de paquetes y el establecimiento de rutas. (Hernández Camacho, 2015)

Para entender la potencialidad de MPLS, podemos decir que integra la performance de administración de tráfico de capa 2 con la escalabilidad y flexibilidad del enrutamiento de capa 3 (Gerometa, 2012), o como lo expresa Hernández Camacho, (2015) combina la inteligencia del routing con la velocidad del switching.

En este capítulo se describen los principios básicos de funcionamiento de MPLS con el fin de entender cómo se realizará luego la optimización de su funcionamiento mediante técnicas heurísticas.

## 2.2 MPLS: su historia, presente y futuro

### 2.2.1 Sus predecesores

Para entender el origen de MPLS se considera interesante entender a sus antecesores lo que se resume a continuación.

Los años setenta fueron testigos del nacimiento de las redes privadas, que permitían a las empresas interconectar sus sedes principales mediante líneas alquiladas independientes para voz y datos, con anchos de banda fijos. Pero la demanda de transmisión de datos a mayores velocidades y rendimientos surgió en la década de los 80, propiciada por el uso de arquitecturas cliente/servidor, así como el desarrollo de nuevas aplicaciones. De este modo, aparecieron nuevos patrones de tráfico, donde el ancho de banda podía permanecer ocioso durante prolongados períodos de tiempo. También en esta época comenzó a detectarse la necesidad de interconectar las distintas redes de área local (LAN) que empezaban a surgir en las organizaciones.

A finales de los 80 y principios de los 90, Frame Relay comenzó a ganar una creciente aceptación, dado que ofrecía capacidad de procesamiento de datos más elevada que X.25, su

tecnología predecesora. Estas mejoras se consiguieron mediante la implementación de un sistema de procesamiento de paquetes simplificado que dividía la información en tramas, cada una de las cuales transportaba una dirección utilizada por los conmutadores para determinar su destino final. El sistema aumentaba la eficiencia en la utilización de los recursos, dado que permitía fragmentar el tráfico en ráfagas y así aprovechar el ancho de banda que antes permanecía ocioso, y reducía significativamente los costes de transmisión frente a los de las líneas alquiladas.

A mediados los noventa, la dependencia de las empresas respecto de sus redes aumentó aún más con la creciente utilización del correo electrónico y la implantación de aplicaciones consumidores de grandes anchos de banda, como, por ejemplo, los sistemas de planificación de recursos empresariales o ERPs (Enterprise resource planning). Esta dependencia no ha hecho desde entonces más que crecer ante la expansión cada vez más generalizada del uso de Internet y de las transacciones de comercio electrónico business-to-business.

Esta expansión, junto a la aparición del concepto de convergencia de las redes de voz y datos en una sola plataforma de networking, condujo al desarrollo de ATM (Asynchronous Transfer Mode), que fue concebida como una tecnología multiservicio de banda ancha capaz de soportar voz, datos, imágenes y vídeo sobre una misma infraestructura de red.

A diferencia de Frame Relay, en la que el tamaño de los paquetes era variable, ATM se basa en conmutación de celdas de tamaño fijo. Esta característica permitía aprovechar todas las ventajas de la multiplexación estadística y ofrecía un rendimiento determinístico. Las conexiones ATM están típicamente basadas en circuitos virtuales permanentes (PVC) con Calidad de Servicio (QoS), capaces de proporcionar transmisiones de extremo a extremo garantizadas y fiables.

Poco después, la red vuelve a actuar como elemento catalizador de un nuevo cambio. Con el creciente uso de Internet para las comunicaciones business-to-business, surgió la necesidad de garantizar mayores niveles de seguridad sobre esta infraestructura, pública y completamente carente de regulación. Así, nació el protocolo IPSec.

IPSec hace posible la creación de “túneles” seguros entre dos gateways, típicamente un router, firewall o, incluso, software sobre un PC conectado a la red privada del usuario, a través de redes públicas. Los túneles IPSec son establecidos dinámicamente y liberados cuando no están en uso. Para establecer un túnel, los dos gateways IPSec han de autenticarse entre sí y definir cuáles serán los algoritmos de seguridad y las claves que utilizarán. Así, IPSec proporciona comunicaciones seguras y la separación lógica entre los flujos del tráfico de la red privada virtual (VPN) frente al resto de las transmisiones que cursan la red IP compartida.

La más reciente tecnología en este campo desemboca hoy en MPLS, que aporta diversas mejoras sobre IPSec. A diferencia de este protocolo, MPLS trabaja enviando los paquetes junto con las instrucciones para su tratamiento contenidas en etiquetas en lugar de en direcciones. De una manera sucinta, se podría decir que preprocesa todas las decisiones de routing y asigna una etiqueta que “dice” al conmutador o router dónde ha de enviar un paquete a partir de la información contenida en ese mismo paquete. La clave de una etiqueta MPLS es que informa al dispositivo no sólo de dónde debe dirigir los paquetes, sino también de cómo hacerlo. Todos los atributos necesarios para la optimización de la VPN eficiente y seguro están codificados en la etiqueta, incluida la clase de servicio que ha de aplicarse a un determinado flujo.

Este método de funcionamiento evita la necesidad de establecer y mantener circuitos virtuales permanentes, algo que –junto con la aplicación de técnicas de priorización de tráfico–

hace de MPLS una solución ideal para crear redes VPN IP completamente malladas. (Network World, 2003)

### 2.2.2 La evolución de MPLS

MPLS se comenzó a utilizar a mediados de la década del 90 como una tecnología que acompañaba el crecimiento de internet, persiguiendo básicamente dos modestos objetivos de diseño. Por un lado, integración entre el modo de transferencia asíncrono (ATM, del inglés Asynchronous Transfer Mode) y el protocolo IP, logrando un solo plano de control basado en IP que abarcara tanto los switches ATM como los routers IP. En segundo lugar, se pretendía incrementar el plano de control IP con algunas funcionalidades adicionales como la ingeniería de tráfico con encaminamiento basado en restricciones que ya estaba presente en el plano de control ATM. (Minei y Lucek, 2011)

Poco tiempo después, el uso de MPLS se extendió a aplicaciones como (CCC del inglés Circuit Cross Connect), servicios ATM and Frame Relay sobre infraestructura IP/MPLS, BGP/MPLS VPNs, y luego en servicios sobre redes privadas virtuales (VPLS), evolucionando el original encaminamiento basado en restricciones más allá de la ingeniería de tráfico para aplicaciones como el redireccionamiento de servicios de tráfico diferenciados (DiffServ-TE).

La idea de un simple plano de control para sitches ATM y routers IP, ha evolucionado en un MPLS generalizado (GMPLS) que provee un plano de control simple, no solo para las tecnologías mencionadas, sino también para SONET/SDH y conexiones ópticas cruzadas.

Es importante tener en mente que, en todas las aplicaciones mencionadas, MPLS es solo uno de los componentes de dichas aplicaciones, aunque uno crítico. (Minei y Lucek, 2011)

Los objetivos originales en la creación de MPLS de alguna manera fueron mutando durante su uso, primeramente, porque la mayoría de las aplicaciones MPLS que tenemos en la actualidad, no fueron concebidas en el diseño original de MPLS. Además, mientras el diseño original se preocupaba por una mejor integración entre IP y ATM fundamentalmente, lo que MPLS significa hoy en día, es totalmente diferente. En lugar de ofrecer un plano de control simple para abarcar los switch ATM y los routers IP, en la actualidad, tiene un doble significado de ser capaz de ofrecer el servicio de ATM a través de una red IP / MPLS que no tiene ningún tipo de switch ATM, tan bien, como conectar switches ATM sobre esas infraestructuras. Del mismo modo, mientras que MPLS fue concebido como una tecnología para los proveedores de servicios, hoy en día ha penetrado también en el plano empresarial.

La Conmutación de etiquetas multiprotocolo (MPLS) ha evolucionado de una tecnología exótica a una herramienta convencional utilizada por los proveedores de servicios en muy poco tiempo. Hubo un despliegue rápido de Servicios habilitados para MPLS y desarrollos de nuevos mecanismos y aplicaciones para MPLS en los organismos de normalización. (Minei y Lucek, 2011)

La primera publicación del IETF (Internet Engineering Task Force) del grupo de trabajo de MPLS se realizó en abril de 1997, y planteaba principalmente cuatro objetivos (Minei y Lucek, 2011):

1. **La escalabilidad del ruteo de la capa de red:** Utilizando etiquetas como medio de agregar información necesaria para los reenvíos de información, mientras se trabaja en presencia de las jerarquías de enrutamiento.

2. **Mayor flexibilidad en la entrega de servicios de enrutamiento:** Utilizando etiquetas para identificar cierto tráfico que necesita servicios especiales, por ejemplo, QoS, usando etiquetas para especificar una determinada ruta específica distinta al del resto de los paquetes. (dos propiedades utilizadas en Diffserv Aware Traffic Engineering).

3. **Incrementar el rendimiento:** Utilizando el paradigma de intercambio de etiquetas para optimizar el rendimiento de la red.

4. **Simplificar la integración de routers con las tecnologías basadas en la conmutación de células**: Haciendo que los conmutadores de células se comporten como pares de los routers, reduciendo así el número de pares de enrutamiento que el router tiene que mantener. Teniendo información sobre la topología física disponible en los procedimientos de ruteo de la capa de red y empleando direccionamiento de ruteo común y procedimientos de gestión.

### 2.2.3 Perspectiva actual de MPLS

Tal como lo describe (Minei y Lucek, 2011) cuando se plantearon los objetivos de MPLS, la mayoría de las redes tenían un núcleo formado por switches ATM rodeados por routers. Estos routers eran típicamente conectados full mesh con las conexiones ATM. Este modelo de superposición resultaba difícil de escalar debido a que las adyacencias de los routers crecían con el cuadrado de los routers involucrados, de ahí que surge la necesidad que los switches ATM actuaran como pares de los routers.

Es interesante notar que en la actualidad la situación es la opuesta. La mayoría de las redes disponen de un núcleo basado en MPLS y los proveedores migran sus servicios ATM interconectando los switches ATM con conexiones de capa 2 sobre ese núcleo MPLS. Esto tiene el problema de que el número de adyacencias entre swtiches ATM crece con el cuadrado del número de conmutadores ATM en cuestión. Por lo tanto, actualmente el trabajo consiste en la fabricación de conmutadores ATM que se comporten de la misma manera que los routers (Walsh y Cherukuri, 2005). Esto es para evitar tener una malla completa de adyacencias entre conmutadores ATM en lugar de evitar que tenga una malla completa de adyacencias entre los routers, como se dijo en el enunciado del problema. En resumen, gran parte del enunciado del problema original sigue siendo relevante hoy en día.

MPLS beneficia en la actualidad principalmente a los servicios WAN y soluciones empresariales como VPNs (tanto de nivel 2 como de nivel 3) ya que garantiza una alta disponibilidad de la red, y es utilizado por muchos operadores como método con el que garantizar la calidad del servicio mediante la Ingeniería de tráfico.

Además, es una de las mejores alternativas para conectar con las entidades a nivel internacional al implementar las soluciones Cloud que permiten almacenar información y comunicarse con los compañeros de trabajo. MPLS Es un estándar que aparte de colaborar con una conexión a Internet rápida de nivel empresarial le permitirá desplegar las soluciones Cloud sin problemas, manteniendo el flujo de su trabajo fluido a la altura de los requerimientos existentes en el mercado. (Mira Telecomunicaciones, 2019)

Los operadores de grandes redes WAN (Wide Area Network) obtienen beneficio de las ofertas de proveedores basadas en MPLS porque, configuradas correctamente, las Label Switched Paths optimizan el tráfico de datos y garantizan que todos los usuarios dispongan

siempre del ancho de banda que necesitan sin un gran coste. El método también constituye una solución adecuada para redes universitarias internas o para redes corporativas, siempre y cuando se cuente con el presupuesto suficiente. (www.ionos.es 2017)

La entrega confiable de paquetes y la excelente calidad de servicios que ofrece MPLS, es especialmente esencial para mantener la calidad de los protocolos en tiempo real que demandan los consumidores de hoy en día que están cada vez más interesados en contenido multimedia que requieren de ancho de banda como videos, realidad aumentada y virtual, etc. Además de garantiza las demandas especialmente de empresas o instituciones que tienen requisitos de conectividad específicos.

## 2.3 Características básicas de las redes MPLS

### 2.3.1 Concepto funcional, términos y dispositivos de una red MPLS

A diferencia del reenvío tradicional a nivel IP donde los routers leen la cabecera de capa 3 en busca de la dirección de IP de la red de destino y mapea dicha dirección en su tabla de ruteo para seleccionar y redireccionar el paquete a la ruta correspondiente, en MPLS los paquetes son reenviado basándose en etiquetas que son generadas por los mismos dispositivos que integran la red, disminuyendo notablemente el procesamiento.

Como las etiquetas tienen significado exclusivamente para el router que las genera, una ruta es definida como una secuencia de etiquetas y recibe el nombre de Label Switched Path o LSP (Gerometa, 2012). Solamente realizarán una búsqueda en la tabla de enrutamiento aquellos dispositivos que se encuentran en los bordes de la red MPLS, los demás solo actuaran mediante el análisis de etiquetas.

Las etiquetas MPLS suelen identificar una red de destino, pero también pueden identificar un nivel de calidad de servicio, una dirección de origen o un circuito de capa 2.

Los dispositivos intermediarios que soportan la red MPLS se los conoce como **LSR** (Label Switched Router), **Intermediate LSR** o **P** (Provider) y son los encargados de recibir y transmitir paquetes en función de las etiquetas, pueden agregar quitar o cambiar las etiquetas de los mismos.

Los dispositivos que operan en el borde de un dominio MPLS, es decir los puntos de entrada y salida de la red MPLS, se lo conoce como **Edge LSR** o **PE (provider EDGE) o** Enrutadores de borde de Etiqueta (**LER)** por su sigla en inglés Label Edge Router y será el encargado de insertar o remover las etiquetas dependiendo si el paquete está entrando o saliendo del dominio.

Como se expresó anteriormente la ruta etiquetada para un paquete de datos, desde un origen hasta su destino a través de la red MPLS se conoce como LSP (Label Switches Path), el primer dispositivo de la ruta es conocido como dispositivo de ingreso y análogamente el último será el dispositivo de egreso. Las rutas LSP son unidireccionales por lo que se requieren 2 LSPs para sostener una comunicación bidireccional.

En la siguiente figura puede observarse una representación del dominio MPLS completo.

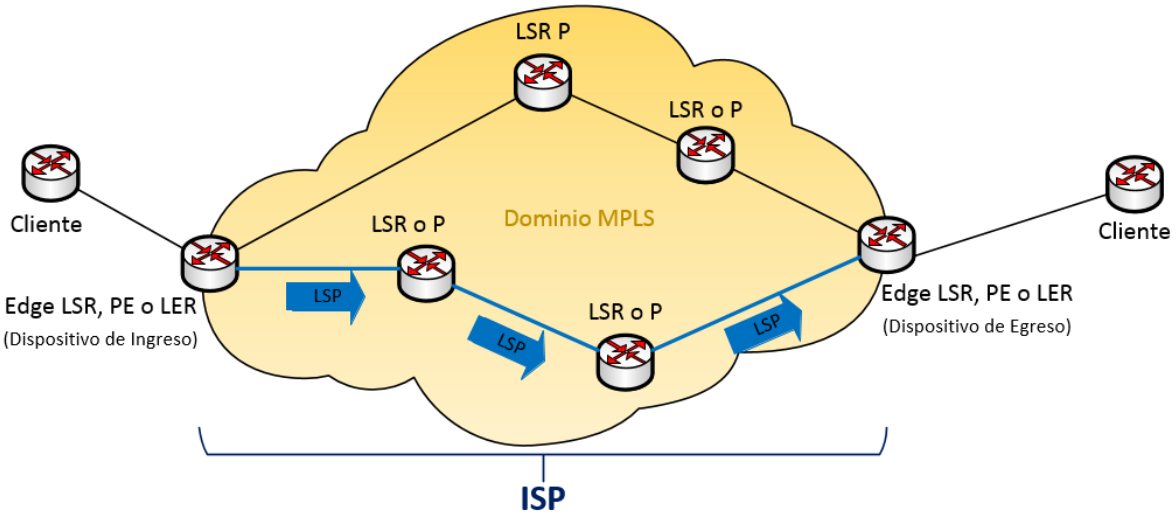


*Figura 2.1: Representación del domino MPLS*

Cuando un grupo de paquetes circulan por la red MPLS y son reenviados de la misma forma, sobre la misma ruta y con el mismo tratamiento poseerán el mismo **FEC – Forwarding Equivalence Class.** Es decir, FEC es un término usado en MPLS para describir un conjunto de paquetes con las mismas características y por consiguiente deberán reenviarse de la misma manera; es decir, pueden ser enlazados a la misma etiqueta MPLS, y todos seguirán el mismo camino. En tal sentido, el reenvío de paquetes MPLS es considerado como orientado a la conexión a diferencia del reenvío IP. (Gerometa, 2012)

MPLS usa un esquema de etiquetado de tráfico, marcándolo en la entrada de la red. Es usado únicamente en los routers y es independiente del protocolo usado, lo que le permite ser utilizado sobre otros protocolos distintos a IP. Además, soporta varias tecnologías de acceso que incluyen T1/E1, ATM, Frame Relay, DSL. Los protocolos de enrutamiento de nivel 3 como OSPF o IS-IS se usan únicamente para funciones de control, ya que las decisiones de enrutamiento se toman en función de la etiqueta MPLS y no de la cabecera IP. MPLS mejora la escalabilidad de la red, reduciendo las tablas de enrutamiento y el retardo de proceso en los routers (Naveed y Kumar 2014), combinando algunas prestaciones de las redes orientadas a conexión con la de las redes sin conexión. Así, un router asigna una etiqueta a cada una de las entradas de la tabla de enrutamiento y las distribuye a sus routers vecinos. Luego, cuando se pasan paquetes entre ellos, los routers solo tienen que leer la etiqueta MPLS para identificar el siguiente salto donde enviar el paquete. De esta forma los paquetes “fluyen” de un extremo a otro de la red y se consigue un enrutamiento a mayor velocidad a la vez que se disminuye el retardo y el jitter.

### 2.3.2 MPLS, Modelo OSI y Etiquetas

Como es sabido, el modelo de referencia de Interconexión de Sistemas Abiertos (OSI) diseñado por La Organización Internacional para la Estandarización (ISO) es el modelo más conocido y utilizado desde la década del 80 y consta de siete capas o niveles para identificar las actividades de la red.

El avance y evolución de las tecnologías utilizadas ha proporcionado ciertos protocolos o mecanismos que no encajan directamente en dicho modelo. Este es el caso de MPLS donde

algunos autores (Felici y Montañana 2009) lo consideran como tecnología de la capa 2.5, porque realiza un encapsulado intermedio entre la capa de enlace (capa 2) y la capa de red (capa 3).

| MPLS y Modelo OSI | | |
|---|---|---|
| **Capa 5 a 7** | Capas superiores | |
| **Capa 4** | Nivel de transporte | TCP / UDP |
| **Capa 3** | Nivel de red | IPv4 / IPv6 |
| **Capa 2.5** | **MPLS** | |
| **Capa 2** | Nivel de Enlace | PPP-Ethernet- HDLC ATM - Frame Relay |
| **Capa 1** | Nivel físico | Señales ópticas / electrícas |

*Figura 2.2: MPLS en el modelo OSI*

Cuando un usuario final envía tráfico a la red MPLS, se agrega una etiqueta mediante un enrutador de entrada MPLS (Edge-LSR) que se encuentra en el borde de la red. El dispositivo verifica la tabla de ruteo, y determina la interfaz de salida, si esta opera con MPLS y existe una etiqueta para el próximo salto se inserta la etiqueta y se modifica el identificador del protocolo en el encabezado de capa 2 para indicar que es un paquete etiquetado y se envía el paquete al próximo salto.

Según las especificaciones del IETF, MPLS tiene que funcionar sobre cualquier tipo de protocolo de transporte de nivel 2: ATM, Frame Relay, LAN, PPP, etc. Por lo tanto, y para aprovechar las características de algunos de ellos, si el protocolo de transporte de datos ya dispone de campos para identificar las etiquetas (como por ejemplo pasa con los campos VPI/VCI (Virtual Path Indenifier / Virtual Channel Identifier) de ATM y DLCI de Frame Relay) pueden utilizarse o no estos campos nativos para las etiquetas. (Fuentes Torruella, 2011) En este caso se dice que MPLS trabaja en Cell-Mode, (Gerometa, 2012)

En cambio, si la tecnología de nivel 2 utilizada no dispone de un campo para etiquetas (por ejemplo, PPP o protocolos LAN), entonces hay que añadir obligatoriamente la cabecera genérica MPLS definida, que contiene un campo específico para la etiqueta. (Fuentes Torruella, 2011) En este caso, cuando el modo de operación consiste en conmutación de tramas, es conocido como MPLS Frame-Mode, (la gran mayoría). (Gerometa, 2012)

En este encapsulado se introduce una etiqueta de 4 bytes donde se identifica el destino, servicio y la FEC específica del paquete, lo que permite a los routers utilizar las técnicas de conmutación para determinar el próximo salto correspondiente a cada FEC.

El utilizar el etiquetado por debajo de capa 3, permite que MPLS pueda funcionar independientemente del protocolo utilizado en dicha capa, de ahí lo de “multiprotocol”. En otras palabras, MPLS puede usarse para crear tablas de reenvío para cualquier protocolo subyacente.

Esta arquitectura de etiquetado es flexible y permite anidar etiquetas, es decir, introducir una trama MPLS dentro de otra.

La cabecera MPLS, conocida como “Shim header”, se añade como mencionamos entre la cabecera de nivel 2 y la de nivel 3, tal y como puede observarse en la siguiente figura.

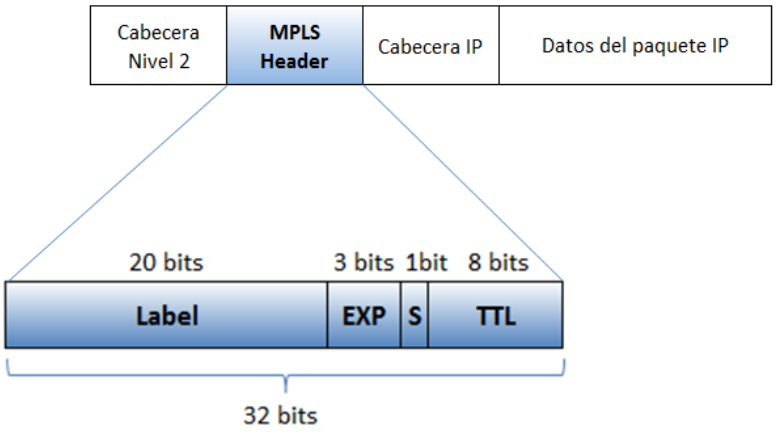


*Figura 2.3: Ubicación y formato de la cabecera MPLS*

La cabecera MPLS consta de cuatro subpartes:

**Etiqueta (Label) [20 bits]:** Es el identificador, el cual tiene sentido local por lo que puede cambiar en cada enlace y contiene la información de los enrutadores MPLS para determinar dónde debe reenviarse el paquete para alcanzar el destino, es decir es el valor que se utiliza como índice en la tabla de reenvío MPLS. Las etiquetas se utilizan en los routers para diferenciar entre los distintos FECs (Forward Equivalance Class).

**EXP (Experimental) [3 bits]:** se usan bits experimentales para Quality of Service (QoS) para establecer la prioridad que debe tener el paquete etiquetado. Algunos autores lo denominan CoS (Class of Service) o TC (Trafic class) y transmiten la clase de servicio que se aplicará al paquete. Por ejemplo, LSR y los LER pueden usar estos bits para determinar la cola en la que el paquete debe ser colocado.

**S (Stack) [1 bits]:** Permite apilar etiquetas en modo jerárquico, indica si existe más de una etiqueta, de modo que el nodo MPLS tratará siempre la que esté más alto en la pila. La parte inferior de la pila le dice al enrutador MPLS si es el último tramo del viaje y no hay más etiquetas que preocuparse. Se suelen utilizar más de una etiqueta por ejemplo en VPNs MPLS (2 etiquetas) Traffic Enginneering (2 o más) o por ejemplo cuando se combinan ambas.

**TTL (Time-To-Live) [8 bits]:** Tiene el mismo significado que en el protocolo IP, identificar cuántos saltos puede hacer el paquete antes de descartarlo.

### 2.3.3 Asignación y distribución de etiquetas

Para mapear las etiquetas en un LSP es necesario contar con protocolos de distribución de etiquetas para lo cual existen distintas alternativas como lo son:

- TDP (Tag Distribution Protocol) un protocolo propietario de Cisco y predecesor de LDP.
- LDP (Label Distribuion Protocol) que fue desarrollado por la IETF el que cuenta con algunas funcionalidades adicionales a TDP.
- RSVP (Resource Reservation Protocol) el cual es utilizado solamente para la implementación de Ingeniería de tráfico. (Gerometa, 2012)

- Protocolo de enrutamiento basado en restricciones LDP (CR-LDP) (Hernández Camacho, 2015).

- Multi-protocolo BGP (Hernández Camacho, 2015).

LDP es el protocolo estándar para el intercambio de etiquetas y enrutamiento implícito utilizado para configuración y establecimiento de los LSPs salto a salto o también llamados "control-driven LSPs" definido por la IETF.

El protocolo LDP funciona sobre TCP y usa la información de enrutamiento subyacente proporcionada por un IGP con el fin de enviar paquetes etiquetados. LDP asocia una FEC con cada camino LSP que se crea, y posteriormente intercambia y distribuye esta información de asociación de las etiquetas entre dos LSR vecinos. Esta asociación es bidireccional y permite que un LSR aprenda del otro. (Hernández Camacho, 2015)

Este protocolo envía mensajes de *hello* cada 5 segundos los que se envían por todas las interfaces que tienen habilitado MPLS. Los vecinos que reciben el *hello* y tienen habilitado MPLS en esa interfaz responden abriendo una sesión TCP puerto 646 y establecen una sesión LDP en formato unicast.

Cuando un dispositivo recibe en una interfaz un *hello* de otro dispositivo, intentará iniciar una sesión LDP con ese equipo conde el dispositivo con ID más alto (IP más alta) será el que inicie la sesión TCP, y una vez establecida la sesión ambos dispositivos seguirán enviando periódicamente *hellos.* (Gerometa, 2012)

Por otro lado, entre los protocolos de enrutamiento explícito más comunes encontramos al protocolo LDP de Ruta Restringida (CR-LDP) y al Protocolo de Reservación de Recursos con Ingeniería de Tráfico (RSVP-TE). El primero de estos protocolos es una extensión del protocolo LDP. CR-LDP incorpora características para poder realizar Ingeniería de Tráfico, como la capacidad de poder señalizar caminos explícitos.

RSVP-TE opera de manera similar que CR-LDP, pues permite negociar un LSP punto a punto que garantice un nivel de servicio de extremo a extremo. El protocolo es una extensión de la versión original RSVP (Hernández Camacho, 2015).

## 2.4 Operación de MPLS

La propiedad fundamental de una red MPLS es que puede ser utilizada para el transporte de múltiples tipos de tráficos a través de túneles, en los cuales solamente los routers al ingreso y egreso del túnel, necesitan entender el contexto del tráfico subyacente llevado en el túnel. Los routers en el núcleo de la red, solo necesitan cambiar los paquetes encapsulados en MPLS sin tener en cuenta contenido subyacente.

Además de las características generales, los túneles MPLS tienen las siguientes propiedades según (Minei y Lucek, 2011):

1. El tráfico puede ser explícitamente ruteados, dependiendo de qué protocolo de señalización se utiliza.

2. Se provee recursividad, por lo que pueden existir túneles dentro de túneles.

3. Hay protección contra Spoofing, debido a que el único lugar donde los datos se pueden inyectar en un túnel MPLS está en el extremo de ese túnel. En contraste en un túnel IP, los datos pueden ser inyectados desde cualquier fuente que tiene conectividad con la red que transporta el túnel.

4. La sobrecarga en la encapsulación general es relativamente baja (4 bytes por encabezado MPLS).

Una red MPLS se compone de dispositivos de borde conocidos como Label Edge Routers (LER) o Provider Edge Routers (PE) y routers centrales conocidas como Label Switching Routers (LSRs) o routers del proveedor (P). Una malla de túneles unidireccionales, llamados Caminos de conmutación de etiquetas (LSP del inglés Label Switched Paths) se construye entre las LER con el fin de que un paquete que entra en la red en el LER de entrada puede ser transportado al LER de salida apropiado. Cuando los paquetes entran en la red, el enrutador de entrada determina a qué FEC pertenecen los paquetes. (Minei y Lucek, 2011)

Para detallar el funcionamiento y tomando la descripción realizada por (Hernández Camacho, 2015), podemos decir que MPLS debe seguir los siguientes pasos:

- Creación y distribución de etiquetas
- Creación de tablas en cada router
- Creación de LSPs.
- Agregar etiquetas a los paquetes con la información de la tabla.
- Envío del paquete.

Las operaciones de MPLS se ejecutan en los LSR. Los routers LER, también llamados router de ingreso como se explicó anteriormente, operan como la principal interface entre la red MPLS y la tecnología de Capa 2 existente. Los routers LER son los encargados de colocar la etiqueta cuando los paquetes ingresan a la red MPLS provenientes de redes externas (Aslam y Aziz 2008) y se utilizan protocolos de distribución para compartir la información de las etiquetas entre los distintos LSRs.

El paquete enviado desde un router CE (customer edge) llega al router del proveedor (PE – Provider Edge) y de ahí se reenvía al primer LSR o LSR de ingreso. Aquí el LSR determinará la FEC y entonces insertará una o varias etiquetas MPLS. A continuación, el paquete se envía al siguiente router del backbone MPLS según la información de ese túnel. Cuando un paquete ya etiquetado se recibe en un LSR intermedio, se examina primero la última etiqueta de la pila, realizando con ella alguna operación.

Cada LSR construye una tabla de etiquetas LIB (Label Information Base) a medida que recibe la información de las etiquetas (Figura 2.4). La tabla LIB es donde se especifica el mapeo de cada etiqueta con una interfaz, tanto de entrada como de salida.

Los encaminadores LSR de tránsito no examinan lo que haya debajo de la etiqueta MPLS; he aquí la razón de que MPLS sea independiente del protocolo.

En el ejemplo de la siguiente figura un paquete de entrada por la interfaz 2 con la etiqueta 51, se redirigiría a la interfaz 5 con la etiqueta 37.

| Tabla de Etiquetas LIB | | | |
|---|---|---|---|
| Interfaz de Entrada | Etiqueta de Entrada | Etiqueta de Salida | Interfaz de salida |
| 2 | 51 | 37 | 5 |
| 3 | 15 | 84 | 6 |
| 3 | 45 | 22 | 4 |
| ... | ... | ... | ... |

*Figura 2.4: Tabla de Etiquetas LIB (Label Information Base)*

Los LSR construyen también la tabla LFIB (Label Forwarding Information Base) mediante los protocolos de distribución de etiquetas utilizados en el plano de datos y se utiliza para el reenvío de paquetes dentro de un dominio MPLS (Hernández Camacho, 2015).

El siguiente paso es la creación de los LSP, los cuales se crean en orden inverso a la trayectoria del paquete, lo que significa que el LSP se crea en el nodo destino hacia el nodo origen.

Para la transmisión de un paquete sobre la red MPLS, una vez que el paquete ya está etiquetado, se envía al siguiente salto LSR usando la tabla LFIB. Este paquete va saltando de LSR en LSR basándose en la tabla LFIB de cada router.

Para realizar el reenvío de paquetes en una red MPLS los LSR utilizan las siguientes operaciones:

- **SWAP:** La etiqueta se intercambia por una nueva etiqueta y el camino asociado con la nueva etiqueta es utilizado para el reenvío de paquetes.

- **PUSH:** Una nueva etiqueta es colocada sobre la etiqueta existente, lo que se denomina pila de etiquetas. Esto ayuda a encapsular el paquete en otra capa de MPLS. Esta operación se utiliza en MPLS VPN.

- **POP:** La etiqueta se elimina del paquete. En este punto, el paquete ya puede salir de la red MPLS y rutearse como cualquier otro paquete IP.

Cuando un LSR recibe un paquete etiquetado, a la etiqueta superior se le debe aplicar una de las operaciones indicadas, de acuerdo a lo que determine el LSR basándose en la tabla LFIB.

Normalmente lo que hacen estos routers es hacer un SWAP de la etiqueta. Finalmente, el paquete llega el router LER de salida, el cual es el encargado de quitar la última etiqueta realizando una operación POP y enviar el paquete hacia su destino por enrutamiento IP convencional (si es un paquete IP o lo que haya por encima). Puede acordarse también entre el último y el penúltimo nodo que sea el penúltimo quien retire la etiqueta, con lo cual puede evitarse en el último nodo dos búsquedas en las tablas de envío, primero en la tabla de MPLS y luego en IP. Es decir, este último router que se encuentra en la frontera debe contener la información de encaminamiento necesaria para que dicho paquete (fuera de la red MPLS) pueda ser reenviado a su destino, que por ejemplo podría ser el router frontera PE del siguiente proveedor hacia el router CE de la red del destinatario.

La Figura siguiente muestra el esquema del funcionamiento de una red MPLS.

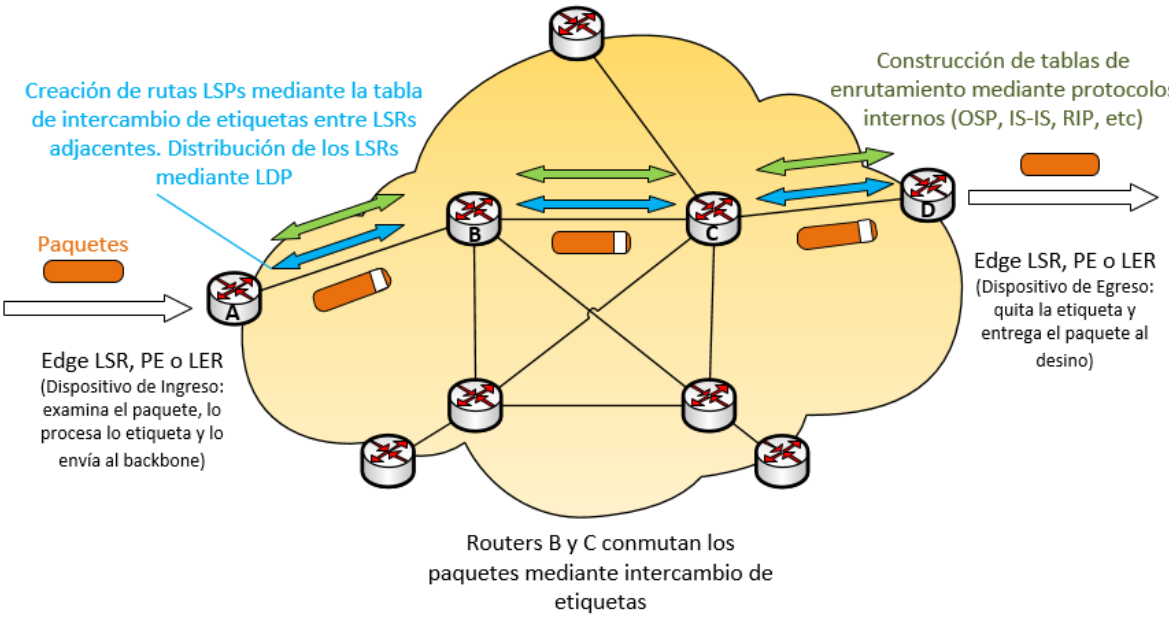


*Figura 2.5: Esquema de funcionamiento de una red MPLS*

Resumiendo, podemos decir que la función del LER de entrada es determinar el LER de salida apropiado y LSP a dicho LER, asociado a la FEC. MPLS tiene la propiedad que múltiples tipos de tráfico pueden ser multiplexados en un único LSP. Por lo tanto, un único LSP puede ser utilizado para transportar todo el tráfico (por ejemplo, L3VPN, IP pública y la capa 2) entre un LER de entrada y uno de salida. Los routers a lo largo del camino toman su decisión de reenvío sobre la base de un formato fijo de cabecera MPLS, por lo que no es necesario el almacenamiento de rutas relativas a los paquetes tunelizados subyecentes. Se trata de una propiedad de escalamiento importante, de lo contrario cada uno de los routers tendría que llevar información de enrutamiento a la suma de la información de encaminamiento realizada por todos los routers de borde de la red. (Minei y Lucek, 2011)

## 2.5 Aplicaciones de MPLS

Según (Gerometa, 2012) MPLS puede ser utilizado para entregar diferentes servicios IP:

**Enrutamiento IP de unicast:** es el servicio básico cuya principal ventaja es que al realizar el reenvío de tráfico sobre la base de las etiquetas ofrece mucho mayor performance que en el enrutamiento IP tradicional en los dispositivos de core.

**Enrutamiento IP de Multicast:** puede soportar enrutamiento multicast utilizando PIMv2 (Proocol Independent Multicast) con extensiones para MPLS para propagar las etiquetas. En este caso una FEC es una dirección multicast de destino almacenada en la tabla de enrutamiento multicast.

**MPLS TE:** La implementación de ingeniería de tráfico permite definir rutas en función de requerimientos específicos como ancho de banda, medios, priorización de tráfico etc.

Permite controlar el flujo de tráfico dentro de la red, reducir la congestión y hacer mejor uso de los recursos de red disponibles.

Para esta prestación se requiere la implementación de OSPF o IS-IS para que cada LSR tenga visión de toda la topología y los enlaces en la red, adicionalmente se implementa RSVP para propagar las etiquetas

Requiere que todos los Edge LSR deben ser capaces de crear túnel LSP bajo demanda.

**QoS:** Es posible aplicar calidad de servicio para dar tratamiento diferenciado a diferentes tráficos a través de la red MPLS, para lo cual se utilizan los bits experimentales o se pueden crear túneles separados para cada clase. En estos casos, una FEC es la combinación de una red de destino y una clase de servicio.

**VPNs MPLS:** Es un servicio muy escalable que permite que cada usuario de la red de transporte MPLS vea la totalidad de la red como un backbone IP privado.

Es una aplicación muy común en proveedores de servicios que permite el desarrollo de verdaderas redes privadas virtuales sobre una infraestructura de transporte compartida, como también la distribución de servicios a grupos de usuarios.

Cada VPN soporta internamente diferentes servicios IP como Multicast, QoS, etc.

Para ello se utilizan un stack de 2 etiquetas: una identifica la ruta hacia el destino (el router de egreso), la otra la VPN (la interfaz de egreso).

Las VPN-MPLS requieren la implementación de MP-BGP.

**AToM:** Es la solución de Cisco para transportar tráfico de capa 2 sobre el backbone MPLS. Permite acomodar tráfico Ethernet, Frame RElay, ATM, PPP y HDLC.

Permite a los proveedores de servicios brindar un servicio de conectividad a clientes con redes de capa 2 utilizando una única infraestructura de red.

Para esto utiliza 2 niveles de etiquetas: la externa identifica el túnel y permite el reenvío del paquete a través de la red, la interna determina la interfaz de egreso.

## 2.6 Ventajas y desventajas de MPLS

La principal ventaja de MPLS radica, como ya se mencionó, en que la conmutación de paquetes mediante el uso de etiquetas es un proceso más rápido que el encaminamiento basado en el análisis de la cabecera.

También es ventajoso el hecho de que es posible clasificar un paquete dentro de una FEC atendiendo otros criterios además de su dirección de destino; por ejemplo, examinando otros campos de la cabecera de red, de la cabecera de transporte o de aplicación, el puerto de entrada en el dispositivo de encaminamiento, etc., además de atender a la calidad de servicio con que se desea que se propague dicho paquete. Este proceso complejo de clasificación, no impacta en los dispositivos intermediarios de la red ya que estos solo se encargarán de encaminar los paquetes en función de sus etiquetas y en tal sentido el sistema resulta fácilmente escalable.

El hecho de establecer una misma FEC para distintos paquetes es beneficioso para el tráfico de aplicaciones en tiempo real las que son extremadamente sensibles a las fluctuaciones

en el retardo y además es importante para la monitorización del tráfico ya que los paquetes pertenecientes a una misma clase de equivalencia FEC son fácilmente identificables gracias a su etiqueta.

El uso de etiquetas también facilita la gestión de rutas explícitas que sirven para emular circuitos virtuales sobre una tecnología no orientada a la conexión, cuyo establecimiento puede formar parte de la ingeniería del tráfico en la red. Además, la posibilidad de proporcionar encaminamiento jerárquico, gracias a la posibilidad de encapsular un "camino" LSP en otro, mediante la anidación de etiquetas, genera un impacto significativo, aunque este puede verse empañado debido al incremento de la proporción de cabecera transportada lo que reduce el rendimiento.

El tráfico en una red MPLS se mueve de forma eficaz a través de la Red del proveedor de Servicios ofreciendo un rendimiento sólido de ancho de banda y garantías de nivel de servicio. En general, podemos decir que en cuanto a calidad de servicio (QoS), MPLS ofrece un ancho de banda garantizado entre el emplazamiento y la red del proveedor de servicios, ya que fue diseñado con el fin de atender en cuanto a clases de servicios (CoS), es decir separar el flujo en video, VoIP y datos.

MPLS también tiene una notable facilidad para la creación de redes privadas virtuales (VPN) y la creación de túneles IP.

En cuanto a las desventajas que podríamos mencionar se encuentra el hecho de que el esquema MPLS es orientado a la conexión, lo que implica una mayor vulnerabilidad en situaciones de fallo, lo que se contrapone con las características de las redes que poseen un elevado grado de fiabilidad, derivado de la naturaleza sin conexión del protocolo IP y su capacidad de tolerancia a fallos debido a la reacción de los protocolos de encaminamiento dinámico. En tal sentido resulta conveniente introducir mecanismos de recuperación de faltas asociadas a la arquitectura MPLS: notificación a los dispositivos de encaminamiento afectados, búsqueda de rutas alternativas, desvío del tráfico a las mismas, etc.

Debemos entender, que la identificación mediante una etiqueta la QoS deseada no implica que esta solicitud se satisfaga por lo que es necesario que las tecnologías de red subyacentes provean los mecanismos necesarios para garantizar dicha calidad.

Otra desventaja de MPLS es el costo del ancho de banda. El alto costo por megabit que las demandas de MPLS pueden estar fuera del alcance. Finalmente, una red MPLS no ofrece protección de datos incorporada, y si se implementa incorrectamente, puede abrir la red a vulnerabilidades.

## 2.7 Conclusiones del capítulo

Se realizó un resumen de las características y funcionamiento de MPLS, entendiendo su historia y predecesores, la utilización en el presente, perspectiva actual y la proyección a futuro de su utilización. Se realizó una descripción breve de su funcionamiento y forma de operación, encuadrando su funcionamiento en el modelo OSI, y describiendo la forma de asignación y distribución de etiquetas. Finalmente se describieron la aplicación de MPLS y se analizaron sus ventajas y desventajas.

Si bien las ventajas que introdujo MPLS son indiscutibles, uno se los problemas que se plantea es la selección del mejor LSP para una determinada demanda y una topología de red determinada. En el capítulo 3 se detalla el problema que se pretende resolver y su formulación matemática.

# Capítulo 3
# Definición del problema y modelado de la red

## 3.1 Introducción

El objetivo básico de la Ingeniería de Tráfico (Traffic Engineering, TE) es adaptar los flujos de tráfico a los recursos físicos de la red. La idea es equilibrar de forma óptima la utilización de esos recursos, de manera que no haya algunos que están sobre-utilizados, creando cuellos de botella, mientras que otros puedan estar subutilizados (Delfino *et al.,* 2006).

Las redes de comunicaciones han evolucionado hacia esta infraestructura capaz de transmitir flujo de información multiservicio sobre una misma plataforma. Por ejemplo, es posible transmitir datos, voz y video en la misma infraestructura basada en IP. Debido a las características específicas de cada tipo de tráfico, la red debe tratar cada uno de ellos de manera diferencial para garantizar una calidad de servicio demandada por los usuarios (Cruz *et al.,* 2013).

Cuando se consideran redes MPLS, una de las tareas de la ingeniería de tráfico consiste en seleccionar LSPs adecuados para un correcto equilibrio de carga en el sistema, disminuyendo la congestión y maximizando la utilización de los enlaces. Los protocolos de enrutamiento juegan un papel vital para el cálculo de rutas óptimas en la entidad MPLS TE, donde se pueden utilizar varios algoritmos para el cálculo de rutas, dependiendo de las funciones objetivo (Masood *et al.,* 2018),

El mecanismo MPLS puede canalizar múltiples tipos de tráfico a través de la red central. El túnel es la ruta por donde fluye el tráfico en la red central MPLS. La tunelización es una herramienta poderosa porque solo los enrutadores de entrada y salida necesitan conocer el contenido del tráfico transportado a través de dicho túnel. Los detalles están ocultos a los enrutadores en el núcleo. Con el uso de túneles MPLS, el tráfico se puede enrutar explícitamente siguiendo las políticas de tráfico especificadas (Ridwan *et al.,* 2019)

Sin embargo, más allá del avance impuesto por MPLS, el problema de la optimización de la selección de las rutas para dirigir el flujo de una determinada demanda sigue existiendo y es un problema complejo de abordar.

En este sentido han sido aplicadas distintas técnicas para encontrar soluciones óptimas a este problema y se pueden utilizar varios algoritmos para el cálculo de dichas rutas y con diferentes funciones objetivo. En general, esto implica resolver un problema de optimización combinatorio considerado NP-hard (Barabas *et al.,* 2012; Masood *et al.,* 2017; Masood *et al.,* 2018).

La teoría de la complejidad estudia cómo crece el costo computacional para resolver un problema determinado en función del tamaño de dicho problema. Así podrá haber problemas de complejidad lineal, polinómica, logarítmica, exponencial, dependiendo de cómo varia, por ejemplo, el tiempo necesario para resolver el problema cuando crece su tamaño. Además, un problema determinado puede ser resuelto por diferentes algoritmos y esto trae aparejada la diferencia conceptual entre complejidad de un problema y complejidad de un algoritmo, puesto que puede

haber algoritmos más eficientes que otros en la resolución del mismo problema. Se admite que la complejidad de un problema es la del mejor algoritmo que lo resuelve.

Se le llama P al conjunto de problemas en los que es posible encontrar una respuesta en un tiempo acotado polinomialmente. Se denomina NP al conjunto de problemas en los que se puede comprobar, en un tiempo polinomial, si una solución es correcta o no. Todos los problemas que están en P, también están en NP, dado que, si es posible encontrar una solución en tiempo acotado, también es posible verificar si una solución es correcta o no dentro de la misma cota.

Lo contrario no está probado. El hecho de comprobar, en tiempo polinomial si una solución, es correcta no necesariamente implica que es posible encontrar una solución al problema en dicho tiempo. En otras palabras, la pregunta ¿P=NP? aún es un problema abierto.

Por otra parte, hay problemas en NP para los cuales no se conocen algoritmos que los resuelvan en tiempos acotados polinomialmente. Eso no significa que el problema no pertenezca también al conjunto P, sino que aún no se conoce un algoritmo que lo resuelva en tiempos aceptables.

Un problema X es NP-Completo si hay un problema Y perteneciente a NP, tal que Y es reducible a X en tiempo polinomial. Esto significa que, si existe un algoritmo que resuelva Y eficientemente, ese algoritmo puede usarse como subrutina para resolver X eficientemente. Los problemas NP-Completo son tan difíciles como los problemas NP.

Un problema X es NP-Difícil si hay un problema NP-Completo Y, tal que Y es reducible a X en tiempo polinomial. Los problemas NP-Hard son tan difíciles como los problemas NP-Completo, pero no necesariamente están incluidos en dicho conjunto

Recientemente, las herramientas de optimización basadas en aplicaciones de inteligencia artificial han obtenido un reconocimiento importante por resolver problemas de optimización en diferentes campos. Estas herramientas o técnicas de optimización se desarrollan en varios enfoques entre los que se cuentan los algoritmos bioinspirados, basados en la genética o inspirados en el estudio de comportamiento de animales que trabajan en enjambres para conseguir alimento (Gero 1987; Masood *et al.*, 2018). Estos métodos son agrupados dentro de las estrategias heurísticas y metaheurísticas las que serán tratadas en el capítulo 4.

Para la aplicación de estos métodos, es necesario un modelado matemático del problema y una definición de la función objetivo que se desea optimizar, lo que se desarrollará a continuación.

## 3.2 Definición del problema

El problema de la optimización a considerar es encontrar la mejor ruta para una determinada secuencia de flujo de datos que llegan a una red multiservicio, de tal manera que se minimice algún criterio de desempeño. El abordaje de dicho problema, supone como dijimos, un modelo para representar la red y una función objetivo a optimizar. Se han considerado varias alternativas, tanto mono objetivo como multiobjetivo teniendo en cuenta diferentes factores tales como la disponibilidad y utilización del ancho de banda, medición de congestión a través de diferentes métricas, tipo de tráfico o calidad de servicio requerido. Más allá de la mayor o menor

complejidad que pueda imponerse a la función objetivo, el problema puede considerarse una variante del problema general de enrutamiento (Reis, 2019).

## 3.3 Representación de una red MPLS

Para la representación de una red MPLS, se considera la topología modelada mediante un grafo *G=(V, E)* donde *V* es el conjunto de *n* nodos que representan los enrutadores en la red mientras que *E* es el conjunto de los *m* enlaces entre nodos de *G*. Cada enlace tiene un costo $c_e$ y una capacidad $Cap_e$ asociados.

Matemáticamente, la red y situación a resolver, quedará determinada mediante cuatro matrices:

- *Matriz de adyacencia (A):* se corresponde con la matriz de adyacencia del grafo, las filas y columnas se corresponden con los nodos, y cada componente valdrá uno si existe conexión (nodo fila →nodo columna) y un cero si no existe tal conexión. Es decir, la representación se realiza mediante un grafo dirigido, donde por ejemplo puede existir un canal de transmisión desde el nodo A hasta el nodo B, pero puede existir o no un canal de transmisión en sentido contrario (desde B hacia A).

- *Matriz de Costo de los enlaces ($Costo_e$):* representará el costo de cada enlace convirtiéndose en un grafo ponderado. Este costo podría calcularse utilizando diferentes métricas, que permitan la selección de los caminos. Por ejemplo, la utilizada en OSPF, un ancho de banda de referencia dividido por el ancho de banda de la interfaz, es una métrica posible.

- *Matriz de Capacidad ($Cap_e$):* Representa la capacidad de transmisión de cada enlace.

- *Matriz de Requerimientos (T):* esta matriz representará los requerimientos, cada fila representa un requerimiento, la primera columna representa el nodo origen, la segunda columna el nodo destino, la tercera columna el volumen de información a transmitir.

Definiendo estas 4 matrices quedará representada la topología de la red que se desea optimizar y éstos serán los insumos necesarios para ejecutar los algoritmos de optimización.

## 3.4 Modelado matemático

Cada requerimiento establecido en la matriz T, se considera como una solicitud de ancho de banda asegurado entre dos puntos (dos nodos) de la red. Por tanto, un requerimiento *t* es especificado mediante una terna y se representa mediante un vector de tres componentes como se muestra a continuación:

$$t = (S_t, D_t, d_t) \qquad (3.1)$$

Donde:
$S_t$ es el nodo origen
$D_t$ el nodo destino
$d_t$ la demanda requerida o el flujo

La demanda requerida se representa de forma genérica en [*unidades/seg*] y se corresponde con la cantidad de información que se desea transportar entre los nodos origen y destino suponiendo que entre ambos nodos existe al menos un camino (de uno o más enlaces). Si existiese más de un camino posible, dicha demanda podrá dividirse entre los caminos existentes. Al conjunto de los caminos posibles entre $S_t$ y $D_t$ se lo denomina $P_t$ y a la cantidad de caminos se lo denota como $L_t$.

$$P_t = \{caminos\ posibles\ para\ el\ requerimiento\ t\} \qquad (3.2)$$

La figura 3.1 muestra un ejemplo de requerimiento entre un origen $S_1$ y un destino $D_1$. Se han identificado dos caminos posibles para el flujo entre los nodos extremos. El primero, identificado como $p_1$ atraviesa los enlaces $e_1$, $e_2$, $e_4$ y $e_5$. Otro camino, de menor número de saltos, está identificado como $p_2$ y utiliza los enlaces $e_1$, $e_3$, $e_5$. En este ejemplo para el requerimiento $t=(S_1, D_1, d_1)$, la cantidad de caminos posibles es $L_t=2$, y el conjunto de los caminos posibles es $P_t=\{p_1, p_2\}$.

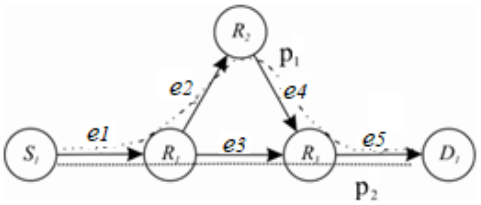


*Figura 3.1: Dos caminos posibles para un requerimiento de tráfico*

El costo total de un camino *p* para una solicitud *t* se puede calcular como:

$$costo_{p,t} = \sum_{e \in E} c_e \, a_{p,t,e} \qquad (3.3)$$

Donde:
$c_e$ corresponde al costo del enlace *e*
$a_{p,t,e}$ una variable binaria cuyo valor es 1 en el caso del que camino *p* de la solicitud *t* use el enlace *e* y 0 en caso contrario.

Si consideramos que $x_{p,t}$ es la porción de tráfico de una solicitud *t* que se enruta por el camino *p*, se debe considerar que la demanda de dicha solicitud debe ser satisfecha, es decir, que debe cumplirse:

$$\sum_{p=1}^{L_t} x_{p,t} = d_t \quad \forall t \in T \qquad (3.4)$$

El costo total del sistema está dado por:

$$z = \sum_{t \in T} \sum_{p=1}^{L_t} costo_{p,t} \, . \, x_{p,t} \qquad (3.5)$$

Es decir, que el problema a optimizar se corresponde con minimizar z. La formulación matemática del problema queda representada por las ecuaciones 3.6, 3.7 y 3.8:

$$min \left\{ \sum_{t \in T} \sum_{p=1}^{L_t} costo_{p,t} \, . \, x_{p,t} \right\} \qquad (3.6)$$

s.a.

$$\sum_{p=1}^{L_t} x_{p,t} = d_t \quad \forall t \in T \qquad (3.7)$$

$$\sum_{t \in T} \sum_{p=1}^{L_t} x_{p,t} \quad a_{p,t,e} < Cap_e \qquad (3.8)$$

La ecuación 3.7 exige el cumplimiento de la demanda mientras la ecuación 3.8 representa la restricción de la capacidad del enlace, es decir, que la suma de los flujos asignados a cada enlace no puede superar su capacidad de transmisión.

## 3.5 Representación de una solución

Una solución a un requerimiento consiste en la descomposición del ancho de banda requerido en los caminos posibles del conjunto $P_t$.

Una propuesta de solución, entonces, consiste en un conjunto de vectores, en general de diferente longitud, cuyas componentes representan la porción de flujo por cada camino alternativos para transportar el tráfico demandado, es decir, la demanda se divide entre los caminos alternativos elegidos.

Para aclarar el modelo propuesto, considérese el ejemplo planteado en (Cruz, *et al.,* 2013) en el que las demandas pueden representarse también mediante una matriz *D* (que deriva de la matriz de requerimientos T) y cuyos elementos no nulos especifican la demanda en cada enlace. Para el ejemplo planteado en la figura 3.2, se presenta una red cuya matriz de adyacencia es ***A*** y cuya matriz de demanda es ***D***.

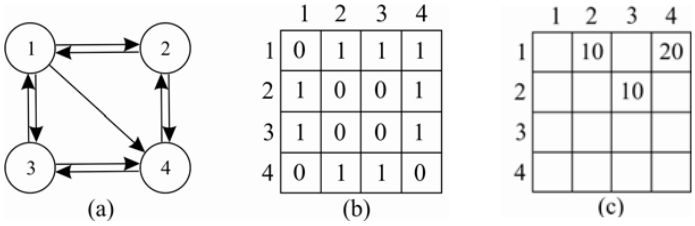


*Figura 3.2: (a) Una red de ejemplo (b) Matriz de adyacencia A (c) Matriz de Demanda D*

Analizando la matriz de demanda con lo expuesto anteriormente, existen tres solicitudes de tráfico en los enlaces (1,2) (1,4) y (2,3), de 10 *u/seg*, 20 *u/seg* y 10 *u/seg* respectivamente.

Utilizando la nomenclatura ya definida, es posible expresar el conjunto de solicitudes de la siguiente manera:

$$T=\{t_1, t_2, t_3\} \tag{3.9}$$

con

$$t_1=(1,2,10) \tag{3.10}$$

$$t_2=(1,4,20) \tag{3.11}$$

$$t_3=(2,3,10) \tag{3.12}$$

El primer requerimiento, indica que se debe transportar un flujo de 10 *u/seg* entre los nodos 1 y 2. Los caminos posibles entre ambos nodos son:

$$p_{1,t1}=\{(1,2)\} \tag{3.13}$$

$$p_{2,t1}=\{(1,3)\ (3,4)\ (4,2)\} \tag{3.14}$$

$$p_{3,t1}=\{(1,4)\ (4,2)\} \tag{3.15}$$

Una propuesta de solución podría ser descomponer el flujo de 10 *u/seg* en dos caminos de 5 *u/seg*. En otras palabras, la mitad del flujo pasará por el enlace (1,2) y la otra mitad por los enlaces (1,3) (3,4) y (4,2). El camino $p_3$ no se utiliza en esta propuesta de distribución. Tal situación se muestra en la figura 3.3.

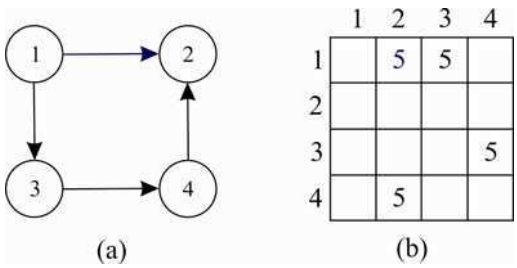


*Figura 3.3: (a) Dos caminos en la red. (b) Distribución de flujo para el requerimiento t1*

La propuesta para este requerimiento se representa mediante un vector de longitud $L_t$ donde cada componente indica el flujo asignado a cada camino en $P_t$. En este ejemplo, el vector será:

$$x_{t1} = \left[x_{p1,t1}\ x_{p2,t1}\ x_{p3,t1}\right] = [5\ 5\ 0] \quad (3.16)$$

Para los demás requerimientos de tráfico se trabaja de manera similar. Así, al considerar el segundo requerimiento en el ejemplo,

$$t_2=(1,4,20) \quad (3.17)$$

pueden identificarse los caminos

$$p_{1,t2}=\{(1,2)\ (2,4)\} \quad (3.18)$$

$$p_{2,t2}=\{(1,3)\ (3,4)\} \quad (3.19)$$

$$p_{3,t2}=\{(1,4)\} \quad (3.20)$$

Mientras que, en el tercer y último requerimiento,

$$t_3=(1,4,10) \quad (3.21)$$

los caminos posibles son

$$p_{1,t3}=\{(2,1)\ (1,3)\} \quad (3.22)$$

$$p_{2,t3}=\{(2,1)\ (1,4)\ (4,3)\} \quad (3.23)$$

$$p_{3,t3}=\{(2,4)\ (4,3)\} \quad (3.24)$$

y el flujo podría repartirse sobre estos caminos de distintas maneras, siempre y cuando por supuesto, el flujo total sobre un enlace no sobrepase la capacidad del mismo.

Una solución estará representada por un conjunto de vectores: uno por cada uno de los requerimientos de tráfico que están explicitados en la matriz de demanda. Si se considera que el conjunto de requerimientos $T$ tiene una cardinalidad igual a $r$, entonces la solución es un conjunto de $r$ vectores de dimensión $L_{t_i}, \forall\ i = 1{:}r$, y una solución podría representarse como en la ecuación 3.25

$$x = [(x_{p1,t1}, x_{p2,t1}, \dots x_{pL_{t1},t1})(x_{p1,t2}, x_{p2,t2}, \dots x_{pL_{t2},t2}) \dots (x_{p1,tr}, x_{p2,tr}, \dots x_{pL_{tr},tr})\ ] \quad (3.25)$$

De esta manera, $x$ tendrá tantos vectores como requerimientos se tengan, y cada vector tendrá tantas componentes como caminos alternativos tengan cada uno de esos requerimientos.

## 3.6 Dimensión del espacio de soluciones

Cuando se plantea el formato de una solución (Ec. 3.25), parece simple encontrar un esquema que cumpla con los requisitos de demanda (Ec. 3.7). Sin embargo, no cualquier combinación de las componentes del vector *x* (Ec. 3.25) satisface la restricción de no sobrepasar la capacidad de los enlaces, y aun cuando se satisfagan la demanda y no se sobrepase la capacidad de los enlaces, es complejo encontrar la mejor solución, es decir aquella que minimice el costo total del sistema (Ec.3.6).

Para entender la complejidad, obsérvese la cantidad de posibles soluciones para el ejemplo planteado anteriormente. Si su solución está dada por:

$$x = [(x_{p1,t1}, x_{p2,t1}, x_{p3,t1})(x_{p1,t2}, x_{p2,t2}, x_{p3,t2}) \dots (x_{p1,t3}, x_{p2,t3}, x_{p3,t3})] \quad (3.26)$$

Donde:

$x_{pt}$ representa la porción de flujo del requerimiento *t* que se envía por el camino *p.*

Entonces, la cantidad posible de variaciones para un vector solución para cualquiera de los requerimientos, podría calcularse como

$$Variaciones\ con\ repetición = (t_x(3) + 1)^{L_t} \quad (3.27)$$

Donde:

$t_x(3)$ corresponde a la cantidad de unidades de flujo requerido

$L_t$ corresponde a la cantidad de caminos de un determinado requerimiento, coincidente con la cantidad de componentes del vector solución pertenecientes al requerimiento *t* (3 para cualquiera de los requerimientos del ejemplo).

Recordando que los dos primeros requerimientos son de 10 *u/seg* y el último de 20 *u/seg* la cantidad de variaciones para cada subvector de x es;

$$Variaciones\ con\ repetición(t1) = (11)^3 = 1331 \quad (3.28)$$

$$Variaciones\ con\ repetición(t2) = (11)^3 = 1331 \quad (3.29)$$

$$Variaciones\ con\ repetición(t3) = (21)^3 = 9261 \quad (3.30)$$

De todas estas alternativas se deben considerar solamente aquellas que cumplan con la demanda, es decir se deberá cumplir que:

$$\sum(x_{p1,t1}, x_{p2,t1}, x_{p3,t1}) = 10 \quad (3.31)$$

$$\sum(x_{p1,t2}, x_{p2,t2}, x_{p3,t2}) = 10 \quad (3.32)$$

$$\sum(x_{p1,t3}, x_{p2,t3}, x_{p3,t3}) = 20 \quad (3.33)$$

Con estas restricciones ecuaciones 3.28 a 3.30 se reducen a:

$$Variaciones\ que\ cumplen\ la\ demanda\ (t1) = 69 \quad (3.28)$$

$$Variaciones\ que\ cumplen\ la\ demanda(t2) = 69 \quad (3.29)$$

$$Variaciones\ que\ cumplen\ la\ demanda(t3) = 210 \quad (3.30)$$

Es decir que la cantidad de variaciones para el vector solución que satisfacen la demanda se puede calcular como:

$$Cantidad\ de\ Variaciones = VarValidas(t1) * VarValidas(t2) * VarValidas(t3) \quad (3.31)$$

$$Cantidad\ de\ Variaciones = 69 * 69 * 210 = 999810 \quad (3.32)$$

De este total de variaciones será necesario entender cuáles son las que satisfacen la restricción de la capacidad de los enlaces, y del conjunto restante cuál de ellas es la que mejor minimiza el costo de la red.

El espacio de soluciones crece abruptamente a medida que crece el tamaño de la red, o los caminos posibles para cada requerimiento, ya que se incrementan la cantidad de componentes del vector solución y por ende existirá un mayor número de variaciones. Además, si los requerimientos son de mayor cantidad de unidades de flujo, también crece el espacio de soluciones. Por ejemplo, si el requerimiento *t1* del problema cambiara de *10 u/seg* a *20 u/seg,* se incrementaría el valor de la ecuación 3.28 y la cantidad de variaciones posibles (ecuación 3.32) ascendería a 3042900, lo que representa un espacio que triplica al espacio original.

Debido a esta alta complejidad computacional del problema de optimización combinatoria, la búsqueda exhaustiva que asegure encontrar la óptima solución existente es inviable debido a que los tiempos computacionales involucrados en el proceso podrían llegar a ser prohibitivos por lo que se propone la optimización mediante técnicas heurísticas en las cuales el objetivo es encontrar buenas soluciones en tiempos computacionales acotados, aun cuando no haya garantía de encontrar de manera exacta la solución de mínimo costo global.

## 3.7 Conclusiones del capítulo

Se estableció una representación matemática de la red, se ha propuesto una representación de la solución y se ha determinado la dimensión del espacio de búsqueda de las soluciones, y su abrupto crecimiento conforme aumenta la cantidad de caminos entre los nodos involucrados en las demandas sobre la red y las unidades de flujo demandadas.

Esto produce que, cuando el tamaño de la instancia crece, los tiempos necesarios para la obtención del óptimo tornan inviable la aplicación de algoritmos deterministas, lo que justifica la necesidad de aplicar otros métodos de resolución del problema de optimización combinatoria planteado. En el próximo capítulo, se presentarán algunas estrategias heurísticas, las cuales permitirán la obtención de soluciones en tiempos aceptables.

# Capítulo 4
# Metaheurísticas

## 4.1 Introducción

En el capítulo 3 se ha planteado el problema de asignación de flujos en una red multiservicio a partir de un conjunto de requerimientos y se ha determinado que las soluciones provienen de la resolución de un problema combinatorio catalogado como NP-duro. En este capítulo se seleccionan como base, heurísticas inspiradas en la naturaleza y, más específicamente tres heurísticas basadas en inteligencia de enjambre. Se realiza una revisión del estado del arte, se justifica la elección de las estrategias mencionadas y se diseñan, sobre esa base cuatro algoritmos adaptados para la resolución del problema de optimización combinatoria de asignación de flujos en redes multiservicio.

En el campo de la optimización es posible realizar una clasificación de los métodos a utilizar en dos grandes grupos, los algoritmos deterministas o tradicionales y los métodos estocásticos. Si bien los primeros proporcionan soluciones exactas y repetibles, para los problemas no lineales, multimodales o combinatorios, está demostrado que los métodos deterministas no son aptos cuando el tamaño de la instancia del problema crece, dado que necesitan tiempos prohibitivos para alcanzar el óptimo. Por otro lado, los métodos estocásticos, no pueden garantizar la obtención del óptimo global, aunque, al incorporar términos aleatorios aumentan la diversidad de soluciones obtenidas (Yang, 2018). En el campo de la ingeniería, la obtención de buenas soluciones en tiempos aceptables, es muy apreciable y el desconocimiento del óptimo global, en muchos casos, es un precio que se está dispuesto a pagar.

Dentro de estos métodos estocásticos se encuentran los heurísticos y metaheurísticos. La diferencia radica en que los heurísticos usan un enfoque de prueba y error, proporcionando soluciones aceptables, mientras que los algoritmos metaheurísticos comprenden un nivel superior, debido a que no necesitan de una información detallada del espacio de búsqueda, sino solo la función objetivo y el dominio de las variables (Rajpurohit, *et al.,* 2017).

Un método heurístico es un procedimiento que, teniendo conocimiento de un problema y de las técnicas aplicables, aporta soluciones o se acerca a ellas usando una cantidad de recursos razonable. Por lo general, este recurso es el tiempo de cómputo que se emplea en la búsqueda de la solución. Se trata de procedimientos inteligentes en el sentido de realizar una tarea que no es producto de un riguroso análisis formal ya conocido y aplicable al tema, ya que en muchos casos sería ineficiente en cuanto al tiempo de cómputo, sino de algoritmos de simulación, de búsqueda o evolutivos que utilizan y también aportan al conocimiento experto sobre la tarea. Una solución heurística de un problema es aquella proporcionada por un método heurístico, es decir, aquella solución sobre la que se tiene cierta confianza de que alcanza algún grado de optimalidad y/o factibilidad (Reeves, *et al.,* 2014 y Tetzlaff, *et al.,* 2021)

Las metaheurísticas, también llamadas por algunos autores como heurísticas modernas (Tetzlaff *et al.,* 2021), se han desarrollado principalmente a partir de la década de 1980 y definen métodos que pueden ser aplicados a un conjunto de problemas diferentes. Son procedimientos genéricos de exploración del espacio de soluciones para problemas de optimización y búsqueda.

Son procedimientos iterativos que guían una heurística subordinada combinando en forma inteligente distintos conceptos para explorar y explotar adecuadamente el espacio de búsqueda. Las metaheurísticas brindan un mecanismo por el cual podemos encontrar buenas soluciones a un problema en tiempos de ejecuciones razonables, aunque estas soluciones no sean las mejores. (Depaux, 2011).

Según Depaux (2011) se puede mencionar como ventajas de las metaheurísticas que son algoritmos de propósito general, que son exitosos en la práctica, que por lo general son fácilmente implementables, que pueden paralelizarse fácilmente etc., a costa de encontramos con algoritmos no exactos sino aproximados, que no necesariamente llegan a una solución óptima, son no determinísticos y generalmente tienen poca base teórica. Mas allá de estas ventajas y desventajas, logrando un balance entre la intensificación, entendida como el esfuerzo empleado en la exploración en la región donde se busca, y la diversificación, el esfuerzo empleado en la búsqueda de regiones distantes del espacio (exploración) estos algoritmos alcanzan soluciones de buena calidad a un costo computacional aceptable.

Existe una concepción de ver a las metaheurísticas como un marco algorítmico general que puede ser aplicado a varios tipos de problema diferentes conservando su estructura y con relativamente pocas modificaciones para cada uno de ellos. En general, las metaheurísticas toman inicialmente una solución factible, la cual es luego mejorada usando heurísticas de mejoramiento embebidas en una estructura más general, i.e., Recocido Simulados (Simulated Annealing, SA), Algoritmos Genéticos (Genetic Algorithms, GA), Búsqueda Tabú (Tabu Search, TS), etc. (Tetzlaff *et al.,* 2021)

Muchos de los algoritmos metaheurísticos se sirven de inspiración en la naturaleza para su funcionamiento y dentro de ellos se encuentran los inspirados en la biología o bio-inspirados y a su vez un subconjunto de estos se los conoce como algoritmos basados en inteligencia de enjambre. Estos algoritmos están dentro de los más populares. (Fister *et al.,* 2013)

Si bien no es objeto de esta tesis un estudio detallado de las técnicas heurísticas y metaheurísticas es interesante una revisión rápida de las distintas posibilidades con que se cuenta para encarar el problema planteado.

Existen muchas formas de clasificar a los algoritmos dependiendo del enfoque, perspectiva de aplicación, criterios con que se los intenta agrupar e inspiración. Por ejemplo, se puede clasificar a los algoritmos con respecto a la ruta de búsqueda como algoritmos basados en trayectoria y algoritmos basados en población; cuando se trabaja con algoritmos con múltiples agentes se pueden clasificar como basados en la atracción o no basados en atracción; otras clasificaciones los separan en algoritmos basados en reglas y basados en ecuaciones, etc.

Es difícil encuadrar cada algoritmo en una clasificación, de hecho, algunos algoritmos pueden encuadrar en más de una categoría dependiendo del enfoque; sin embargo, una forma simple de agrupar los algoritmos, y siguiendo el criterio de (Fister *et al.,* 2013): en forma resumida, pueden establecerse cuatro categorías principales como se detallan a continuación y exponer algunos ejemplos de algoritmos clasificados en cada categoría como se detalla en la tabla 4.1.

1. **Algoritmos basados en Inteligencia de Enjambre (Swarm Intelligence SI):** Se basa en el comportamiento de una población o enjambre con múltiples agentes que interactúan de acuerdo a varias reglas establecidas. Por ejemplo, el comportamiento colectivo de insectos y otros animales como pájaros o peces. Si bien cada agente individualmente no presenta

una inteligencia propia, el sistema completo puede demostrar comportamientos de autoorganización mediante una inteligencia colectiva.

Se han desarrollado muchos algoritmos dentro de esta clasificación inspirados en el comportamiento colectivo de insectos sociales, como hormigas, termitas, abejas y avispas, así como de otras sociedades animales como bandadas de pájaros o peces etc. (Fister *et al.,* 2013)

2. **Algoritmos Bio-inspirados, pero no basados en SI:** Dentro de este grupo se encuentran algoritmos de inspiración biológica pero que no utilizan de forma directa el comportamiento de conjunto, sino que utilizan diferentes procedimientos evolutivos, como la selección natural, Muchos autores utilizan el nombre de algoritmos evolutivos para referirse a este grupo.

Los algoritmos basados en inteligencia de enjambre son un subconjunto de los algoritmos bio-inspirados y estos a su vez son un subconjunto de los algoritmos inspirados en la naturaleza. (Fister *et al.,* 2013)

Algunos algoritmos no son fáciles de clasificar dentro de estas categorías, y es necesario algunas consideraciones que toma el autor, por ejemplo, el algoritmo de evolución diferencial (DE), estrictamente hablando, no está bio-inspirado porque no existe un vínculo directo con ningún comportamiento biológico; sin embargo, como tiene cierta similitud con los algoritmos genéticos y también tiene una palabra clave “evolución”, lo ubicamos tentativamente en la categoría de algoritmos bio-inspirados. (Fister *et al.,* 2013)

3. **Algoritmos basados en Física y Química:** Encuadran aquí los algoritmos inspirados en la naturaleza, pero no necesariamente bio-inspirados. Son algoritmos que se basan en otros fenómenos. Existen algoritmos metaheurísticos basados en leyes físicas o químicas como cargas eléctricas, la gravedad, sistemas fluviales, entre otros. Por lo que estos algoritmos son Inspirados en la naturaleza, pero no Bio-inspirados.

4. **Otros Algoritmos:** En esta clasificación se encuentran aquellos algoritmos que no se pueden asociar a ninguna de las clasificaciones anteriores, en cambio utilizan fuentes como el comportamiento cultural o emociones.

En la siguiente tabla se muestran ejemplos de algoritmos dentro de la clasificación descripta, pudiendo ampliarse la información correspondiente a cada uno de ellos en (Fister *et al.,* 2013).

| Algoritmos basados en Inteligencia de Enjambre | Algoritmos Bio-inspirados, pero no basados en SI | Algoritmos basados en Física y Química: |
|---|---|---|
| Accelerated PSO | Atmosphere clouds model | Big bang-big Crunch |
| Ant colony optimization | Biogeography-based optimization | Black hole |
| Artificial bee colony | Brain Storm Optimization | Central force optimization |
| Bacterial foraging | Differential evolution | Charged system search |
| Bacterial-GA Foraging | Dolphin echolocation | Electro-magnetism optimization |
| Bat algorithm | Japanese tree frogs calling | Galaxy-based search algorithm |
| Bee colony optimization | Eco-inspired evolutionary algorithm | Gravitational search |
| Bee system | Egyptian Vulture | Harmony search |
| BeeHive | Fish-school Search | Intelligent water drop |
| Wolf search | Flower pollination algorithm | River formation dynamics |
| Bees algorithms | Gene expression | Self-propelled particles |
| Bees swarm optimization | Great salmon run | Simulated annealing |
| Bumblebees | Group search optimizer | Stochastic difusion search |

| Algoritmos basados en Inteligencia de Enjambre | Algoritmos Bio-inspirados, pero no basados en SI | Algoritmos basados en Física y Química: |
|---|---|---|
| Cat swarm | Human-Inspired Algorithm | Spiral optimization |
| Consultant-guided search | Invasive weed optimization | Water cycle algorithm |
| Cuckoo search | Marriage in honey bees | |
| Eagle strategy | OptBees | |
| Fast bacterial swarming algorithm | Paddy Field Algorithm | **Otros Algoritmos:** |
| Firefly algorithm | Roach infestation algorithm | Anarchic society optimization |
| Fish swarm/school | Queen-bee evolution | Artificial cooperative search |
| Good lattice swarm optimization | Shuffled frog leaping algorithm | Backtracking optimization search |
| Glowworm swarm optimization | Termite colony optimization | Differential search algorithm |
| Krill Herd | | Grammatical evolution |
| Monkey search | | Imperialist competitive algorithm |
| Particle swarm algorithm | | League championship algorithm |
| Virtual ant algorithm | | Social emotional optimization |
| Virtual bees | | |
| Weightless Swarm Algorithm | | |

*Tabla 4.1: Lista de ejemplos de algoritmos por categoría (Fister et al., 2013)*

Los algoritmos detallados en la tabla anterior constituyen una lista necesariamente incompleta, de hecho, existen otros desarrollados con posterioridad a esa publicación, así como también existen otras clasificaciones, como se mencionó con anterioridad. Torres-Treviño, por ejemplo, realizó una interesante taxonomía de algoritmos inspirados en el comportamiento de los seres vivos desarrollados hasta el 2020 y clasificados de acuerdo a los reinos como lo usan los biólogos. (Torres-Treviño, 2021). Con el objeto de no extender más esta introducción y solo a modo ilustrativo de la diversidad de algoritmos, se muestra una imagen representativa de la taxonomía del citado autor.

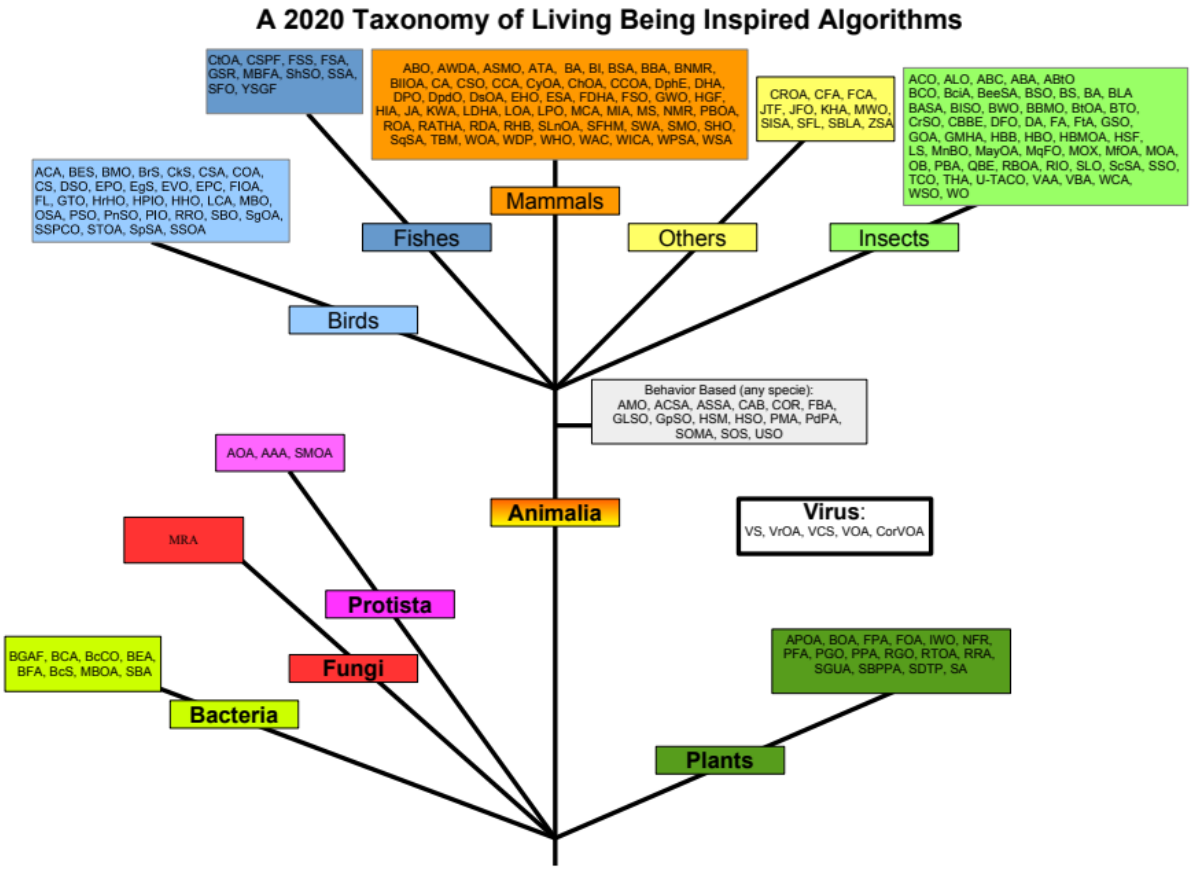


*Figura 4.1: Taxonomía de algoritmos inspirados en comportamiento de seres vivos. (Torres-Treviño, 2021).*

Como se detalló en el capítulo anterior, se intenta resolver es un problema de optimización combinatoria, donde la complejidad de la solución aumenta en forma exponencial no solo cuando se modifica la infraestructura de red, sino también cuando existen modificaciones en los requerimientos.

El avance constante de las tecnologías de cómputo provee recursos computacionales con más rapidez de procesamiento, lo cual permite a investigadores y profesionales de la ciencia y la ingeniería resolver problemas combinatorios complejos y de gran escala. Esto genera un interés creciente en resolver problemas complejos del mundo real utilizando métodos metaheurísticos implementados con técnicas que mejoran la eficiencia. (Tetzlaff *et al.,* 2021)

Esta potencialidad en los equipos de cómputos hace descender los tiempos necesarios para la actuación de los algoritmos metaheurísticos, y que entreguen soluciones aceptables en tiempos aceptables.

Las metaheurísticas pueden manejar variables discretas y reales, siendo aplicadas en la actualidad a una amplia gama de problemas de optimización de manera efectiva. Básicamente, los enfoques metaheurísticos basados tanto en la trayectoria (sucesión de posibles soluciones) como en la población (conjunto adaptable de posibles soluciones) apuntan a ubicar el óptimo global en el espacio de la solución a través de movimientos aleatorios. La diferencia clave entre las metaheurísticas está en la forma en que proponen el siguiente movimiento en el espacio de la solución (Tetzlaff *et al.,* 2021)

El problema planteado podría intentar resolverse con cualquiera de estas técnicas, por lo que surge ahora un nuevo problema que consiste en la selección de las estrategias a utilizar ya que, como se detalló, las posibilidades son muchas y la implementación de cada una requiere su tiempo tanto de diseño como de experimentación.

La inteligencia de enjambre es una disciplina, dentro del campo de la inteligencia artificial, que trata de imitar el comportamiento de seres vivos basado en las interacciones entre los integrantes de un enjambre tanto con sus pares como con el ambiente para conseguir un objetivo común. Generalmente basado en acciones simples, individuales y coordinadas a través de una fuerte comunicación mantienen, en el proceso, un control descentralizado y una organización colectiva.

Según (Fister *et al.,* 2013) los algoritmos basados en inteligencia de enjambre se encuentran entre los más populares y ampliamente utilizados. Hay muchas razones para tal popularidad, una de ellas es que los algoritmos basados en Inteligencia de enjambre generalmente comparten información entre múltiples agentes, por lo que la autoorganización, la coevolución y el aprendizaje durante las iteraciones pueden ayudar a proporcionar la alta eficiencia de la mayoría de los algoritmos basados en inteligencia de enjambre. Otra razón es que varios agentes se pueden paralelizar fácilmente, de modo que la optimización a gran escala se vuelve más práctica desde el punto de vista de la implementación.

Los científicos e investigadores están muy interesados en imitar el comportamiento de estas entidades inteligentes para formar inteligencia colectiva formulando ecuaciones matemáticas paso a paso o construyendo algoritmos avanzados para resolver problemas reales (Okwu y Tartibu, 2021).

En esta tesis se focaliza el interés en la primera clasificación de Fister: los algoritmos basados en inteligencia de enjambre. En particular se seleccionaron tres exponentes del grupo: el algoritmo de enjambre de partículas (PSO), el algoritmo basado en colonia de hormigas (ACO) y el algoritmo basado en murciélagos (BA). Considerando la clasificación de Torres-Treviño los tres algoritmos corresponden al reino animal, y corresponden uno a cada una de las familias con más desarrollos, aves, insectos y mamíferos.

Se presenta a continuación una breve reseña de la inspiración, reglas del movimiento del enjambre, formulación matemática y el diseño cada uno de estos algoritmos para la resolución del problema planteado. Finalmente se muestra un algoritmo híbrido desarrollado tras las conclusiones de la primera etapa experimental y las conclusiones del capítulo.

## 4.2 Optimización por enjambre de partículas (PSO – Particle Swarm Optimization)

### 4.2.1 Introducción a PSO

La optimización por enjambre o cúmulo de partículas (Particle Swarm Optimization PSO) es un método de optimización estocástica desarrollado por Kennedy y Eberhart (1995).

La abstracción del mecanismo descripto tras la observación del comportamiento de los individuos de ciertas especies de la naturaleza que actúan en conjunto, como bandadas de pájaros, algunos insectos, bancos de peces etc., que demuestran una inteligencia social y colectiva para conseguir sus objetivos (generalmente presas o fuentes de alimentación), inspiró al desarrollo de estos algoritmos de optimización que permiten resolver problemas de la vida real.

Estos grupos de individuos o partículas interconectadas, se enlazan, interactúan y se comunican de manera confiable entre sí moviéndose en línea recta, circular o serpenteante usando direcciones de búsqueda o gradientes (Okwu y Tartibu, 2021). El algoritmo PSO utiliza partículas establecidas que “vuelan”, “nadan” o se “desplazan” sobre un espacio de búsqueda para la ubicación óptima global. A lo largo del proceso de iteración de PSO, cada partícula actualiza su ubicación en función del conocimiento o experiencia anterior y de la experiencia del resto de los individuos del enjambre.

Dicho de otro modo, en la búsqueda y movimiento de cada individuo del enjambre puede tomar decisiones basado en su conocimiento sobre el entorno, lo que se traduciría en su valor de fitness, los procesos cognitivos propios, y el conocimiento del resto de los individuos o conocimiento social. (Poli 2007) (Poli 2008)

En términos generales y según la revisión general presentada por (Wang *et al.,* 2017) PSO tiene una excelente robustez y se puede utilizar en diferentes entornos de aplicación con pequeñas modificaciones además de poder paralelizarse e hibridarse fácilmente para converger al valor de optimización en forma eficiente.

El algoritmo PSO está ganando popularidad en el campo de la inteligencia computacional y se ha aplicado con éxito a una variedad de problemas de optimización y búsqueda. PSO es uno de los algoritmos de enjambre más utilizados y mejor valorados en la literatura de inteligencia computacional, metaheurísticas y optimización. El algoritmo ha sido ampliamente utilizado en una variedad de campos como la ciencia y la ingeniería (Machesa *et al.,* 2019). Las redes de datos no

quedan excluidas y hay suficientes antecedentes de su utilización en este campo es suficiente que justifican la elección de PSO como una de las posibles herramientas para la solución del problema planteado en el capítulo 3.

Yousif (2021) propone un algoritmo híbrido de enjambre de partículas para resolver el problema de congestión de una red MPLS. En (Masood *et al.,* 2019-1) los autores realizaron una mejora del algoritmo PSO básico para la optimización multibojetivo de redes MPLS/GMPLS. Existen también antecedentes de la utilización de PSO para la resolución de distintos problemas en SDN (Latah y Toker, 2018) como enrutamiento (Awad *et al.,* 2017), (Xiong *et al.,* 2017), gestión de recursos en 5g (Chang *et al.,* 2017), personalización de red virtual multiusuario (Li *et al.,* 2017), enrutamiento multiclase (Abdulqadder *et al.,* 2017) y detección de ataques DDoS (Dayal y Srivastava,2018). (Guo y Yuan, 2018) también utilizaron optimización por enjambre de partículas para la optimización del tráfico de una red definida por software.

### 4.2.2 Movimiento de las Partículas

En un determinado enjambre, las partículas tienen diferentes velocidades que las mueven en busca de una posición más favorable (una mejor solución). Las partículas intentan moverse desde la posición actual hacia la mejor posición local o global desplazándose por el espacio de búsqueda.

Como en todo movimiento físico, existe una inercia que intenta que el movimiento continúe en la misma dirección en que se movían previamente, pero también existe una “aceleración” que hará cambiar tanto la dirección como la magnitud de la velocidad. Ésta dependerá de dos factores: en primer lugar, sufrirá una atracción hacia la mejor posición histórica que ha tenido (mejor solución de la partícula) y en segundo lugar la partícula es atraída hacia la mejor posición encontrada por todo el enjambre en el espacio de búsqueda (mejor solución global).

La figura 4.2 ilustra una representación vectorial en el espacio del movimiento de las partículas.

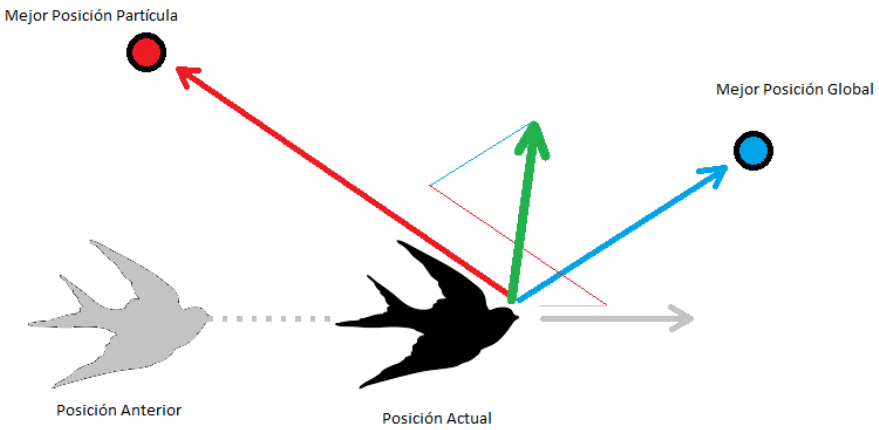


*Figura 4.2 Representación vectorial del movimiento de las partículas*

La flecha verde representa la nueva velocidad que surge como una sumatoria vectorial de los vectores de velocidad previa (vector gris) afectado por un factor de inercia, el vector de atracción hacia la mejor posición conocida de la partícula (vector rojo) afectada por un factor de aprendizaje individual y el vector de atracción hacia la mejor posición del enjambre (vector azul) afectada por un factor de aprendizaje social.

Una vez que la partícula alcance la nueva posición se actualizarán los tres vectores y se podrá calcular un nuevo vector de dirección para el próximo posicionamiento de la partícula, y de esta manera el conjunto de partículas se moverá por el espacio de soluciones hasta que se considere que se alcanzó una solución de buena calidad.

### 4.2.3 Formulación matemática

La formulación matemática que sintetiza en movimiento de las partículas se establece de acuerdo a las siguientes ecuaciones:

La posición de la partícula está dada por:

$$x_i^t = x_i^{t-1} + v_i^t \tag{4.1}$$

Donde:
$x_i^t$ *y* $x_i^{t-1}$ representan las posiciones de la partícula *i* en el espacio de búsqueda en los tiempos *t* y *t-1 respectivamente.*
$v_i^t$ es la velocidad de la partícula *i* en el instante *t,*

y la velocidad puede calcularse como:

$$v_i^t = w * v_i^{t-1} + c_1 * r_1^t * \left(p_i^{best} - x_i(t-1)\right) + c_2 * r_2^t * (G_{Best} - x_i(t-1)) \tag{4.2}$$

Donde:
$v_i^t$ es la velocidad de la partícula i en el instante t.
$w$ es el factor de inercia
$c_1$ es el factor de aprendizaje individual
$c_2$ es el factor de aprendizaje social
$r_1^t$ *y* $r_2^t$ son valores aleatorios en el rango [0,1] que se renuevan en cada tiempo t.
$p_i^{best}$ representa la mejor posición conocida de la partícula i
$G_{Best}$ representa la mejor posición conocida por el enjambre (Óptimo Global)

Con la manipulación de los factores $c_1$, $c_2$, y *w* pueden obtenerse distintas variantes del método, que permiten mejorar el comportamiento exploratorio para el problema que se pretende solucionar. En términos generales se puede afirmar que cuanto mayor es $c_1$ respecto de $c_2$ las partículas tienen mayor independencia, lo que permite una mayor exploración a costa de una convergencia lenta, por el contrario, valores grandes de c2 respecto de c1 favorecen una convergencia rápida ya que las partículas son empujadas a moverse hacia la mejor región descubierta hasta el momento por el enjambre, pero con el peligro de caer en un óptimo local.

Los valores aleatorios $r_1^t$ *y* $r_2^t$ también juegan un papel importante en el desempeño del algoritmo, e introducen variaciones en la técnica, pudiéndose manejar como vectores o escalares. El hecho de multiplicar cada componente de la velocidad por valores aleatorios distintos puede mejorar el poder exploratorio del algoritmo ya que añade mayores fluctuaciones al movimiento de las partículas, lo que, aun a riesgo de retrasar la convergencia, suele generar mejores resultados. (Amat Rodrigo, 2019).

### 4.2.4 Diseño del Algoritmo

Si bien existe en el estado del arte un sinfín de variaciones al algoritmo, dependiente en general del problema a resolver, lo interesante del modelo clásico planteado es que cada partícula se mueve con ecuaciones muy básicas y de bajo costo computacional (Ec. 4.1 y 4.2) por lo que la

complejidad del proceso recae en la evaluación de la aptitud o fitness de cada partícula en cada iteración.

Para el problema planteado, la solución a encontrar y por consiguiente el vector posición *x,* es el descripto en la ecuación 3.25, un vector formado por tantos subvectores como requerimientos se tengan. Cada sub vector tendrá tantas componentes como caminos alternativos tengan cada uno de esos requerimientos. El vector de velocidades *v* tendrá las mismas dimensiones, en tanto los factores de aceleración, aprendizaje social e individual se trabajarán como escalares.

Para cada partícula se iterará un número de veces que será determinado en forma empírica, generando el movimiento del enjambre. Una vez aplicada las ecuaciones para encontrar una nueva posición de las partículas, y dada la naturaleza no continua del espacio de búsqueda, se procede a realizar una reparación de la solución, ya que los elementos del vector solución no pueden ser negativos y se considerarán solamente valores enteros.

El proceso de reparación consta de los siguientes pasos:

1- Poner en cero todas las componentes negativas del vector
2- Redondear todas las componentes del vector a un valor entero
3- Las componentes del vector que sobrepasen la demanda total de cada requerimiento se reducirán al valor de la demanda.
4- Si faltan unidades de flujo para alcanzar el flujo demandado, se sumarán de a una unidad al camino que menos flujo asignado tenga.
5- Si se sobrepasa el flujo demandado, se restará de a una unidad de flujo al camino que tenga mayor carga de flujo asignado.

Los pasos 1 y 2 garantizan que las componentes del vector sean componentes válidas (valores enteros mayores o iguales a cero), y los pasos 4 y 5 son los encargados de cumplimentar la condición de demanda de cada requerimiento (Ecuación 3.7) a la vez que su forma operativa intenta mantener un balanceo de cargas, finalmente el paso 3 solo realiza una simplificación que acelera el paso 5.

Finalmente, el cálculo del fitness de cada solución será el que se encarga de penalizar las soluciones que no cumplan con la restricción de la capacidad de los enlaces. (Ecuación 3.8)

Se muestra a continuación el pseudocódigo del algoritmo.

Algoritmo PSO

***Cargar parámetros iniciales de la estructura de la red a optimizar*** *(Grafo, requerimientos, etc)*
***Definir:***
*Cantidad de Partículas*
*Número de iteraciones*
*Factor de aprendizaje individual (c1) y aprendizaje social (c2)*
*Factor de inercia (w)*
***Definir posición xi de cada partícula*** (aleatoriamente)
***Calcular el Fitness de cada partícula*** $p_i^{best}$
***Determinar el mínimo Fitness del enjambre*** $G_{Best}$
***Desde movimiento=1 hasta Nº Iteraciones***
***Para cada partícula i***
Generar los valores aleatorios $r_1$ y $r_2$
Calcular la velocidad de la partícula
Calcular la posición de la partícula
Reparar la posición para encontrar una solución
Calcular el Fitness de la nueva solución

***Si Fitness solución actual < Fitness*** $p_i^{best}$
*Actualizar* $p_i^{best}$
***Si Fitness solución actual < Fitness*** $G_{Best}$
*Actualizar* $G_{Best}$
***Fin si***
***Fin si***
***Siguiente Partícula***
***Siguiente iteración***
Mejor solución=$G_{Best}$
Mostrar la mejor solución

# 4.3 Optimización por colonia de hormigas (ACO – Ant Colony Optimization)

## 4.3.1 Introducción

Las hormigas son insectos sociales que viven en colonias y que han desarrollado un sistema de colaboración mutua que les permiten desarrollar tareas complejas.

Entre las habilidades dadas por su accionar colaborativo, se encuentra la de encontrar los caminos más cortos hacia las fuentes de alimento, este hecho toma aún más relevancia si consideramos que muchas de las especies de hormigas tienen un sentido de la visión totalmente disminuido.

Durante el proceso de exploración del territorio en busca de alimento, las hormigas se mueven estableciendo caminos al azar de un lugar a otro desde el hormiguero hasta la fuente de alimentación. En ese movimiento, algunas especies de hormigas, (Dorigo *et al.,* 2006), depositan una sustancia química denominada feromona (una sustancia que puede “olerse").

Ante la falta de rastros de feromona, las hormigas se moverán en forma aleatoria, pero si detectan un rastro de feromona en el camino, tenderán a seguir el rastro. Cuando varios caminos se entrecruzan las hormigas tenderán a seguir aquel camino con mayor cantidad de feromona, a mayor cantidad de feromona, mayor será la probabilidad de elegir ese camino. Este comportamiento lleva a un proceso de retroalimentación que concluye con la formación de rastros señalados por una concentración de feromona elevada (Deneubourg y Pasteels 1983)

Según toma Dominguez Medina (2011) de (Hopkin 2004) este comportamiento permite a la colonia de hormigas encontrar los caminos más cortos entre su hormiguero y la fuente de alimento. Las hormigas solo se comunican de manera indirecta, a través de las modificaciones del espacio físico que perciben. Esta forma de comunicación se denomina estimergia

El rastro de feromonas es la razón principal por la que las hormigas son capaces de formar o mantener una línea. Mientras viajan, las hormigas liberan una cierta cantidad de feromona que proporciona un camino para que otras hormigas lo sigan. Este instinto básico de las hormigas explica por qué generalmente buscan el camino más corto incluso en presencia de un obstáculo que dificulta un camino inicial. Por defecto, las hormigas viajan en línea recta desde el nido (N) hasta la fuente de alimento (S). En caso de que haya un obstáculo como se muestra en la Fig. 4.3, las hormigas líderes tendrían que elegir entre el camino izquierdo o derecho. Se puede suponer a partir de la investigación que la mitad de Las hormigas dan vuelta a la izquierda y el resto a la derecha. Las hormigas que siguieron el camino más corto reconstituirían el rastro de feromonas más rápidamente en comparación con las que optaron por el camino más largo. Se

deposita una mayor cantidad de feromonas en la ruta más corta por unidad de tiempo, lo que resulta en una mayor probabilidad de que las hormigas elijan esto camino más corto (Dorigo *et al.,* 2006) (Fig. 4.4)

*Figura 4.3 Camino típico alrededor de un obstáculo*

*Figura 4.4 Cantidad de rastro de feromonas según el camino de las hormigas*

Las hormigas pueden localizar fuentes de alimento alrededor de su nido sin ninguna información espacial y encuentran el camino más corto que conduzca a él. Cuando una hormiga encuentra comida, regresa a su nido dejando rastros de feromonas como se muestra en la Fig. 4.5 (Okwu y Tartibu, 2021). Este rastro de feromonas será seguido por las otras hormigas que son menos susceptibles de viajar al azar. A medida que más hormigas encuentran comida, el camino se reforzará con más feromonas como se muestra en la Fig. 4.5; sin embargo, debido al comportamiento estocástico, algunas hormigas no seguirán el rastro de feromonas producido por hormigas anteriores y en su lugar descubrirán más caminos posibles como se muestra en la Fig. 4.5. Con el tiempo, la feromona que queda en los senderos que no se siguen se evaporará, lo que reduce la fuerza del atractivo, y debido a ello se mantendrá el camino más corto. (Okwu y Tartibu, 2021)


*Figura 4.5 Ilustración de camino de hormigas y senderos de feromonas.*

Cuanto más tarden las hormigas en viajar de un lado a otro a lo largo de un camino específico, menor la densidad de feromonas como resultado de la evaporación. Lo que evitará que las hormigas utilicen estos caminos.

Inspirado en el comportamiento anteriormente explicado y su semejanza con problemas cotidianos, es que se desarrollaron y aplicaron los algoritmos de optimización bio-inspirados en hormigas conocidos como ACO (Ant colony optimization) en diferentes problemas.

ACO fue propuesta como una nueva Metaheurística en 1999 (Dorigo y DiCaro 1999) con resultados prometedores y aunque no resultó competitivo como algoritmos de optimización combinatoria para algunos problemas conocidos en ese momento, ha recibido una creciente atención de la comunidad científica, existiendo en la actualidad varias versiones del algoritmo y con diversas aplicaciones (Domínguez Medina, 2011)

En cuanto a la aplicación en redes de datos, ACO ha sido ampliamente utilizada, desde que DiCaro y Dorigo en 1997 aplicaron su propia técnica como algoritmo de ruteo dinámico. ACO se ha utilizado en mecanismos de control de congestión (Rajagopalan *et al.,* 2011)(Naganathan *et al.,* 2014); se reportan resultados de ACO aplicados al logro de políticas de enrutamiento multicapa completamente distribuidas en redes IP / MPLS a través de redes ópticas (Amorin y Pavoni 2019); se ha aplicado en enrutamiento energéticamente eficiente a través de MPLS-MANET (Ambika y Banga 2021) y se encuentran también varios antecedentes de uso en redes definidas por Software (SDN) aplicando ACO desde distintas ópticas y recopiladas por (Latah y Toker, 2018) (Dobrijevic *et al.,* 2015)(Wang *et al.,* 2017)(Gao *et al.,* 2016)(Di Stefano *et al.,* 2015)(Wang *et al.,* 2016)(Parsaei *et al.,* 2017)(Chen *et al.,* 2016)(Liu *et al.,* 2017)(Yi *et al.,* 2016).

Todos estos antecedentes motivan a la selección de esta técnica para el problema propuesto.

## 4.3.2 Formulación Matemática

La formulación matemática básica y que origina la técnica es simple. Frente a una toma de decisión (una bifurcación en el camino) las hormigas deberán escoger un camino para continuar su viaje. Para describir la función matemática es posible analizarlo de manera análoga el típico problema de ruteo donde se quiere encontrar la ruta más corta entre un nodo origen y un nodo destino de una red (un grafo). Supóngase que, desde un nodo intermedio (*r*) del camino, se desea continuar hacia el próximo nodo, la probabilidad para que una hormiga *k* en el nodo *r* seleccione un nodo de destino *s* en el próximo salto, se describe por la siguiente ecuación (Dorigo y Dicaro 1999):

$$p_k(r,s) = \begin{cases} \dfrac{\tau_{rs}^{\alpha} * \eta_{rs}^{\beta}}{\sum_{uM_k}\left(\tau_{rs}^{\alpha} * \eta_{rs}^{\beta}\right)} & \text{Para } s \in M_k \\ 0 & \text{en otro caso} \end{cases} \tag{4.3}$$

Donde:
$\tau_{rs}^{\alpha}$ es la feromona del enlace *r-s*
$\alpha$ es el peso de feromona $\tau$
β es el peso del valor heurístico η
$M_k$ representa el conjunto de nodos conectados
$\eta_{rs}$ es la información heurística

La relación entre los valores $\alpha$ y β será la que incline la probabilidad en función de la feromona o la información heurística η, en la toma de una decisión. η es una función que depende del entorno que asigna a cada paso de la construcción un valor heurístico para cada componente de solución factible. Por ejemplo, para el caso planteado podría ser la capacidad del enlace entre *r* y *s*, considerando que, a mayor capacidad, mayor probabilidad de ser elegido. Este parámetro puede ser totalmente independiente del camino que recorran las hormigas y depender solamente del entorno (en este caso los parámetros de la red).

Una vez concluido el viaje, el "Tour" seguido por la hormiga se constituye en una solución sobre la que puede considerarse la aplicación de acciones auxiliares necesarias para adaptar la solución y tomar decisiones sobre como actualizar el rastro de feromona. Este rastro está sujeto a cambios tanto a nivel local como global y es almacenado en una matriz $\boldsymbol{\tau}$ que guarda el nivel de feromona de cada enlace *(i, j)* del grafo. Si $\Delta\tau$ representa la cantidad de feromona depositada por la hormiga en cada segmento visitado, es posible calcular la cantidad de feromona asociada a cada enlace *r-s* mediante la ecuación 4.4.

$$\tau(r,s)^t = \tau(r,s)^{t-1} + \Delta\tau \tag{4.4}$$

Donde:
$\tau(r,s)^t$ es la feromona del enlace r-s en el instante t
$\Delta\tau$ es el rastro de feromona que deja la hormiga.

$\Delta\tau$ deberá ser mayor a medida que la solución es mejor, es decir, las mejores hormigas dejarán un rastro mayor que las hormigas cuya solución es de mala calidad, generalmente para los problemas de minimización como el caso propuesto $\Delta\tau$ se puede elegir como la inversa de la longitud del camino. En términos generales, para problemas de minimización, el $\Delta\tau$ suele calcularse como:

$$\Delta\tau = \frac{c}{fitness} \tag{4.5}$$

Donde:
$c$ es una constante, generalmente 1
$fitness$ es el valor de aptitud de la solución (o de la hormiga).

A medida que las hormigas recorren el camino los segmentos elegidos por las mismas incrementan su valor $\tau$. En general, los enlaces que son utilizadas por muchas hormigas y los de recorridos de longitud pequeña recibirán más feromona y por lo tanto serán más frecuentemente elegidas en las iteraciones futuras del algoritmo. (Domínguez Medina, 2011)

A lo largo del tiempo e independientemente de los valores de feromonas depositados por las hormigas, y debido al desempeño estocástico de la técnica, puede producir que alguna hormiga tome caminos alternativos a la mayoría, lo cual si bien es deseable con el fin de magnificar el poder exploratorio y no estancarse en óptimos locales es necesario que si la opción no es buena el rastro se evapore para no confundir la decisión de futuros individuos.

La evaporación se realiza en forma global a todos los enlaces y puede representarse matemáticamente como:

$$\tau(r,s)^t = (1-\rho)\tau(r,s)^{t-1} \tag{4.6}$$

Donde:
$\rho$ es una constante perteneciente al intervalo [0 1]

Al igual que la técnica de optimización por enjambre de partículas, ACO ha recibido cientos de modificaciones y adaptaciones que mejoraban el desempeño de la técnica para la optimización de distintos problemas particulares, pero desde esta fundamentación matemática se presentarán los algoritmos para dar una alternativa de solución al problema planteado en el capítulo anterior.

### 4.3.3 Diseño del Algoritmo

La metaheurística de optimización por colonia de hormigas permite que sus propios miembros (hormigas) tengan una memoria basada en su experiencia y en su comportamiento (debida al depósito de feromona), lo que les permite en conjunto tener un comportamiento dinámico que les facilite encontrar rutas más cortas rápidamente (Cruz H y Viteri 2007), lo cual sería aplicable si no conociéramos los caminos. Es decir, si consideremos un grafo como el descripto en el capítulo anterior para describir la red MPLS, en cada iteración del algoritmo, cada hormiga *k* construye un recorrido completo $M_k$, (desde el origen al destino) creándose por lo tanto un conjunto de recorridos $M = \{M_k\}$. Para la construcción de cada uno de esos recorridos, la hormiga parte del nodo origen y selecciona iterativamente otro nodo hasta completar un recorrido válido. Cuando estemos en el nodo *i*, $N_i$ representa el conjunto de nodos que pueden seleccionarse como próximo salto en la transmisión; la probabilidad de elegir el nodo *j* desde el nodo *i* está dada por la ecuación 4.2 donde *r=i* y *s=j*. Este es básicamente el principio del algoritmo AntNet creado por Dorigo y DiCaro en 1998, y que ha tenido excelentes resultados.

Pero en este caso, los caminos posibles son conocidos, y el problema radica en como distribuir el flujo sobre esos caminos, por lo que no sirve utilizar esa lógica en el algoritmo. De todos modos, la filosofía de ACO puede plasmarse en un algoritmo que permita encontrar la solución al problema propuesto. En ese sentido se propone el diseño de dos alternativas de algoritmos inspirados en ACO, un primer algoritmo que distribuirá el flujo requerido en los distintos caminos sobre la base de la probabilidad ACO, construyendo la solución, y un segundo algoritmo que parte de una solución aleatoria e intenta mejorar su fitness realizando cambios de porciones de flujo entre un camino y otro, a través de una redistribución de flujos. Se detallan a continuación ambos algoritmos:

#### 4.3.3.1 Algoritmo ACO 1 – Distribución de flujos basado en ACO

Dado un conjunto de requerimientos *T*, cada "hormiga" tomará al azar de a un requerimiento por vez para hacer la asignación, lo que expande la capacidad exploratoria del algoritmo. De otro modo, considerando el problema planteado en la sección 3.5, si todas las hormigas tomaran el requerimiento en el mismo orden, existe una mayor probabilidad que el requerimiento 1 ocupe determinados caminos que el requerimiento 3.

Una vez tomado el requerimiento, para cada unidad de flujo demandado, la "hormiga" elegirá por qué camino enviarlo. Esta determinación se realiza de acuerdo a la probabilidad que cada camino tenga calculada en función de la ecuación 4.4. Es decir que el "Tour" de la hormiga se constituirá en la solución buscada.

La información heurística de cada camino *η(p)* se almacena en un vector de las mismas dimensiones de la solución, donde cada componente representa la información correspondiente a cada camino de cada requerimiento y se calculará como:

$$\eta(\mathrm{p}) = \frac{CapU_p(\mathrm{p})}{Costo(p)} \tag{4.7}$$

Donde:
$CapU_p(p)$ corresponde a la capacidad útil del camino *p.*
$\mathrm{Costo}(\mathrm{p})$ costo del camino *p (Ec.3.3)*

Debido a que distintos caminos pueden compartir ciertos enlaces, la capacidad útil de un determinado camino se calculará como

$$CapU_p(p) = \min\{Cap_e - Carga_e\} \; \forall \; e \in p \tag{4.8}$$

Donde:
$Cap_e$ corresponde a la capacidad útil del enlace *e* del camino *p.*
$Carga_e$ carga del enlace *e* del camino *p*

La utilización de la ecuación 4.7 garantiza por un lado que se penalicen los enlaces de mayor costo con menos probabilidad de ser elegidos, y por otro lado que se prioricen los enlaces con mayor capacidad útil.

Como $\eta=0$ cuando el canal se encuentre ocupado en su máxima capacidad, no existirá necesidad de revisar la restricción de los enlaces impuesta por la ecuación 3.8, ya que no existirá posibilidad de asignar flujo a un enlace que se encuentre saturado.

Por otro lado, la feromona $\tau$, se representará de la misma manera, con un vector de la misma dimensión que la solución, y representará la feromona depositada por cada hormiga en un determinado camino *p,* para todos los caminos de todos los requerimientos. Al comienzo el vector se inicializa con un valor $\tau_0$ que constituye un parámetro del algoritmo y su actualización se realizará de acuerdo al siguiente criterio:

$$\tau(p)^t = \tau(p)^{t-1} + \frac{x_p}{Fitness} \tag{4.9}$$

Donde:
$\tau(p)^t$ corresponde a la feromona del camino *p* en el instante *t*.
$x_p$ corresponde a la cantidad de flujo asignado al camino p
$Fitness$ corresponde al valor de aptitud de la solución construida.

De alguna manera esta actualización del vector de feromona garantiza que el enjambre tenga en cuenta no solo la selección de los caminos sino la distribución en magnitud de flujo que transportará cada uno.

Se detalla a continuación el pseudocódigo del algoritmo que aclarará el funcionamiento completo.

Algoritmo ACO 1

***Cargar parámetros iniciales de la estructura de la red a optimizar*** *(Grafo, requerimientos, etc)*
***Definir Parámetros ACO:***
*Cantidad de hormigas*
*Número de iteraciones*
*α - β - ρ - $\tau_0$*
***Desde iteración=1 hasta Nº Iteraciones***
***Desde k=1 hasta Cantidad de hormigas***
req=Permuto{requerimientos}
***Para cada requerimiento req***
***Para flujoasignado=1 hasta Demanda(req)***
Actualizar η
Calcular la Probabilidad de los caminos
Seleccionar un camino en función de la probabilidad (Método de la ruleta)
Incrementar en 1 el flujo del camino seleccionado
***Siguiente iteración***

***Siguiente requerimiento***
Calcular el Fitness de la solución generada por la hormiga k
***Si Fitness solución actual < Fitness de la Mejor Solución***
*Actualizar la Mejor Solución*
***Fin si***
***Siguiente hormiga***
***Desde k=1 hasta Cantidad de hormigas***
Calcular $\Delta\boldsymbol{\tau}$ de la hormiga k
Actualizar el vector $\boldsymbol{\tau}$
***Siguiente hormiga***
Calcular la evaporación y actualizar el vector $\boldsymbol{\tau}$
***Siguiente iteración***
Mostrar la Mejor solución

### 4.3.3.2 Algoritmo ACO 2 – Redistribución de flujos basado en ACO

Este segundo algoritmo, si bien sigue la estrategia ACO, tiene un enfoque totalmente distinto al anterior, partiendo de una solución generada aleatoriamente (un vector solución) al cual cada hormiga le sumará un vector de operadores de la misma longitud de la solución y cuyos componentes pertenecen al conjunto {-1 0 1}. Esto significa que a algunas componentes del vector solución se le restará una unidad de flujos, a otras se le sumará y a otras quedarán inalteradas.

El "tour" de cada hormiga consiste en determinar la mejor secuencia de estos operadores de forma de ir mejorando gradualmente la solución hasta encontrar un óptimo satisfactorio. Frente a la selección del operador para cada componente del vector de operadores, se utiliza la ecuación de probabilidad ACO (Ec.4.3).

En esta ecuación, la matriz de feromona $\boldsymbol{\tau}$ tendrá 3 filas y *n* columnas donde *n* es la dimensión de la solución y contendrá la feromona correspondiente a los operadores -1, 0 y +1 en las filas 1, 2 y 3 respectivamente.

Retomando el ejemplo de la sección 3.5 donde la solución tenía 9 componentes la matriz será:

$$\tau = \begin{bmatrix} \tau_{x1}^{-1} & \tau_{x2}^{-1} & \tau_{x3}^{-1} & \tau_{x9}^{-1} \\ \tau_{x1}^{0} & \tau_{x2}^{0} & \tau_{x3}^{0} \ldots & \tau_{x9}^{0} \\ \tau_{x1}^{+1} & \tau_{x2}^{+1} & \tau_{x3}^{+1} & \tau_{x9}^{+1} \end{bmatrix} \tag{4.10}$$

Donde:
$\tau_{xi}^{-1}$ corresponde a la feromona para restar 1 a la componente *xi*
$\tau_{xi}^{0}$ corresponde a la feromona para no alterar a la componente *xi*
$\tau_{xi}^{+1}$ corresponde a la feromona para sumar 1 a la componente *xi*

La información heurística η para la selección de cada operador, también será una matriz de *3* filas x *n columnas*, y se encargará de colaborar en la construcción de la solución y el cumplimiento de la demanda de cada requerimiento. Y se construirá de la siguiente manera:

$$\eta(1,j) = \begin{cases} 1 \; \forall \; j \; donde \; x(j) \; \neq 0 \\ 0 \; \forall \; j \; donde \; x(j) = 0 \end{cases} \tag{4.11}$$

$$\eta(2,j) = 1 \; \forall \; j \tag{4.12}$$

$$\eta(3,j) = \begin{cases} 1 \; \forall \; j \; donde \; x(j) < demanda \\ 0 \; \forall \; j \; donde \; x(j) = demanda \end{cases} \quad (4.13)$$

Donde:

$\eta(i,j)$ corresponde a la información heurística para operar (i=1 restar, i=2 no alterar, i=3 sumar) en la componente *j* de la solución.

$x(j)$ corresponde a la componente *j* de la solución a la que se le aplicarán los operadores.

La ecuación 4.11 que representa la construcción de la primera fila de la matriz de información heurística η garantiza que ningún elemento de la solución sea negativo, ya que al ser cero una componente de la solución, el η hará nula la probabilidad de seleccionar un -1 como operador. Del mismo modo, la ecuación 4.13 garantiza que ninguna componente del vector solución supere las unidades de flujo demandadas por un requerimiento ya que al alcanzar la demanda el η anulará la probabilidad de sumar una unidad de flujo. La ecuación 4.12 por su parte, garantiza que $\boldsymbol{\tau}$ y η tengan las mismas dimensiones, facilitando las operaciones algorítmicas.

Si bien la construcción de η garantiza que cada camino individualmente no supere la demanda, no asegura que la suma de los flujos de los distintos caminos se equilibre con la correspondiente demanda, por lo que será necesario una reparación de la solución posterior a su generación para que esta solución sea factible.

Considérese el requerimiento 1 del problema planteado. El flujo requerido de 10 unidades deberá repartirse en los 3 caminos posibles para dicho requerimiento, a modo de ejemplo supongamos la distribución [4 4 2]. Los tres operadores que afectarán a estas componentes de la solución podrán ser cualesquiera de las 27 combinaciones posibles de los elementos {-1 0 1}. La suma de las componentes del subvector de operadores correspondientes al requerimiento indicarán si es necesaria o no una reparación de la solución. Si la suma es igual a cero significa que algunas componentes se sumaron y otras se restaron, pero no se alteraron la cantidad total de unidades de flujo repartidas en el camino. Por ejemplo, si se tiene la distribución de operadores [0 -1 1] la nueva solución será [4 3 3]. En el caso que la sumatoria de los operadores sea distinto de 0 habrá que compensar el exceso o defecto para satisfacer la restricción de demanda.

La reparación lleva a cabo con la misma lógica utilizada por algoritmo, si es necesario restar o sumar unidades de flujo, estas operaciones se realizan sobre los caminos elegidos de acuerdo con la probabilidad que resulte de aplicar la ecuación 4.3. El valor de η será actualizado en cada acción de reparación con la misma lógica descripta anteriormente.

Una vez calculadas las secuencias de operadores (Tours) de las hormigas y calculadas las soluciones a partir de la solución inicial se procede a actualizar la matriz de feromona de acuerdo a:

$$\tau(i,j)^t = \tau(i,j)^{t-1} + \frac{1}{Fitness} \; \forall \; solución \quad (4.14)$$

Donde:

$\tau(i,j)^t$ corresponde a la componente *(i,j)* de la matriz feromona en el instante *t*.

*i* representa el índice operador utilizado (1 para -1, 2 para 0 y 3 para1)

*j* representa la componente a la cual se le aplicó ese operador.

$Fitness$ corresponde al valor de aptitud de la solución construida luego de aplicar los operadores.

Tras las primeras pruebas del algoritmo, se detectó una tendencia al estancamiento en óptimos locales, que se agravaba conforme crecía el tamaño de la instancia. Para paliar este problema se adoptó un sistema de reinicio de la feromona con el fin de dispersar a las hormigas de sus caminos. Esta estrategia resulta efectiva, de muy sencilla implementación y le da la oportunidad al algoritmo de salir del óptimo local y reiniciar su búsqueda, pero ahora partiendo con una solución con mejor calidad.

Se muestra a continuación el pseudocódigo del algoritmo.

Algoritmo ACO 2

***Cargar parámetros iniciales de la estructura de la red a optimizar*** *(Grafo, requerimientos, etc)*
***Definir Parámetros ACO:***
    *Cantidad de hormigas*
    *Número de iteraciones*
    *Número de iteraciones para reiniciar*
    $\alpha$ - $\beta$ - $\rho$ - $\tau_0$
*Generar una solución aleatoria y adoptarla como mejor solución*
***Desde iteración=1 hasta Nº Iteraciones***
    ***Desde k=1 hasta Cantidad de hormigas***
        Actualizar η
        ***Desde operador=1 hasta Dimensión de la Solución***
            Calcular la Probabilidad de los operadores (-1 , 0 , 1)
            Seleccionar un operador en función de la probabilidad (Método de la ruleta)
        ***Siguiente operador***
        Calcular la nueva solución como la Mejor Solución + el vector de operadores
        Reparar la solución en función de $\tau$ y η
        Calcular el Fitness de la solución generada por la hormiga k
        ***Si Fitness solución actual < Fitness de la Mejor Solución***
            *Actualizo la Mejor Solución*
            *Contador de iteraciones sin mejora=0*
        ***Si no***
            *Incremento un contador de iteraciones sin mejora*
            ***Si Iteraciones sin mejora=Iteraciones para reiniciar*** $\tau$
                $\tau = \tau_0$
                *Contador de iteraciones sin mejora=0*
            ***Fin Si***
        ***Fin si***
    ***Siguiente hormiga***
    ***Desde k=1 hasta Cantidad de hormigas***
        Calcular $\Delta\tau$ de la hormiga k
        Actualizar el vector $\tau$
    ***Siguiente hormiga***
    Calcular la evaporación y actualizar el vector $\tau$
***Siguiente iteración***
Mostrar la Mejor solución

## 4.4 Optimización por Algoritmo de Murciélagos (BA – Bat Algorithm)

### 4.4.1 Introducción

Existen en la actualidad más de 1000 especies distintas de murciélagos, interesantes animales que utilizan un sistema de ecolocalización como sonar emitiendo pulsos ultrasónicos y escuchando el eco que se recupera al rebotar en los objetos del medio que los rodea con el fin de encontrar su sustento. Basado en el comportamiento de un conjunto de estos pequeños animales (específicamente los micro murciélagos) y encuadrando en el conjunto de metaheurísticas basado

en población o enjambres, se encuentra el algoritmo de murciélago (Bat Algorithm, BA), una técnica propuesta por Yang (2010) donde se toma la inspiración de estos animales en búsqueda de alimento utilizando su capacidad de ecolocalización (*EL*) para encontrar la solución a problemas de optimización.

Este algoritmo puede utilizarse para resolver problemas complejos en diversos campos de operaciones que van desde la ingeniería, los negocios, el transporte y otros campos de la actividad humana (Okwu y Tartibu, 2021).

Es importante comprender el patrón de comunicación y navegación de los murciélagos mientras se define el algoritmo. (Yang, 2010). El sistema de Ecolocalización *(EL)* se observa como una onda de sonido que producen los murciélagos que no solo utilizan en la búsqueda de alimentos, sino que sirve también para otros fines (Yang, 2011), por ejemplo, mientras busca comida en un ambiente completamente oscuro, le permite detectar obstáculos o peligros en su camino.

El algoritmo de murciélago (BA) es un algoritmo poderoso, ya que utiliza la metodología de sintonización de frecuencia para la intensificación del rango de las soluciones en la población. Al mismo tiempo, el sistema BA implementa el procedimiento instintivo de disparar o hacer zoom para mantener la estabilidad durante el proceso de búsqueda o exploración de alimentos. Al desarrollar el BA, el patrón de búsqueda de dirección, manipulación y exploración durante la caza de presas se tiene en cuenta al imitar las disparidades en términos de tasas de emisión de pulso e intensidad del sonido liberado por los murciélagos mientras cazan o buscan presas. El algoritmo demuestra un alto nivel de eficiencia con un proceso de inicio rápido (Okwu y Tartibu, 2021).

Si bien la técnica ha recibido algunos cuestionamientos (Gagnon *et al.,* 2020), hay reportes de su utilización para la toma rápida de decisiones y la resolución de problemas complejos en diferentes campos de aplicación (Okwu y Tartibu, 2021).

En resumen, el algoritmo de murciélago fue presentado como una metaheurística novedosa, inspirada en el mecanismo de ecolocalización de los quirópteros y, según su autor, generalmente superior a la Optimización de Enjambre de Partículas y los Algoritmos Genéticos (GA). Gagnon *et al.,* (2020) ha cuestionado tal afirmación sosteniendo que, en realidad, no es un modelo perfecto y sugiriendo que resulta un híbrido entre las técnicas de optimización por enjambre de partículas y recocido simulado.

Independientemente de las críticas que ha recibido el método, existen algunos antecedentes de uso de este algoritmo en el ámbito de las redes de datos, que motivan su aplicación para resolver el problema de la asignación de flujos planteado en el capítulo anterior. Se ha aplicado a redes GMPLS para minimizar el costo de ruteo (Masood *et al.,* 2017), para optimización de recursos y selección de ruta óptima (Masood *et al.,* 2018), y para el balanceo de cargas (Glesk, *et al.,* 2018). Alkasassbeha, *et al.,* 2018 trabajó para implementar redes MPLS con SDN. Lin y Ye (2018) propusieron la utilización de un algoritmo basado en BAT para asignación de recursos en redes definidas por software (SDN) y Sathya y Thangarajan (2015) emplearon un algoritmo de murciélago binario (BBA) para la selección de características en un sistema de detección de intrusiones basado en SDN.

### 4.2.2 Reglas del movimiento de los murciélagos virtuales

Los murciélagos emiten sonidos de diferente amplitud y frecuencia, escuchan el eco reflejado por los objetos circundantes y pueden distinguir la presa de otros obstáculos y predadores. A medida que se acerca a la presa, el volumen del sonido disminuye y la frecuencia de los pulsos emitidos aumenta. El patrón de vuelo de los animales es complejo y la técnica de optimización es una simplificación de estas observaciones de la naturaleza, aunque, aun así, la metáfora sigue siendo válida

Para simular el comportamiento por ecolocalización y usarla en el diseño del algoritmo, Yang (2010) propuso una serie de reglas idealizadas:

- Todos los murciélagos emplean ecolocalización para percibir, detectar o discernir presas, barreras, depredadores y todo tipo de obstáculos de una forma “mágica” no comprensible para el hombre.

- En el curso de búsqueda los murciélagos vuelan al azar a una cierta velocidad $v_i$ en una posición $x_i$ emitiendo ultrasonido con frecuencia en un rango [$f_{min}$, $f_{max}$] con una tasa de pulsos $r_i$ y con un volumen máximo $A_0$ que puede variar hasta un valor mínimo $A_{min}$

- A medida que se acercan a la presa, disminuyen el volumen y aumentan la tasa de pulsos.

### 4.2.3 Formulación matemática

Para la aplicación del algoritmo es necesario definir los pasos y formulación matemática que gobiernan el comportamiento de los quirópteros.

En primer lugar, se genera una población aleatoria de “murciélagos virtuales”, la cual se dota un volumen aleatorio con un máximo preestablecido ($A_0$), una tasa de pulsos ($r_i$) aleatoria comprendida entre [1,0], y se inicializa la frecuencia y velocidad en 0.

El proceso será iterativo y en cada iteración se irá ajustando la posición de todos los murciélagos con el fin de encontrar una solución.

Para simular una nueva posición (nueva solución potencial), en cada iteración deben ajustarse los valores de frecuencia y velocidad para calcular una nueva posición.

$$f_i = f_{min} + (f_{max} - f_{min})\,\beta \qquad (4.15)$$

$$v_i^t = w * \; v_i^{t-1} + (x_i^t - x^*) f_i \qquad (4.16)$$

$$x_i^t = x_i^{t-1} + v_i^t \qquad (4.17)$$

Donde:
$x_i$ representa la posición del murciélago en el espacio de búsqueda
$v_i$ es la velocidad
$f_i$ la frecuencia de los pulsos emitidos

*x** es la mejor solución obtenida hasta ese momento
β es un valor aleatorio en el rango (0,1).
*w* es un factor de inercia.

Las dimensiones del problema de optimización determinarán los valores límites de la frecuencia.

En cada iteración todos los murciélagos de la población se mueven usando sus nuevas velocidades, acorde con las ecuaciones 4.15, 4.16 y 4.17.

Una vez calculadas todas las nuevas posiciones de los "murciélagos virtuales" se procederá a actualizar el *x** si es que alguna de las nuevas posiciones alcanzadas es superadora.

Posteriormente se procede a generar una nueva solución localmente (Caminata aleatoria) por cada murciélago:

$$x_{nuevo}^{t} = x^{t} + \epsilon A^{t} \qquad (4.18)$$

Donde:
$\epsilon$ es un valor aleatorio comprendido entre [-1,1]
$A^t$ es el promedio del volumen de todos los murciélagos en este paso.

A medida que van transcurriendo las generaciones, cuando un individuo se acerca a la solución *x** el volumen del sonido (*A*) debe disminuir y la tasa de pulsos ($r_i$) se debe incrementar lo cual, si consideramos para un determinado tiempo *t*, se representa matemáticamente como sigue:

$$A_i^t = \alpha A_i^{t-1} \qquad (4.19)$$

$$r_i^t = r_i^0\left[1 - exp\left(-\gamma(t-1)\right)\right] \qquad (4.20)$$

Donde:
$r_i^0$ representa la tasa de pulsos inicial.
$\alpha$ y $\gamma$ son constantes y parámetros del problema.

Para un valor $0 < \alpha < 1$ y $\gamma > 0$ podemos afirmar que:

$$A_i^t \to 0,\ r_i^t \to r_i^0\ \ cuando\ t \to \infty \qquad (4.21)$$

## 4.2.4 Diseño del Algoritmo

Como se ha explicado anteriormente, la solución es un vector de valores enteros positivos coincidente en este caso con la posición de los murciélagos, es decir, que la posición de cada murciélago constará de un vector n-dimensional donde *n* corresponde a la dimensión de la solución. Del mismo modo, la frecuencia y la velocidad serán n-dimensionales.

El algoritmo procederá a general una población aleatoria sobre la cual operará las ecuaciones 4.15 a 4.17 para encontrar una nueva solución. Debido a la naturaleza de dichas ecuaciones, el resultado de su aplicación generará un vector n-dimensional de componentes reales, las cuales deberá ser reparada para que pueda constituirse en una solución. Esta reparación se llevará a cabo de la misma forma que se realizó en el algoritmo PSO (Sección 4.2.4).

Con algunas pequeñas variaciones, el diseño del algoritmo se basa en la formulación matemática, y se detalla a través del pseudocódigo mostrado a continuación.

Algoritmo BA

***Cargar parámetros iniciales de la estructura de la red a optimizar*** *(Grafo, requerimientos, etc)*
***Definir parámetros BA:***
    *Cantidad de Murciélagos*
    *Número de iteraciones*
    *Frecuencia Mínima y Máxima*
    $\alpha$ y $\gamma$
***Definir posición xi de cada murciélago*** (generada aleatoriamente)
***Calcular el Fitness*** para cada murciélago en la posición $x_i$ y encontrar la mejor solución ($x^*$)
***Definir*** para cada murciélago los valores de $\boldsymbol{A_i}$ ***y*** $\boldsymbol{r_i}$ en forma aleatoria
***Desde movimiento=1 hasta Nº Iteraciones***
    ***Para cada murciélago***
        Crear nueva posición (usando ec 4.15 a 4.17)
        Seleccionar un valor aleatorio *rnd*
        ***Si rnd>ri***
            *Cambiar la posición con caminata aleatoria*
            $X_{nuevo} = X_{nuevo} + \varepsilon * A_{prom}$ (con $\varepsilon$ entre [-1,1])
        ***Fin_si***
        Seleccionar otro valor aleatorio *rnd*
        ***Si rnd<Ai & f(xi)<f(x*)***
            *Aceptar la nueva posición*
            incrementar $r_i$
            reducir *Ai*
        ***Fin_si***
        *Calcular el fitness*
        ***Si*** $\boldsymbol{Fitness_{movimiento\ (i)} < Fitness_{movimiento(i-1)}}$
            *Aceptar la nueva posición*
            incrementar $r_i$
            reducir *Ai*
            ***Si Fitness< Fitness**** *(mejoró el fitness global)*
                *Guardar el índice del mejor*
                *Actualizar la Mejor Solución* ($x^*$)
            ***Fin si***
        ***Fin si***
    ***Siguiente Murciélago***
***Siguiente iteración***
Mostrar la Mejor solución

# 4.5 Algoritmo híbrido murciélago - hormiga (BPA – BA post-optimizado con ACO)

## 4.5.1 Introducción

Es común que los métodos heurísticos y meta-heurísticos no se utilicen en forma única para optimizar un problema, sino que es posible unir más de un método para lograr un algoritmo con mayor eficiencia al momento de encontrar una solución a un determinado problema.

Luego de la experimentación realizada con los algoritmos descriptos, y cuyos resultados se exponen en el capítulo 5, pudo observarse que el algoritmo de murciélago tenía la capacidad de llegar a soluciones óptimas actuando con mayor rapidez que sus pares. Sin embargo, frente a instancias de redes grandes, cuando los espacios de búsqueda toman un tamaño considerable, crece la probabilidad de quedar atrapado en óptimos locales.

Por su parte, el algoritmo basado en colonia de hormigas que se encarga de redistribuir los flujos entre los caminos (ACO 2) demostró durante la experimentación tener aparentes capacidades de sortear los óptimos locales, pero a costa de una convergencia lenta, siendo necesaria una cantidad de iteraciones elevadas, sobre todo cuando las soluciones iniciales generadas en forma aleatoria tienen valores de aptitud grandes.

Con este análisis posterior a algunos ensayos con distintas instancias del problema, se decide hibridar el algoritmo inspirado en murciélagos con el algoritmo inspirado en hormigas para potenciar por un lado la velocidad del algoritmo de murciélagos, pero intentando sortear los óptimos locales en los que este suele estancarse con una estrategia de redistribución de flujos basada en ACO.

La formulación matemática e inspiración de estos algoritmos son las ya explicadas por lo que se describirá a continuación el diseño del algoritmo.

## 4.5.2 Diseño del algoritmo

En sentido estricto no es exactamente un algoritmo híbrido, sino que podría pensarse como un algoritmo inspirado en murciélagos con un post-optimizador inspirado en ACO, o dicho de otro modo se ejecutará en primer lugar un algoritmo de murciélagos y a la solución encontrada, la cual se supone buena calidad, se le aplicará una algoritmo inspirado en colonia de hormigas que reasignará los flujos entre los caminos con el fin de escapar de óptimos locales y lograr mejorar los valores de aptitud de dicha solución.

El algoritmo denominado BPA (BA post-optimizado ACO) es una conjunción con las adaptaciones necesarias de los algoritmos ya descriptos. Se presenta el pseudocódigo a continuación.

Algoritmo BPA

***Cargar parámetros iniciales de la estructura de la red a optimizar*** *(Grafo, requerimientos, etc)*
***Definir Parámetros BA:***
    *Cantidad de Murciélagos*
    *Número de Movimientos de los murciélagos*
    *Frecuencia Mínima y Máxima*
    $\alpha_{BA}$ y $\gamma$
***Definir Parámetros ACO:***
    *Cantidad de hormigas*
    *Número de iteraciones ACO*
    $\alpha_{aco}$ - $\beta$ - $\rho$ - $\tau_0$
***Definir posición xi de cada murciélago*** (generada aleatoriamente)
***Calcular el Fitness*** para cada murciélago en la posición $x_i$ y encontrar la mejor solución ($x^*$)
***Definir*** para cada murciélago los valores de $A_i$ ***y*** $r_i$ en forma aleatoria
***Desde movimiento=1 hasta Nº Movimientos de los murciélagos***
    ***Para cada murciélago***
        Crear nueva posición (usando ec 4.15 a 4.17)
        Seleccionar un valor aleatorio *rnd*
        ***Si rnd>ri***
            *Cambiar la posición con caminata aleatoria*
            $X_{nuevo} = X_{nuevo} + \varepsilon * A_{prom}$ (con $\varepsilon$ entre [-1,1])
        ***Fin_si***
        Seleccionar otro valor aleatorio *rnd*
        ***Si rnd<Ai & f(xi)<f(x*)***
            *Aceptar la nueva posición*
            incrementar $r_i$
            reducir *Ai*
        ***Fin_si***

*Calcular el fitness*
***Si Fitness**$_{movimiento\ (i)}$ < **Fitness**$_{movimiento(i-1)}$*
*Aceptar la nueva posición*
incrementar $r_i$
reducir *Ai*
***Si Fitness< Fitness*** *(mejoró el fitness global)*
*Guardar el índice del mejor*
*Actualizar la Mejor Solución* ($x^*$)
***Fin si***
***Fin si***
***Siguiente Murciélago***
***Siguiente iteración***
***Desde iteración=1 hasta Nº Iteraciones ACO***
***Desde k=1 hasta Cantidad de hormigas***
Actualizar η
***Desde operador=1 hasta Dimensión de la Solución***
Calcular la Probabilidad de los operadores (-1 , 0 , 1)
Seleccionar un operador en función de la probabilidad (Método de la ruleta)
***Siguiente operador***
Calcular la nueva solución como la Mejor Solución + el vector de operadores
Reparar la solución en función de $\boldsymbol{\tau}$ y η
Calcular el Fitness de la solución generada por la hormiga k
***Si Fitness solución actual < Fitness de la Mejor Solución***
*Actualizar la Mejor Solución*
***Fin si***
***Siguiente hormiga***
***Desde k=1 hasta Cantidad de hormigas***
Calcular $\Delta\boldsymbol{\tau}$ de la hormiga k
Actualizar el vector $\boldsymbol{\tau}$
***Siguiente hormiga***
Calcular la evaporación y actualizar el vector $\boldsymbol{\tau}$
***Siguiente iteración***
Mostrar la Mejor solución

## 4.6 Conclusiones del capítulo

En este capítulo se ha presentado una taxonomía de estrategias heurísticas y se ha seleccionado tres de ellas para realizar la adaptación e implementación de algoritmos para resolver el problema de la asignación óptima de flujos en redes MPLS. Se optó por analizar el desempeño de algoritmos bioinspirados y se han realizado cuatro adaptaciones de las estrategias propuestas y una hibridación del algoritmo de murciélagos conjuntamente con uno basado en colonia de hormigas.

El algoritmo basado en enjambre de partículas al igual que basado en murciélagos son similares en cuanto a la estrategia algorítmica ya que la posición de las partículas o murciélagos representan la solución de optimización. Dichas posiciones, se representan mediante vectores *n*-dimensionales coincidentes con el tamaño del vector solución. Los algoritmos difieren en la forma de moverse o desplazarse a lo largo del espacio de búsqueda, mientras PSO tiene en cuenta la atracción hacia buenas soluciones globales y particulares, BA, actúa en función de la comunicación lograda con distintos niveles de volumen, frecuencia y pulsos que se adaptan acorde a la aptitud de la posición encontrada. Estos valores servirán de guía para actualizar las posiciones de los murciélagos virtuales.

En cuanto a la estrategia de optimización por colonia de hormiga se presentan dos alternativas, una de distribución (ACO 1) y otra de redistribución (ACO 2) de flujos. En la primera el “tour” n-dimensional de las hormigas se construye distribuyendo flujos sobre los caminos

posibles constituyéndose en una posible solución. La segunda estrategia parte de una solución inicial generada en forma aleatoria y las hormigas virtuales se encargarán de encontrar los operadores necesarios para redistribuir el flujo de manejar de mejorar dicha solución.

Finalmente se presenta una alternativa de hibridación entre el algoritmo BA y ACO 2, diseñado tras el análisis de comportamiento de los algoritmos en las primeras pruebas realizadas que intenta aprovechar la velocidad de BA y el poder de escapar de los óptimos locales mediante la estrategia ACO 2.

En el capítulo 5 se muestran los resultados obtenidos por cada implementación y para diferentes tamaños de instancias del problema planteado.

# Capítulo 5
# Experimentación y Resultados

## 5.1 Introducción

En el capítulo 4 se presentaron las bases teóricas y metodológicas para la aplicación de estrategias heurísticas inspiradas en la naturaleza, seleccionadas para servir de base al desarrollo de algoritmos que permitan realizar la asignación óptima de flujos en una red multiservicio.

En este capítulo se muestran los resultados obtenidos por cada implementación teniendo en cuenta diferentes escenarios que consideran instancias del problema que van de menor a mayor tamaño. Se comparan los resultados obtenidos por las diferentes implementaciones en cada uno de los casos de estudio.

Los algoritmos planteados en el capítulo 4 podrían codificarse fácilmente en muchos lenguajes de programación para proceder a la experimentación y ensayo con distintas instancias del problema. En ingeniería, uno de los lenguajes más utilizados es Matlab, un software de cálculo numérico que ofrece un ambiente para realizar operaciones aritméticas y matriciales con mucha potencialidad. Matlab ha sido utilizado en la implementación de algoritmos de optimización, debido a su facilidad de programación y su poder de cómputo en el manejo de vectores y matrices además de contar con herramientas que facilitan tareas desarrolladas sobre grafos.

Una vez determinado el lenguaje de programación, es necesario definir redes conocidas con el fin de entender la precisión y exactitud de los algoritmos a la hora de encontrar el óptimo global.

Las redes a optimizar estarán definidas algorítmicamente por tres matrices: la matriz de adyacencia, la matriz de costo de los enlaces y la matriz de capacidad de los enlaces. La formulación del problema se completa con la matriz de requerimientos.

Las primeras pruebas de los algoritmos se llevaron a cabo sobre una red conocida, ya utilizada en el problema planteado en el capítulo 3, una red de cuatro nodos sobre los cuales se establecieron tres requerimientos y cuyos detalles se muestran a continuación.

La matriz de Adyacencia está dada por la ecuación 5.1 y la figura 5.1 representa la topología de la red descripta por dicha matriz.

$$A = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix} \qquad (5.1)$$

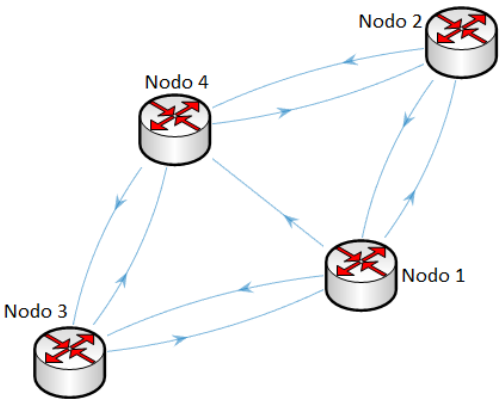


*Figura 5.1 – Topología de la red de ensayo. Caso 1: Red de 4 nodos.*

La matriz de requerimientos "T" se expresa como:

$$T = \begin{bmatrix} 1 & 2 & 10 \\ 2 & 3 & 10 \\ 1 & 4 & 20 \end{bmatrix} \tag{5.2}$$

Lo que significa que el primer requerimiento especifica una reserva de 10 unidades de flujo desde el nodo 1 hasta el nodo 2, un segundo requerimiento desde el nodo 2 hasta el nodo 3 con 10 unidades de flujo y finalmente desde el nodo 1 hasta el nodo 4 existe un requerimiento con 20 unidades de flujo.

Los algoritmos están diseñados para cualquier costo y capacidad de los enlaces y se representan a través de matrices de las mismas dimensiones que la matriz de adyacencia. Para los ejemplos planteados, todos los enlaces tendrán costo unitario (1) y una capacidad máxima de 15 unidades de flujo, con un costo mínimo para distribuir los tres requerimientos de 55 unidades.

La cantidad total de caminos posibles por cada requerimiento puede ser representada mediante un vector cuyas componentes coinciden con la cantidad de caminos posibles para rutear cada uno de los requerimientos (definido en el capítulo 3 como $L_t$), es decir un vector de 3 componente para este caso, y que calculados los caminos posibles será:

$$Cantidad\ de\ Caminos = [3, 3, 3] \tag{5.3}$$

Cada una de las componentes del vector *Cantidad de Caminos* representa el tamaño del subvector solución para un determinado requerimiento; es decir que la suma de la cantidad de caminos será coincidente con el tamaño del vector solución **x**.

Con esta red que denominada Caso 1 – Red de 4 nodos, se realizaron los primeros ajustes y puesta a punto de todos los algoritmos, por cuanto el tamaño de la instancia permite la obtención rápida de resultados, la agilidad en el ajuste de parámetros y en la determinación de su influencia sobre el desempeño de los algoritmos.

No obstante, una de las características esperable en el diseño es la escalabilidad. Por lo tanto, para continuar con la experimentación, y analizar el desempeño de las estrategias seleccionadas, es necesario definir redes de mayores tamaños. Se procede a definir 3 redes más, de 6, 8 y 10 nodos respectivamente las cuales se realizan a partir de ampliaciones de la red del caso 1, sin modificar los enlaces comprometidos en los tres requerimientos, con ello se logran topologías con mayor complejidad, donde se aumenta considerablemente el número de opciones

de caminos para repartir los flujos, aunque se conserva el valor de mínimo costo de la primera red considerada. De esta manera, es posible realizar un análisis de desempeño de las estrategias propuestas en esta tesis.

Las topologías de las redes consideradas son representadas mediante la matriz de adyacencia, y siempre se respetan todos los enlaces con un costo unitario y una capacidad máxima de 15 unidades al igual que en el primer caso. Los detalles de las topologías se muestran a continuación:

**Caso 2:** Red de 6 nodos, con la siguiente matriz de adyacencia y la topología representada en la figura 5.2

$$A = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 \end{bmatrix} \tag{5.4}$$

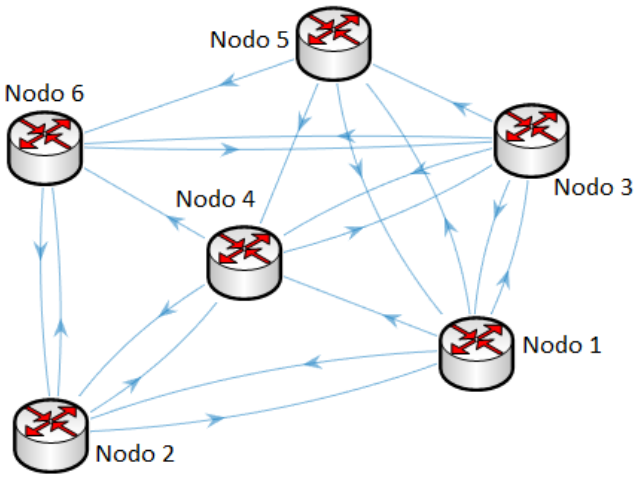


*Figura 5.2 – Topología de la red de ensayo Caso 2: Red de 6 nodos.*

Para los mismos requerimientos planteados en la ecuación 5.2 y con la ayuda del software se calculan la cantidad de caminos por requerimiento:

$$Cantidad\ de\ Caminos = [16,9,11] \tag{5.5}$$

Por lo que el vector solución **x** tendrá una dimensión de 36 componentes.

**Caso 3:** Red de 8 nodos, cuya topología se representa en la figura 5.3 y la matriz de adyacencia viene dada por:

$$A = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix} \quad (5.6)$$

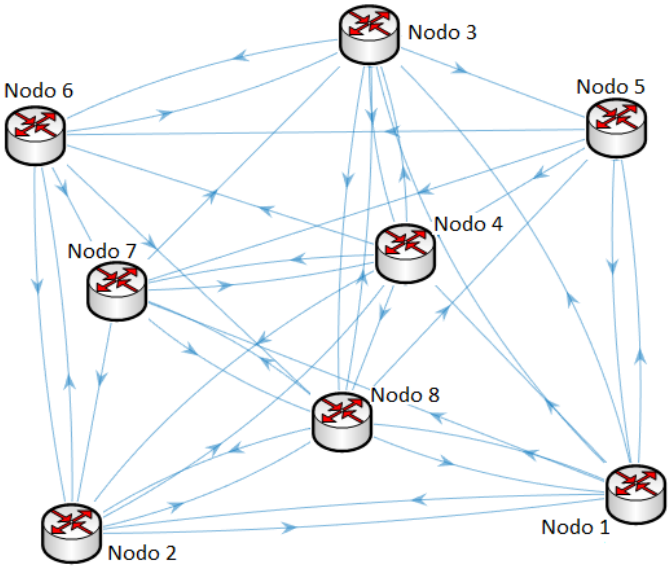


*Figura 5.3 – Topología de la red ensayo Caso 3: Red de 8 nodos.*

Para los mismos requerimientos planteados en 5.2 la cantidad de caminos por requerimiento es:

$$Cantidad\ de\ Caminos = [289, 170, 179] \quad (5.7)$$

El vector solución **x** tendrá una dimensión de 638 componentes.

**Caso 4:** Red de 10 nodos, cuya topología se puede observar en la figura 5.4 y su matriz de adyacencia viene dada por:

$$A = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \end{bmatrix} \quad (5.8)$$

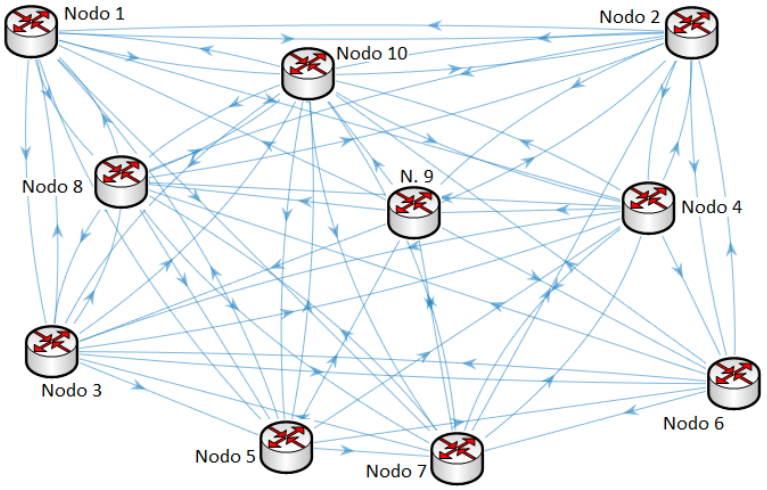


*Figura 5.4 – Topología de la red ensayo Caso 4 Red de 10 nodos.*

Para los mismos requerimientos planteados en la ecuación 5.2 se calculan la cantidad de caminos por requerimiento:

$$Cantidad\ de\ Caminos = [5807, 5025, 4016] \tag{5.9}$$

Si bien no existe una relación entre la cantidad de nodos y la cantidad de caminos, sino que estos dependerán de la interconexión de los nodos, se observa, para los ejemplos planteados, un crecimiento abrupto de la cantidad de caminos y por consiguiente de la dimensión del vector solución **x**, que en este caso tendrá 14848 componentes.

En los cuatro casos planteados la matriz de requerimiento será la misma, el costo de los enlaces será unitario y la capacidad máxima será de 15 unidades en todos los enlaces de todas las redes, y debido a la forma en que fueron conformadas las redes de mayor tamaño, la solución óptima en todos los casos tendrá el valor 55 lo que permitirá validar los resultados.

Para el ensayo de las distintas estrategias seleccionadas, se consideraron, en primera instancia, los valores de los parámetros reportados en la bibliografía con el objeto de establecer una línea base respecto a su desempeño, para posteriormente llevar a cabo los ensayos que se describen a continuación.

Se diseñaron herramientas gráficas con el fin de ver el “movimiento” de los miembros del enjambre y entender la influencia de los parámetros.

Se expone a continuación la experimentación llevada a cabo con cada uno de los algoritmos en las distintas redes propuestas como problemas a optimizar, y el análisis comparativo.

Se resumen en este capítulo toda la experimentación, mostrando los distintos ensayos realizados para cada algoritmo. El orden en que se muestran los ensayos para cada estrategia aplicada a un caso determinado, no representa el orden en que necesariamente se realizaron, sino que este orden viene dado por la lectura de los archivos donde se guardan los ensayos.

## 5.2 Experimentación con CASO 1 – Red de 4 nodos

### 5.2.1 Ensayos del algoritmo PSO (Caso 1)

Con pocos ensayos es sencillo determinar los parámetros adecuados de configuración para que el algoritmo encuentre el óptimo global en tiempos aceptables. Con los ajustes necesarios se alcanza el óptimo global en el 100% de las pruebas y en pocas iteraciones.

En la siguiente tabla se muestran los parámetros utilizados para distintos ensayos de prueba.

| Ensayo | Cantidad de Partículas | Iteraciones | Factor de Aprendizaje individual *C1* | Factor de Aprendizaje Social *C2* | Factor de Inercia w | Efectividad |
|---|---|---|---|---|---|---|
| 1-PSO | 10 | 150 | 1 | 1 | 1 | 62% |
| 2-PSO | 10 | 150 | 1 | 0.5 | 1 | 94% |
| 3-PSO | 20 | 150 | 1 | 1 | 1 | 77% |
| 4-PSO | 20 | 150 | 2 | 1 | 1 | 95% |
| 5-PSO | 20 | 150 | 3 | 1 | 1 | 100% |
| 6-PSO | 20 | 150 | 5 | 1 | 1 | 100% |
| 7-PSO | 20 | 150 | 6 | 1 | 1 | 99% |
| 8-PSO | 20 | 150 | 7 | 1 | 1 | 99% |

*Tabla 5.1: Parámetros de configuración de ensayos PSO para el Caso 1*

Para cada ensayo se evaluó 100 veces el algoritmo sobre el problema propuesto, y se analizaron estadísticamente los resultados. La figura 5.5 muestra los resultados representados por gráficas de caja y bigote, de la experimentación.

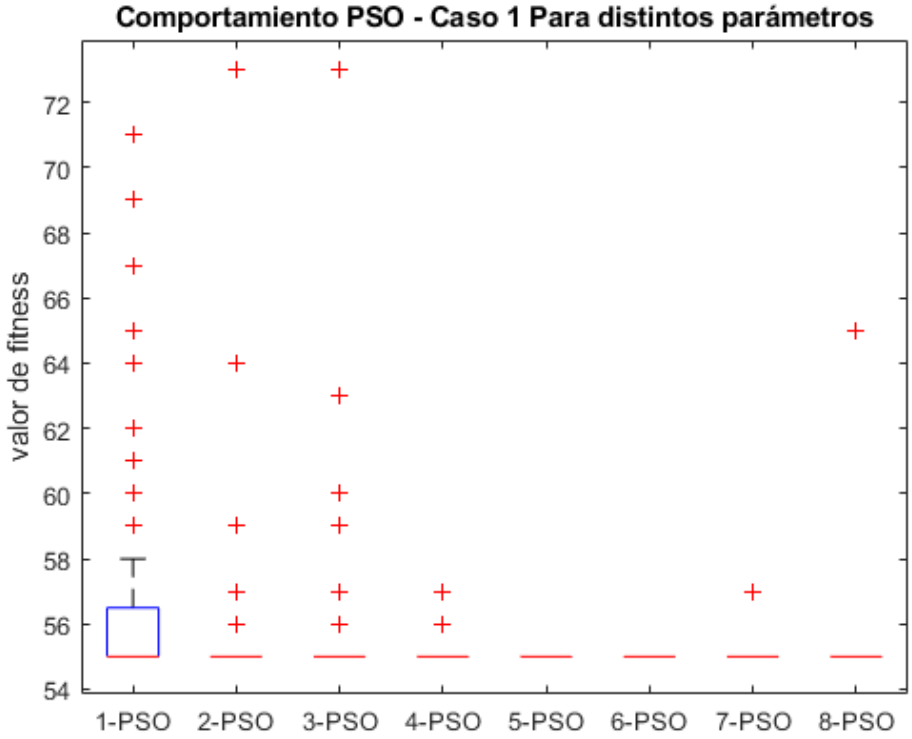


*Figura 5.5 – Ensayos PSO Caso 1 para distintos parámetros.*

Se observa que para valores de factores de aprendizaje individual mayores que el factor de aprendizaje social generan resultados con una eficiencia estadística del 100%. Se puede observar en el gráfico algunos datos atípicos que van disminuyendo a medida que se incrementa la relación $c_1/c_2$.

A partir de una relación $c_1/c_2$ de 3 a 1 la efectividad real es del 100 % desapareciendo los datos atípicos. La relación 5 a 1 aparenta ser el limite a partir de la cual el algoritmo presenta una disminución en su rendimiento real, aunque estadísticamente sigue siendo óptimo. Por otro lado, el parámetro de inercia no presenta mejoras al utilizar valores distintos de la unidad.

Los ensayos 5 y 6 son equivalentes estadísticamente, aunque el ensayo 6 presenta unas medias de tiempo para encontrar el óptimo global algo menores que el ensayo 5.

Todos los ensayos se realizaron con 150 iteraciones. Sin embargo, el óptimo global fue alcanzado antes de dicho límite establecido. El número de iteraciones necesarias para alcanzar dicho valor óptimo, con los parámetros del ensayo 6, puede observarse en la figura 5.6. Con 60 iteraciones del algoritmo se observa un desempeño óptimo, mientras que con 120 iteraciones se alcanzó el óptimo global en el 100% de los casos.

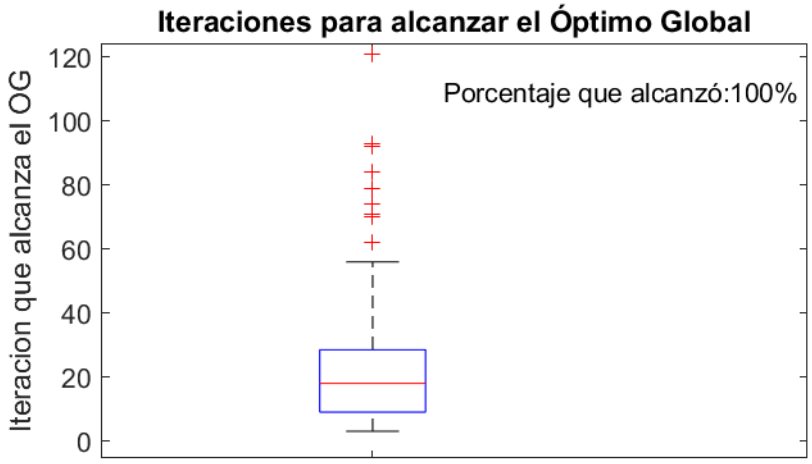


*Figura 5.6: Iteraciones para alcanzar el óptimo con algoritmo PSO para el Caso 1*

Los parámetros óptimos para el algoritmo PSO para el caso 1 se resumen en la tabla 5.2

| Cantidad de Partículas | Iteraciones | Factor de Aprendizaje individual *C1* | Factor de Aprendizaje Social *C2* | Factor de Inercia *w* |
|---|---|---|---|---|
| 20 | 120 | 5 | 1 | 1 |

*Tabla 5.2: Parámetros óptimos de configuración del algoritmo PSO para el Caso 1*

Los resultados estadísticos para los 100 ensayos realizados con dichos parámetros son:

| **Estadístico** | **Valor** |
|---|---|
| Mínimo | 55 |
| Efectividad | 100% |
| Media | 55 |
| Desviación Estándar | 0 |
| Varianza | 0 |
| Coeficiente de Variación | 0 |
| **Cuartiles** | |
| q1 | 55 |

| | |
|---|---|
| q2 | 55 |
| q3 | 55 |
| **Tabla de Frecuencias** | |
| Value Count Percent<br>55 100 100.00% | |

*Tabla 5.3: Datos estadísticos del algoritmo PSO para el Caso 1*

En la siguiente figura, se muestra la evolución del fitness para un ensayo típico del algoritmo aplicado a la red de cuatro nodos. El desarrollo se realiza con una pendiente relativamente constante con algunos estancamientos de pocas iteraciones.

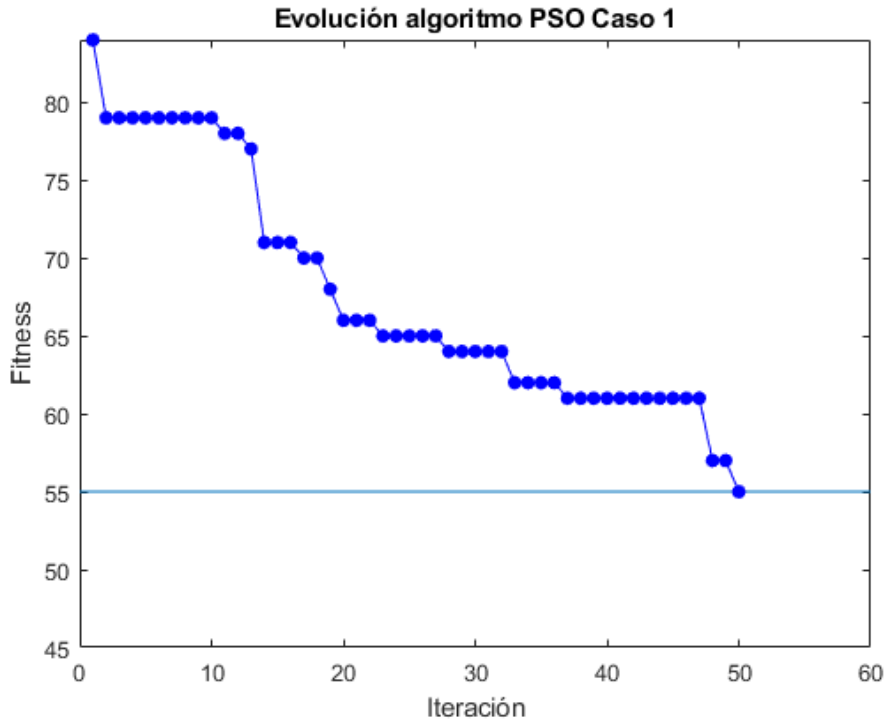


*Figura 5.7: Evolución típica del fitness para un ensayo del algoritmo PSO para el Caso 1*

## 5.2.2 Ensayos del algoritmo ACO 1 – Distribución de flujos (Caso 1)

Para el caso del algoritmo inspirado en colonia de hormigas que distribuye los flujos en los caminos, la determinación de los parámetros óptimos también resultó sencilla, y rápidamente se encontraron los valores que generan un buen desempeño del algoritmo. Se ajustaron estos parámetros con varios ensayos de los cuales de detallan los más significativos en la tabla siguiente.

| Ensayo | Cantidad de Hormigas | iteraciones | Parámetro Influencia feromona **α** | Parámetro Influencia valor heurístico **β** | Coeficiente de evaporación $\rho$ | Valor de Feromona inicial $\tau_0$ | Efectividad |
|---|---|---|---|---|---|---|---|
| 1-ACO1 | 20 | 500 | 3 | 1 | 0.01 | 0.1 | 89% |
| 2-ACO1 | 30 | 500 | 3 | 1 | 0.01 | 0.1 | 91% |
| 3-ACO1 | 40 | 500 | 3 | 1 | 0.01 | 0.1 | 85% |
| 4-ACO1 | 50 | 500 | 3 | 1 | 0.01 | 0.1 | 89% |
| 5-ACO1 | 50 | 500 | 2 | 0.5 | 0.009 | 0.1 | 96% |
| 6-ACO1 | 50 | 500 | 2 | 0.5 | 0.01 | 0.1 | 98% |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 7-ACO1 | 50 | 500 | 2 | 0.5 | 0.05 | 0.1 | 92% |
| 8-ACO1 | 50 | 500 | 5 | 1 | 0.01 | 0.1 | 80% |

*Tabla 5.4: Parámetros de configuración de ensayos ACO 1 para el Caso 1*

El grafico comparativo de los resultados se expone en la figura 5.8, donde se observa que los ensayos son estadísticamente equivalentes, diferenciándose por algunas soluciones atípicas.

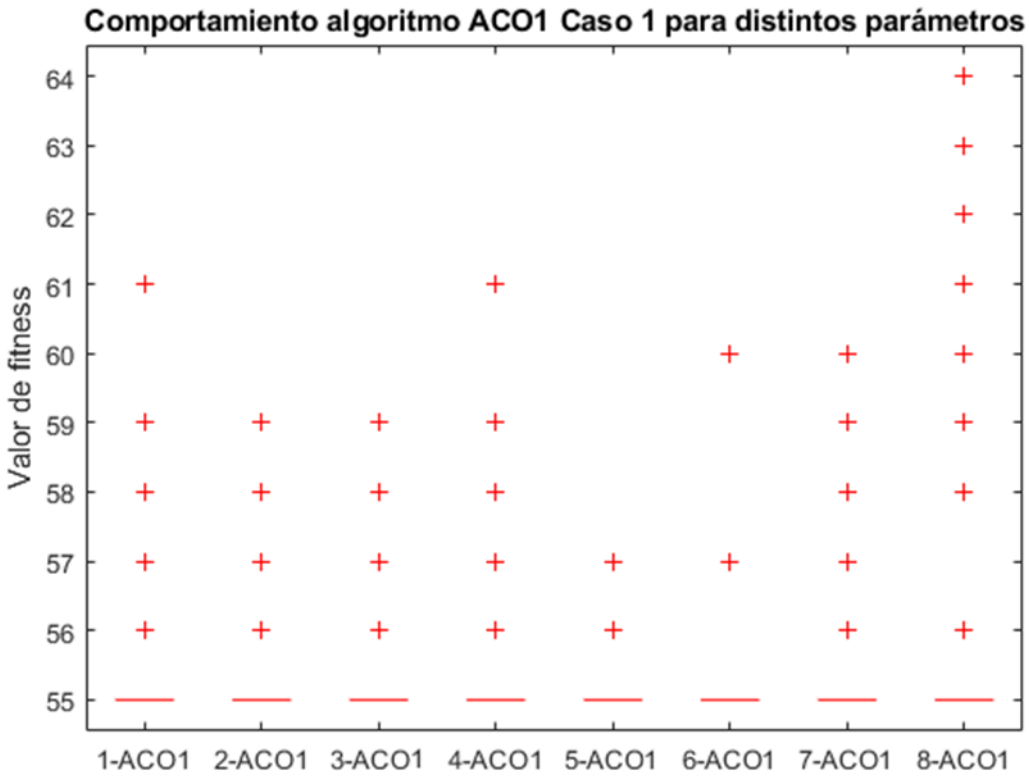


*Figura 5.8 – Ensayos ACO 1 Caso 1 para distintos parámetros.*

Mediante los ensayos se comprueba que el parámetro de influencia de feromona α, tiene que ser mayor al parámetro de influencia del valor heurístico para lograr buenos resultados. Variaciones del valor de $\tau_0$ no reflejan diferencias apreciables en los resultados.

Se analizaron los resultados estadísticamente y se concluye que la mejor configuración de parámetros corresponde al ensayo 5 el cual, si bien tiene un 2% menos de efectividad que otro ensayo con una evaporación de feromona mayor, sus soluciones estuvieron muy próximas al valor óptimo arrojando mejores valores estadísticos.

Los ensayos se realizaron con 500 iteraciones. En la siguiente figura, se muestran las iteraciones necesarias para alcanzar el óptimo global, para el ensayo 5. Con 180 iteraciones estadísticamente se alcanzaría el óptimo global, aunque quedan afuera algunos ensayos atípicos que no encuentran la mejor solución hasta las 250 iteraciones.

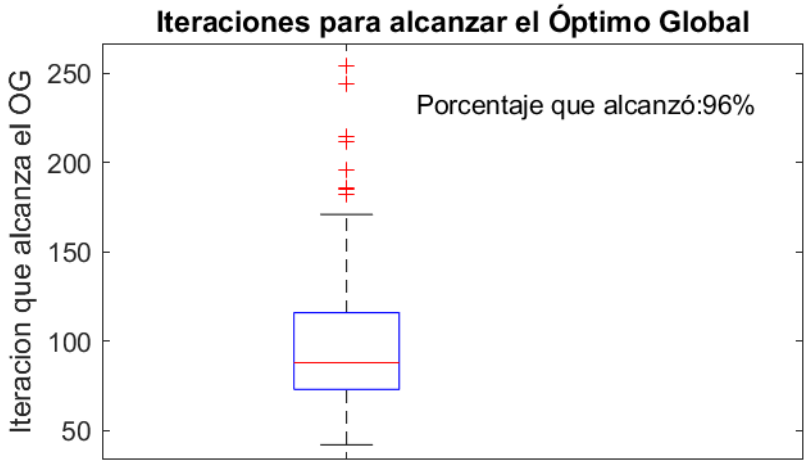


*Figura 5.9: Iteraciones para alcanzar el óptimo con algoritmo ACO 1 para el Caso 1*

En la tabla 5.5 se observan los parámetros óptimos y en la tabla 5.6 los valores estadísticos correspondiente a una ejecución de 100 ensayos con estos valores.

| Cantidad de Hormigas | iteraciones | Parámetro Influencia feromona **α** | Parámetro Influencia valor heurístico **β** | Coeficiente de evaporación $\rho$ | Valor de Feromona inicial $\tau_0$ |
|---|---|---|---|---|---|
| 50 | 250 | 2 | 0.5 | 0.009 | 0.1 |

*Tabla 5.5: Parámetros óptimos de configuración del algoritmo ACO 1 para el Caso 1*

| **Estadístico** | **Valor** |
|---|---|
| Mínimo | 55 |
| Efectividad | 96% |
| Media | 55.07 |
| Desviación Estándar | 0.355 |
| Varianza | 0.126 |
| Coeficiente de Variación | 0.0064 |
| **Cuartiles** | |
| q1 | 55 |
| q2 | 55 |
| q3 | 55 |
| **Tabla de Frecuencias** | |
| Value Count Percent<br>55 96 96.00%<br>56 1 1.00%<br>57 3 3.00% | |

*Tabla 5.6: Datos estadísticos del algoritmo ACO 1 para el Caso 1*

La distribución de soluciones se representa en la siguiente figura, donde se observa la representación de la tabla de frecuencias para los ensayos realizados con los parámetros óptimos. Los valores encontrados como atípicos son tan solo una y dos unidades mayores que el valor óptimo.

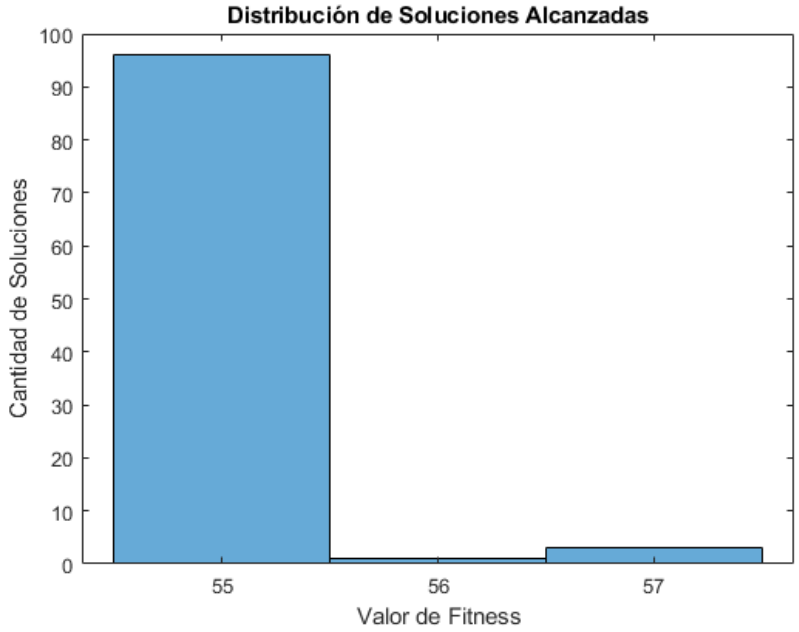


*Figura 5.10: Distribución de soluciones del algoritmo ACO 1 para el Caso 1*

Durante la evolución del algoritmo se producen estancamientos generando una evolución escalonada, y por lo cual se requiere una mayor cantidad de iteraciones, con el consiguiente aumento de costo computacional asociado, que será analizará en la sección 5.6. Una ejecución típica donde se alcanza el óptimo global tiene la siguiente evolución

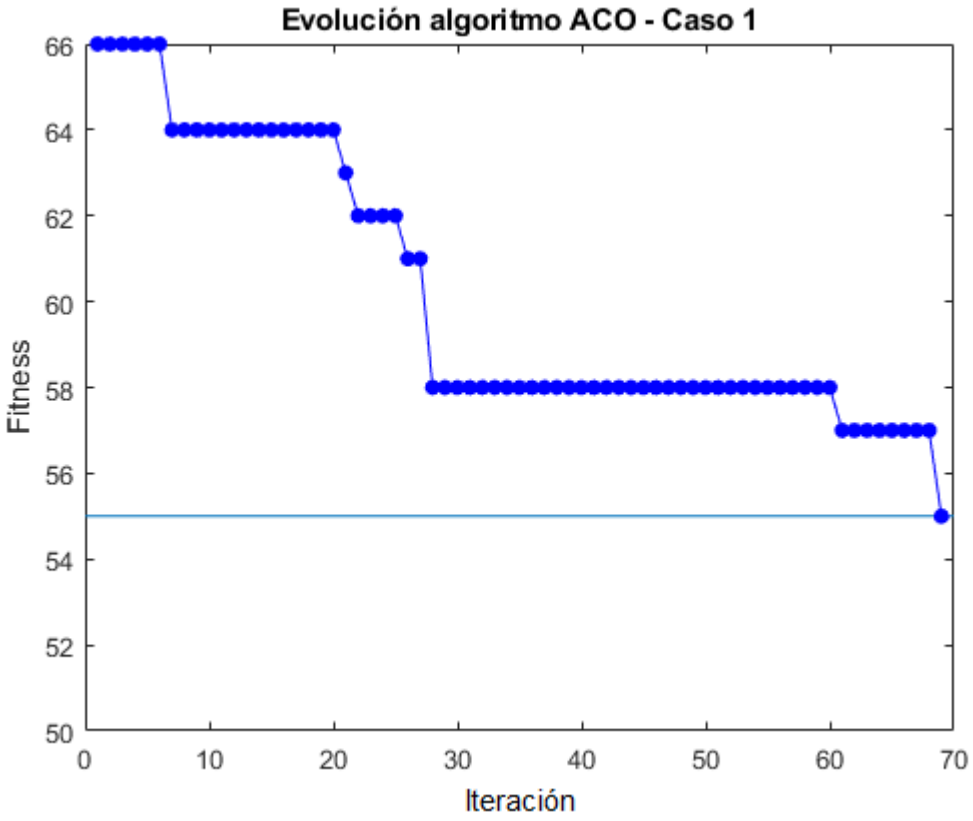


*Figura 5.11: Evolución típica del fitness para un ensayo del algoritmo ACO 1 para el Caso 1*

### 5.2.3 Ensayos del algoritmo ACO 2 – Redistribución de flujos (Caso 1)

Se realizaron varios ensayos con distintos parámetros para encontrar el mejor desempeño del algoritmo para este problema. El factor β se establece en 1 ya que los valores posibles de η, son 0 o 1 actuando como anulador de la probabilidad cuando las componentes de flujo de los caminos se encuentran en cero o el valor total del requerimiento, por lo que no tiene sentido la modificación del exponente.

El resto de los parámetros se fue variando conforme al análisis de los resultados con el fin de encontrar la mejor combinación. En la tabla siguiente se detallan los parámetros utilizados en algunos de los ensayos.

| Ensayo | Cantidad de Hormigas | Iteraciones | Coeficiente de prioridad de feromona **α** | Coeficiente de evaporación $\rho$ | Valor de Feromona inicial $\tau_0$ | Iteraciones para reiniciar | Efectividad |
|---|---|---|---|---|---|---|---|
| 1-ACO2 | 10 | 100 | 0.1 | 0.01 | 0.1 | 100 | 93% |
| 2-ACO2 | 10 | 1000 | -1 | 0.01 | 0.1 | 50 | 93% |
| 3-ACO2 | 100 | 500 | -1 | 0.01 | 0.1 | 50 | 89% |
| 4-ACO2 | 20 | 100 | -0.1 | 0.01 | 0.1 | 100 | 92% |
| 5-ACO2 | 20 | 100 | -0.5 | 0.01 | 0.1 | 100 | 93% |
| 6-ACO2 | 20 | 100 | -1 | 0.01 | 0.1 | 100 | 90% |
| 7-ACO2 | 20 | 100 | 0 | 0.01 | 0.1 | 100 | 93% |
| 8-ACO2 | 20 | 100 | 0.1 | 0.01 | 0.1 | 100 | 93% |
| 9-ACO2 | 20 | 100 | 0.5 | 0.01 | 0.1 | 100 | 88% |
| 10-ACO2 | 20 | 100 | 1 | 0.01 | 0.1 | 100 | 84% |
| 11-ACO2 | 20 | 100 | 2 | 0.01 | 0.1 | 100 | 56% |
| 12-ACO2 | 30 | 100 | 0.1 | 0.01 | 0.1 | 100 | 89% |
| 13-ACO2 | 5 | 100 | 0.1 | 0.01 | 0.1 | 100 | 94% |
| 14-ACO2 | 5 | 1500 | -1 | 0.01 | 0.1 | 50 | 89% |
| 15-ACO2 | 5 | 200 | 0.1 | 0.01 | 0.1 | 100 | 94% |
| 16-ACO2 | 5 | 300 | 0.1 | 0.01 | 0.1 | 10 | 92% |
| 17-ACO2 | 5 | 300 | -1 | 0.01 | 0.1 | 10 | 94% |
| 18-ACO2 | 5 | 300 | -1 | 0.01 | 0.1 | 30 | 95% |
| 19-ACO2 | 5 | 300 | -1 | 0.01 | 0.1 | 50 | 89% |
| 20-ACO2 | 5 | 300 | 0.1 | 0.01 | 0.1 | 100 | 91% |
| 21-ACO2 | 5 | 300 | 0.1 | 0.01 | 0.1 | 50 | 92% |
| 22-ACO2 | 5 | 300 | 1 | 0.01 | 0.1 | 50 | 95% |
| 23-ACO2 | 5 | 300 | 1.5 | 0.01 | 0.1 | 30 | 84% |
| 24-ACO2 | 5 | 500 | -0.1 | 0.01 | 0.1 | 30 | 88% |
| 25-ACO2 | 5 | 500 | -1 | 0.005 | 0.1 | 30 | 92% |
| 26-ACO2 | 5 | 500 | -1 | 0.01 | 0.1 | 30 | 87% |
| 27-ACO2 | 5 | 500 | -1 | 0.05 | 0.1 | 30 | 89% |
| 28-ACO2 | 50 | 500 | -1 | 0.01 | 0.1 | 50 | 90% |

*Tabla 5.7: Parámetros de configuración de ensayos ACO 2 para el Caso 1*

Como puede observarse, los parámetros de efectividad son similares en la mayoría de los casos, y una efectividad menor a 100% se debe fundamentalmente a dos óptimos locales que el algoritmo encuentra dificultad para sortearlos en este tamaño del problema. Uno tiene el valor 56 y otro 75.

La figura siguiente muestra los resultados de los ensayos realizados con los parámetros indicados anteriormente, donde se observan los valores atípicos mencionados.

Salvo el ensayo Nº 11 donde claramente un valor grande de α genera estancamiento en óptimos locales y por tanto malas soluciones, el resto de los ensayos presenta resultados estadísticamente similares con pocas diferencias.

Variaciones de parámetros importantes como número de iteraciones y cantidad de hormigas no generan cambios significativos en los resultados.

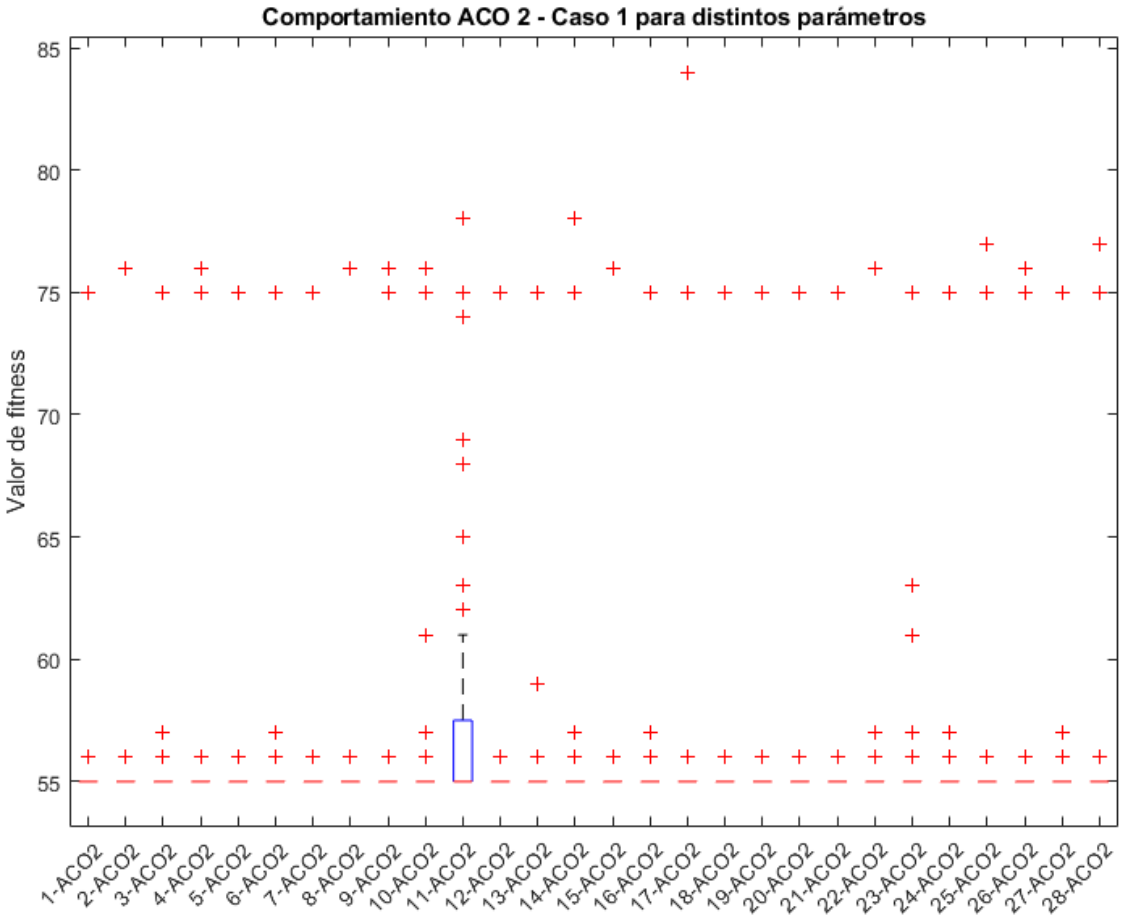


*Figura 5.12 – Ensayos ACO 2 Caso 1 para distintos parámetros.*

Las iteraciones necesarias para alcanzar el óptimo global se muestran en la figura 5.13. Los resultados demuestran que el 100% de las ejecuciones que alcanzó el óptimo global lo hizo con menos de 50 iteraciones

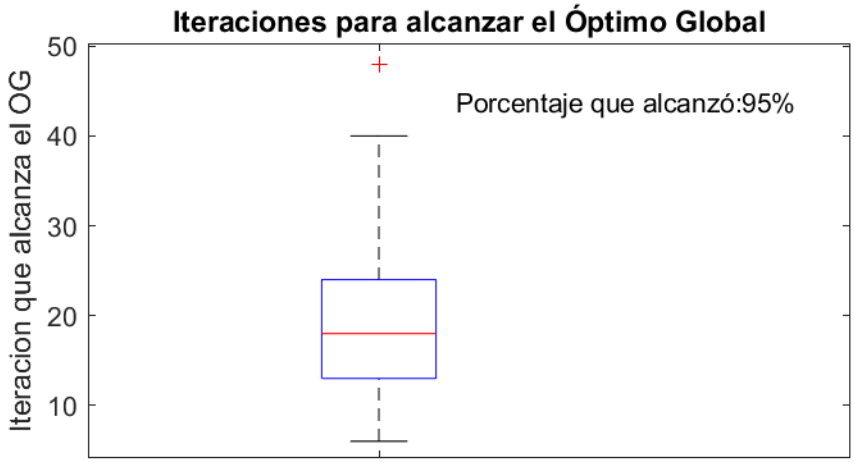


*Figura 5.13: Iteraciones para alcanzar el óptimo con algoritmo ACO 2 para el Caso 1*

La mejor combinación de parámetros hallada se detalla en la tabla 5.8 y corresponde al ensayo 18 con un 95% de efectividad en la búsqueda del óptimo global. La tabla 5.9 muestra los resultados estadísticos del ensayo.

| Cantidad de Hormigas | Iteraciones | Coeficiente de prioridad de feromona **α** | Coeficiente de prioridad del valor heurístico ***β*** | Coeficiente de evaporación ***ρ*** | Valor de Feromona inicial $\tau_0$ | Iteraciones para reiniciar |
|---|---|---|---|---|---|---|
| 5 | 50 | -1 | 1 | 0.01 | 0.1 | 30 |

*Tabla 5.8: Parámetros óptimos de configuración del algoritmo ACO 2 para el Caso 1*

| **Estadístico** | **Valor** |
|---|---|
| Mínimo | 55 |
| Efectividad | 96% |
| Media | 55.24 |
| Desviación Estándar | 2 |
| Varianza | 4.022 |
| Coeficiente de Variación | 0.0036 |
| **Cuartiles** | |
| q1 | 55 |
| q2 | 55 |
| q3 | 55 |
| **Tabla de Frecuencias** | |
| Value Count Percent | |
| 55 95 95.00% | |
| 56 4 4.00% | |
| 75 1 1.00% | |

*Tabla 5.9: Datos estadísticos del algoritmo ACO 2 para el Caso 1*

La distribución de soluciones se representa en la siguiente figura. De las cinco soluciones que no alcanzaron el óptimo global cuatro están a una unidad de diferencia y una en el óptimo local detectado de valor 75.

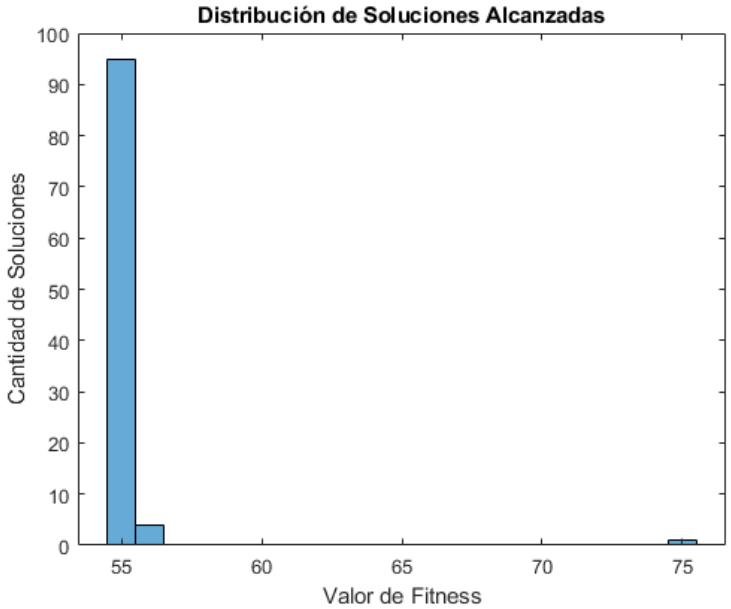


*Figura 5.14: Distribución de soluciones del algoritmo ACO 2 para el Caso 1*

La evolución del algoritmo de una ejecución típica donde se alcanza el óptimo global se muestra a continuación. Para esta estrategia se observaron mejoras más continuas que en el caso

anterior, en general sin estancamiento salvo en los casos que no se alcanza el óptimo global. En estos casos el desarrollo del algoritmo es similar hasta alcanzar el óptimo local (56 o 75 principalmente) para luego estancarse en ese valor hasta finalizar las iteraciones.

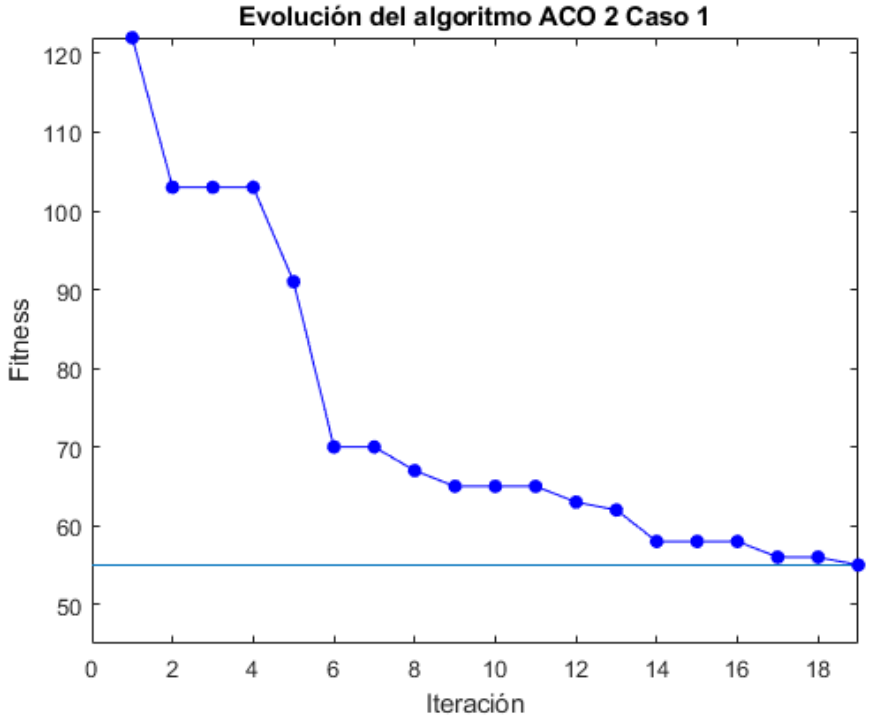


*Figura 5.15: Evolución típica del fitness para un ensayo del algoritmo ACO 2 para el Caso 1*

## 5.2.4 Ensayos del algoritmo de murciélagos (BA) (Caso 1)

Al igual que el algoritmo PSO con pocos ensayos se pudo determinar los parámetros adecuados para llegar a una efectividad del 100 %. La tabla 5.10 muestra los ensayos realizados los cuales se ejecutaron con 50 movimientos de los murciélagos. Y la figura 5.16 expone los diagramas de caja y bigote para comparar los resultados de los ensayos.

| Ensayo | Cantidad de Murciélagos | Frecuencia Mínima $f_{min}$ | Frecuencia máxima $f_{max}$ | Factor de Inercia $w$ | Volumen inicial $A_0$ | Constante de Disminución de volumen α | Constante aumento del pulso γ | Eficiencia |
|---|---|---|---|---|---|---|---|---|
| 1-BA | 10 | 0 | 1 | 0 | 1 | 0.9 | 0.5 | 0% |
| 2-BA | 10 | 0 | 3 | 0 | 1 | 0.9 | 0.5 | 39% |
| 3-BA | 10 | 0 | 5 | 0 | 1 | 0.9 | 0.5 | 84% |
| 4-BA | 10 | 0 | 6 | 0 | 1 | 0.9 | 0.5 | 86% |
| 5-BA | 10 | 0 | 6 | 0 | 1.2 | 0.9 | 0.5 | 94% |
| 6-BA | 10 | 0 | 6 | 0 | 1.4 | 0.9 | 0.5 | 94% |
| 7-BA | 100 | 0 | 5 | 0 | 1.5 | 0.9 | 0.5 | 100% |
| 8-BA | 20 | 0 | 5 | 0 | 1.2 | 0.9 | 0.5 | 95% |
| 9-BA | 20 | 0 | 6 | 0 | 1.2 | 0.9 | 0.5 | 98% |
| 10-BA | 30 | 0 | 5 | 0 | 1 | 0.9 | 0.5 | 99% |
| 11-BA | 30 | 0 | 5 | 0 | 1 | 0.99 | 0.5 | 100% |
| 12-BA | 30 | 0 | 5 | 0 | 1.2 | 0.9 | 0.5 | 99% |
| 13-BA | 30 | 0 | 6 | 0 | 1.2 | 0.9 | 0.5 | 99% |
| 14-BA | 40 | 0 | 6 | 0 | 1.2 | 0.9 | 0.5 | 99% |
| 15-BA | 40 | 0 | 6 | 0 | 1.2 | 0.99 | 0.5 | 100% |

*Tabla 5.10: Parámetros de configuración de ensayos BA para el Caso 1*

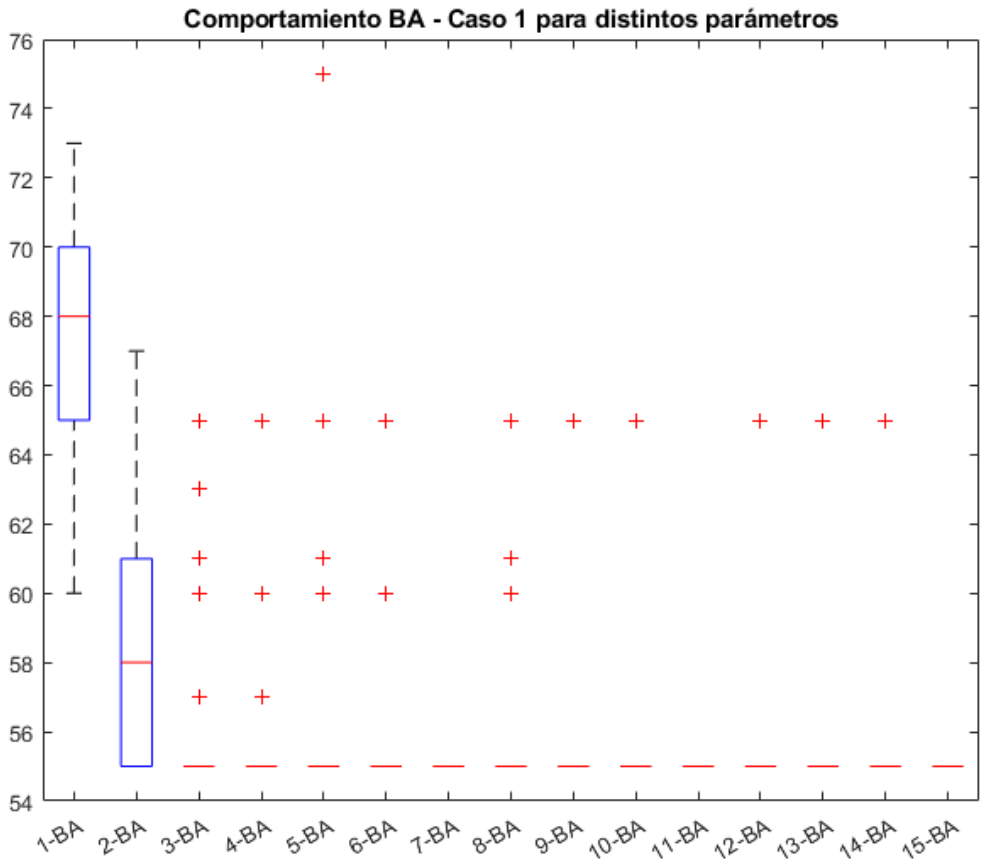


*Figura 5.16 – Ensayos BA Caso 1 para distintos parámetros.*

El algoritmo alcanza valores estadísticos aceptables a partir de una $f_{max}$=5 luego se logran mayores niveles de efectividad a medida que se incrementa el volumen inicial $A_0$. Para los valores ensayados, las variaciones de la constante de disminución de volumen α y la constante de aumento del pulso (γ) no producen mejoras en los resultados obtenidos.

Los resultados mejoran a medida que el valor de la constante de aceleración *w* se hace más pequeño siendo cero el valor óptimo y, por lo tanto, el valor utilizado en los ensayos mostrados anteriormente. Esto respeta la lógica de que cuando el óptimo es alcanzado, la velocidad tome valor cero para que el murciélago virtual no se mueva de dicha posición, en acuerdo con la ecuación 4.16.

La tabla 5.11 muestra los parámetros óptimos de configuración correspondiente al ensayo 15, con el cual el algoritmo inspirado en murciélagos alcanza el óptimo global en el 100% de todas las ejecuciones. La diferencia con las otras combinaciones de parámetros que alcanzan el 100% de efectividad es mínima y radica en el tiempo utilizado para encontrar el óptimo.

La cantidad de iteraciones necesarias para alcanzar el óptimo global con estos parámetros se observan en la figura siguiente, con lo cual se concluye que con 20 iteraciones es suficiente para el alcance del óptimo global en el problema planteado.

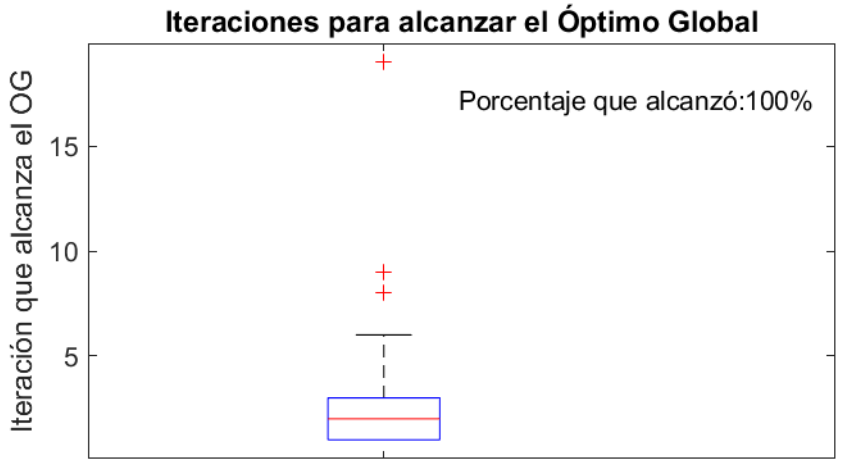


*Figura 5.17: Iteraciones para alcanzar el óptimo con algoritmo BA para el Caso 1*

| Cantidad de Murciélagos | Iteraciones | Frecuencia Mínima $f_{min}$ | Frecuencia máxima $f_{max}$ | Factor de Inercia $w$ | Volumen inicial $A_0$ | Constante de Disminución de volumen $\alpha$ | Constante aumento del pulso $\gamma$ |
|---|---|---|---|---|---|---|---|
| 40 | 20 | 0 | 6 | 0 | 1.2 | 0.99 | 0.5 |

*Tabla 5.11: Parámetros óptimos de configuración del algoritmo BA para el Caso 1*

Con dichos parámetros los resultados estadísticos para 100 ensayos son:

| Estadístico | Valor |
|---|---|
| Mínimo | 55 |
| Efectividad | 100% |
| Media | 55 |
| Desviación Estándar | 0 |
| Varianza | 0 |
| Coeficiente de Variación | 0 |
| **Cuartiles** | |
| q1 | 55 |
| q2 | 55 |
| q3 | 55 |
| **Tabla de Frecuencias** | |
| Value Count Percent | |
| 55 100 100.00% | |

*Tabla 5.12: Datos estadísticos del algoritmo BA para el Caso 1*

El avance típico del algoritmo posee pendientes pronunciadas y en raras ocasiones se producen estancamientos como era común en los algoritmos inspirados en colonias de hormigas. Se muestra a continuación una ejecución típica donde el alcance del óptimo global se logra con solo 5 iteraciones, lo cual se realiza en un tiempo muy reducido.

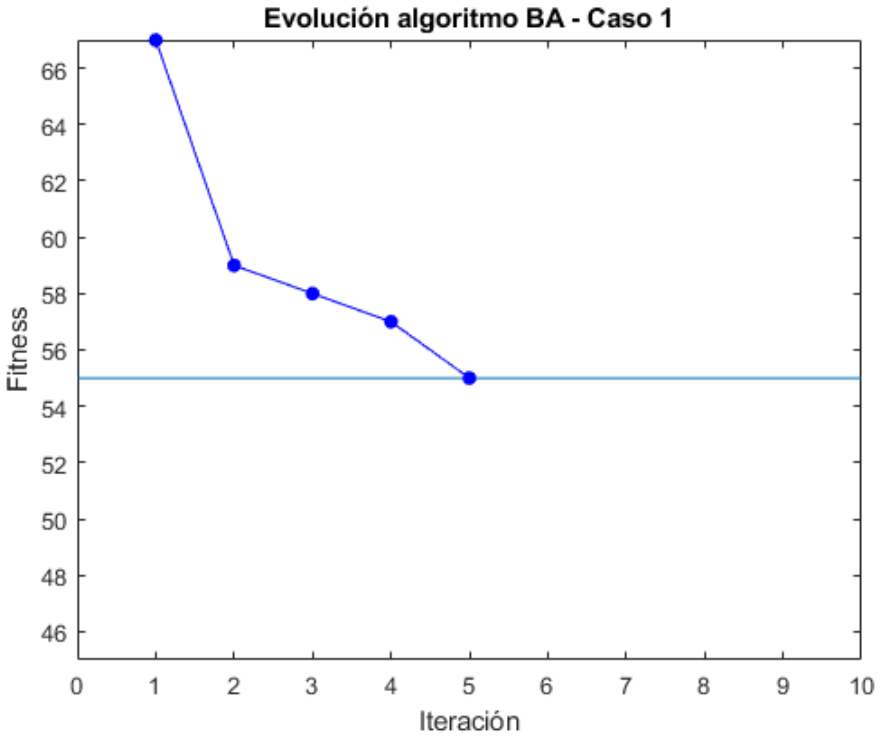


*Figura 5.18: Evolución típica del fitness para un ensayo del algoritmo BA para el Caso 1*

Debido al buen funcionamiento del algoritmo de murciélagos, no tiene sentido la utilización de un post-optimizador por lo que se omiten las pruebas con el algoritmo BPA reservándose para problemas de mayor tamaño.

### 5.2.5 Comparación del desempeño de los algoritmos para el Caso 1

Para el tamaño de la red del caso 1, los desempeños de todos los algoritmos son estadísticamente similares en cuanto a la obtención de resultados, y su comparación es trivial. La mayor diferencia radica en el tiempo empleado para encontrar una solución.

Con el fin de tener parámetros de comparación se detallan en la siguiente tabla la efectividad del algoritmo, los tiempos estadísticos medios y medianos para encontrar el óptimo global, donde intervienen solo los ensayos que llegaron a este valor, y el tiempo que toma al algoritmo hacer una ejecución para un valor desconocido de óptimo global. En todos los casos se toman los parámetros de configuración óptimos definidos para este problema en cada estrategia.

| Algoritmo | PSO | ACO 1 | ACO 2 | BA | BPA |
|---|---|---|---|---|---|
| **Porcentaje de efectividad** | 100% | 95% | 96% | 100% | - |
| **Tiempo medio para obtener óptimo** | 0.34 | 275.33 | 0.24 | 0.085 | - |
| **Tiempo mediano para obtener el óptimo** | 0.251 | 209.78 | 0.13 | 0.067 | - |
| **Tiempo medio para obtener una solución** | 2.25 | 583.63 | 0.96 | 0.884 | - |

| | | | | | |
|---|---|---|---|---|---|
| **Tiempo mínimo para obtener el óptimo** | 0.0138 | 95.68 | 0.036 | 0.0017 | - |
| **Tiempo máximo para obtener el óptimo** | 1.7 | 1201.4 | 2.5 | 0.7 | - |

*Tabla 5.13: Comparativa del tiempo empleado por los algoritmos Caso 1.*

La figura 5.19 muestra la comparación de los tiempos necesarios para alcanzar el óptimo global para 100 ensayos del algoritmo.

El algoritmo BA es el que arriba con mayor celeridad a las soluciones y con muy baja varianza, le sigue el algoritmo ACO 2 (redistribución de flujos), PSO quien presenta una dispersión mayor, y finalmente los tiempos para el algoritmo ACO 1 (Distribución de Flujos) son exageradamente mayores que el resto de las estrategias.

La estrategia de distribución de flujos ACO compensa su lentitud con un balance de cargas que podría ser deseable, y que fue pensado como tal, en el diseño del algoritmo a través de la definición del valor heurístico η (Ecuación 4.7).

Si bien existen distintas soluciones que tienen el mismo valor de aptitud coincidente con el óptimo global (55), la mayoría de las soluciones encontradas por ACO 1, tienen la forma [10 0 0 5 0 5 3 2 15] mientras que las encontrada por BA responde a [10 0 0 0 0 10 0 5 15], donde los flujos se encuentran menos repartidos.

Si bien las distintas soluciones tienen el mismo valor de fitness y cumplen las restricciones tanto de demanda como de capacidad de los enlaces, la característica de distribuir más equitativamente los flujos, siempre minimizando el costo, podría ser una capacidad deseable que compense la diferencia de tiempos para encontrar una solución.

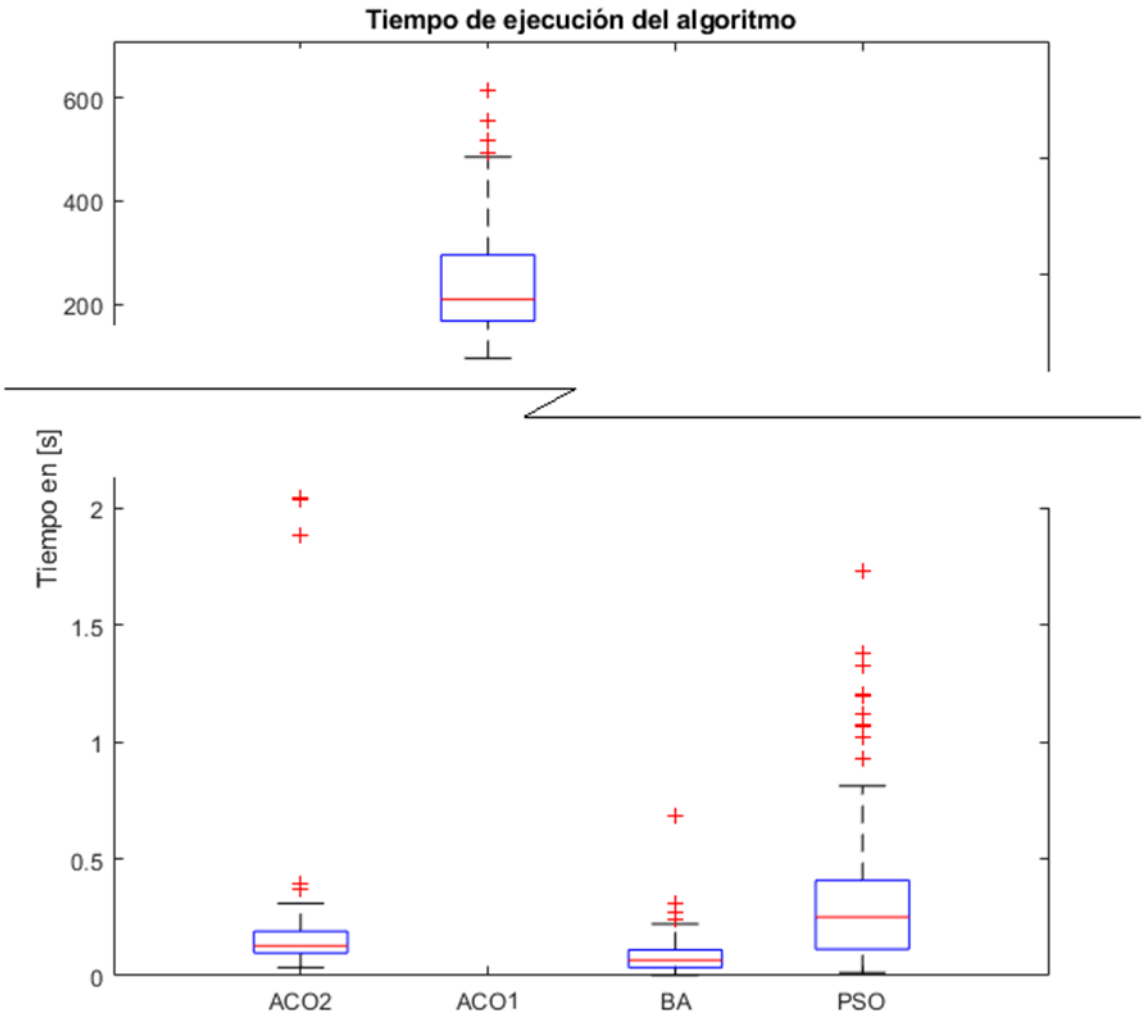


*Figura 5.19: Comparativa de tiempo de ejecución de los algoritmos para encontrar el óptimo global Caso 1*

# 5.3 Experimentación con CASO 2 – Red de 6 nodos

## 5.3.1 Ensayos del algoritmo PSO (Caso 2)

Para este caso la experimentación no resultó trivial como en el caso de la red de 4 nodos, en cuanto a la selección correcta de los parámetros del algoritmo. Se requirieron varios ensayos para determinar los parámetros óptimos. Se exponen en la tabla 5.14 solo los parámetros con los que se realizaron los ensayos, omitiendo algunos ensayos intermedios cuyos resultados fueron similares. La figura 5.20 muestra comparativamente los resultados para estos parámetros.

Se recuerda que el orden en que figuran los ensayos no se condice con el orden en que fueron ejecutados, sino que se corresponde con la lectura de los archivos por parte del software.

| Ensayo | Cantidad de Partículas | Iteraciones | Factor de Aprendizaje individual *C1* | Factor de Aprendizaje Social *C2* | Factor de Inercia *w* | Efectividad |
|---|---|---|---|---|---|---|
| 1-PSO | 100 | 150 | 16 | 1 | 1 | 48% |
| 2-PSO | 100 | 150 | 7 | 0.5 | 1 | 37% |
| 3-PSO | 100 | 150 | 7.8 | 0.5 | 1 | 46% |
| 4-PSO | 100 | 150 | 8 | 0.5 | 0.7 | 20% |
| 5-PSO | 100 | 150 | 8 | 0.5 | 1.1 | 12% |

| | | | | | | |
|---|---|---|---|---|---|---|
| 6-PSO | 100 | 150 | 8 | 0.5 | 1 | 51% |
| 7-PSO | 100 | 150 | 8.5 | 0.5 | 1 | 41% |
| 8-PSO | 100 | 150 | 9 | 0.7 | 1 | 41% |
| 9-PSO | 100 | 50 | 8 | 0.3 | 1 | 10% |
| 10-PSO | 150 | 150 | 8 | 0.5 | 1 | 51% |
| 11-PSO | 200 | 200 | 8 | 0.5 | 1 | 71% |
| 12-PSO | 300 | 200 | 8 | 0.5 | 1 | 84% |
| 13-PSO | 350 | 250 | 8 | 0.5 | 1 | 93% |
| 14-PSO | 400 | 250 | 8 | 0.5 | 1 | 94% |
| 15-PSO | 50 | 150 | 5 | 0.5 | 1 | 21% |
| 16-PSO | 50 | 150 | 5 | 1 | 1 | 20% |
| 17-PSO | 50 | 150 | 7 | 0.5 | 1 | 26% |
| 18-PSO | 50 | 150 | 9 | 0.1 | 1 | 14% |
| 19-PSO | 50 | 150 | 9 | 0.5 | 1 | 20% |
| 20-PSO | 50 | 150 | 9 | 0.7 | 1 | 35% |

*Tabla 5.14: Parámetros de configuración de ensayos PSO para el Caso 2*

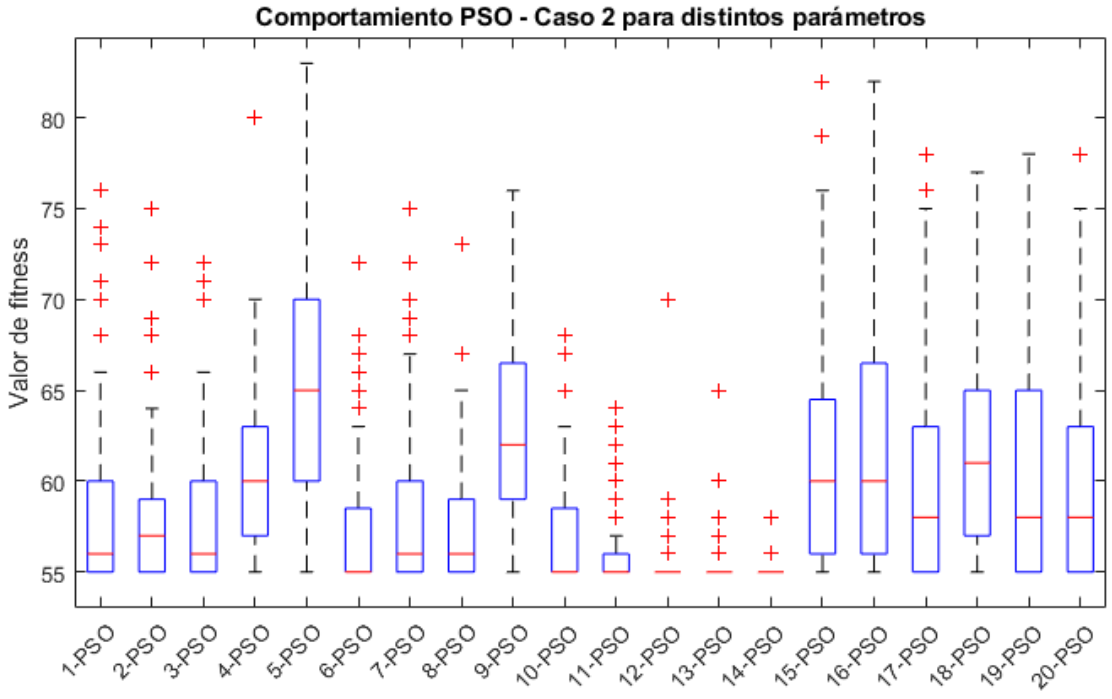


*Figura 5.20 – Ensayos PSO Caso 2 para distintos parámetros.*

El comportamiento vuelve a ser similar que para el caso 1. Los mejores resultados se obtienen con factores de aprendizaje individual mayores que el factor de aprendizaje social, solo que la relación $c_1/c_2$ debe ser mayor que en el problema de la red de 4 nodos para llegar a resultados aceptables.

Una relación $c_1/c_2$ de 16 a 1 con valores de $c_2$ menores a la unidad aparenta ser la combinación buscada. Al igual que en el problema del caso 1, el parámetro de inercia no presenta mejoras al utilizar valores distintos de la unidad.

Comparando con el problema del caso 1, la cantidad de partículas e iteraciones necesarias crece considerablemente, lo cual era esperable.

Las iteraciones necesarias para encontrar el óptimo global son las siguientes

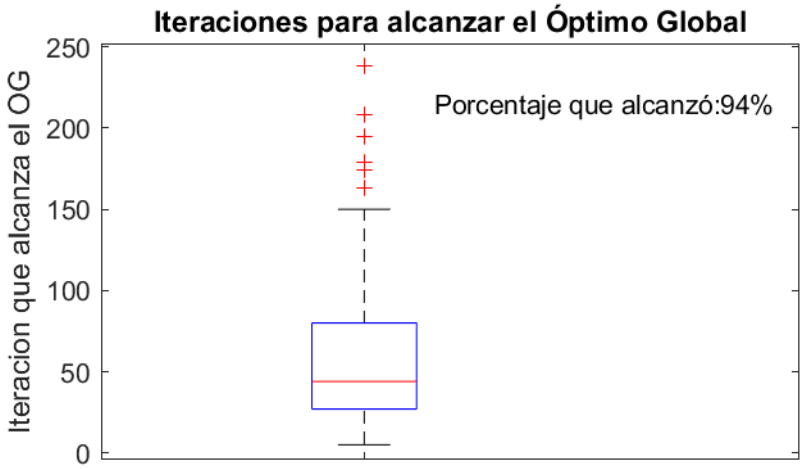


*Figura 5.21: Iteraciones para alcanzar el óptimo con algoritmo PSO para el Caso 2*

Por lo que se entiende que con 250 iteraciones debería alcanzar para encontrar el óptimo.

Los parámetros óptimos para esta instancia del problema se representan en la tabla 5.15. y los resultados estadísticos para dichos parámetros en la tabla 5.16

| Cantidad de Partículas | Iteraciones | Factor de Aprendizaje individual *C1* | Factor de Aprendizaje Social *C2* | Factor de Inercia *w* |
|---|---|---|---|---|
| 400 | 250 | 8 | 0.5 | 1 |

*Tabla 5.15: Parámetros óptimos de configuración del algoritmo PSO para el Caso 2*

| **Estadístico** | **Valor** |
|---|---|
| Mínimo | 55 |
| Efectividad | 94% |
| Media | 55.08 |
| Desviación Estándar | 0.3674 |
| Varianza | 0.1349 |
| Coeficiente de Variación | 0.0067 |
| **Cuartiles** | |
| q1 | 55 |
| q2 | 55 |
| q3 | 55 |
| **Tabla de Frecuencias** | |
| Value Count Percent<br>55 94 94.00%<br>56 5 5.00%<br>58 1 1.00% | |

*Tabla 5.16: Datos estadísticos del algoritmo PSO para el Caso 2*

Si bien hay un 6% de los casos en los que no se llega al óptimo global, la diferencia de valor de aptitud de dichas soluciones es mínima, como puede observarse en la tabla de frecuencias y en la gráfica asociada.

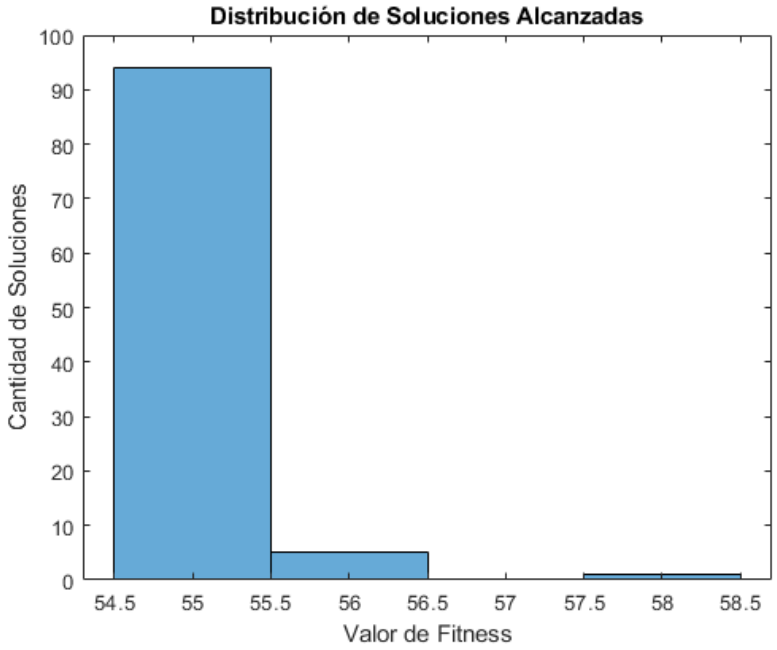


*Figura 5.22: Distribución de soluciones del algoritmo PSO para el Caso 2*

La evolución del algoritmo para un ensayo típico es la siguiente:

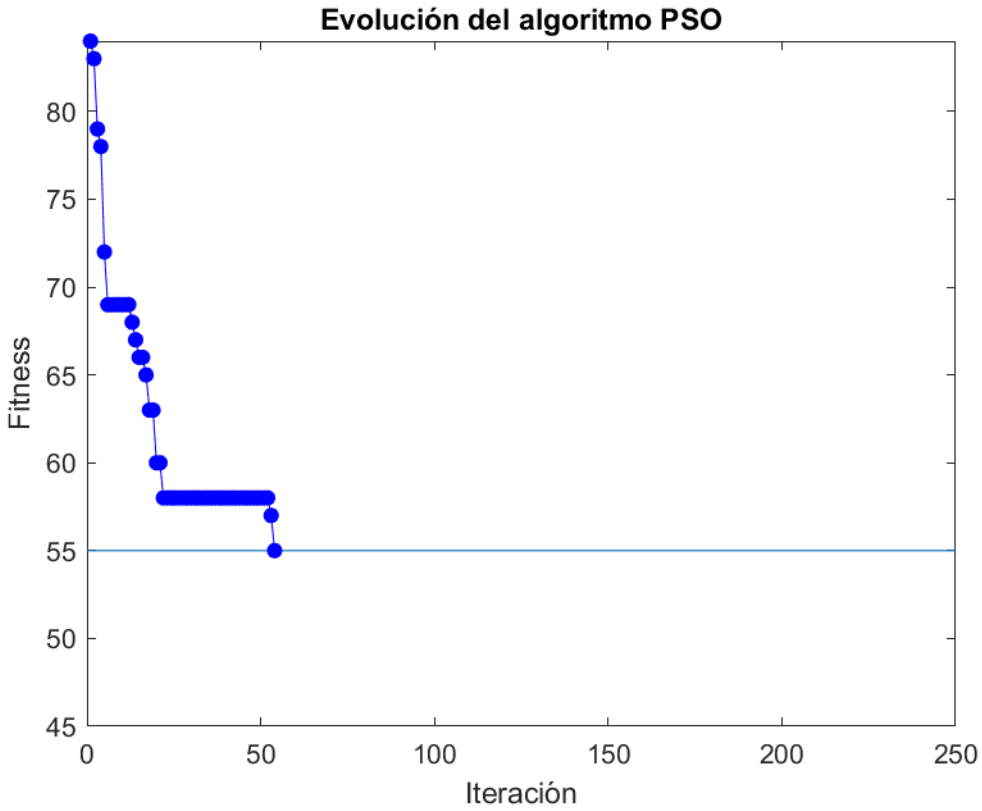


*Figura 5.23: Evolución del fitness para un ensayo típico del algoritmo PSO para el Caso 2*

### 5.3.2 Ensayos del algoritmo ACO 1 – Distribución de flujos (Caso 2)

Para la estrategia de distribución de flujos inspirado en colonia de hormigas que distribuye los flujos en los caminos, se determinó que pequeñas variaciones en los parámetros podía generar variaciones significativas en los resultados. Si bien la dispersión de los resultados para una corrida de 100 ensayos era muy pequeña, el algoritmo no alcanzaba el óptimo global del problema (Fig. 5.24). La tabla 5.17 expone los parámetros para 4 ensayos donde se modifica solamente el parámetro de influencia de la feromona y la cantidad de iteraciones.

| Ensayo | Cantidad de Hormigas | iteraciones | Parámetro Influencia feromona **α** | Parámetro Influencia valor heurístico **β** | Coeficiente de evaporación $\rho$ | Valor de Feromona inicial $\tau_0$ | Efectividad |
|---|---|---|---|---|---|---|---|
| 1-ACO1 | 20 | 300 | 2 | 1 | 0.01 | 0.1 | 0% |
| 2-ACO1 | 20 | 500 | 1 | 1 | 0.01 | 0.1 | 0% |
| 3-ACO1 | 20 | 500 | 2 | 1 | 0.01 | 0.1 | 5% |
| 4-ACO1 | 20 | 500 | 3 | 1 | 0.01 | 0.1 | 100% |

*Tabla 5.17: Parámetros de configuración de ensayos ACO 1 para el Caso 1*

En el gráfico comparativo (Figura 5.24) puede analizarse el comportamiento de los resultados al variar el parámetro de influencia de feromona manteniendo constante el resto de los parámetros para los ensayos 2-ACO1 a 4-ACO1. Análogamente el efecto de la cantidad de iteraciones se observa en los ensayos 1-ACO1 y 3-ACO1 los cuales tienen los mismos parámetros, pero con diferencia en la cantidad de iteraciones.

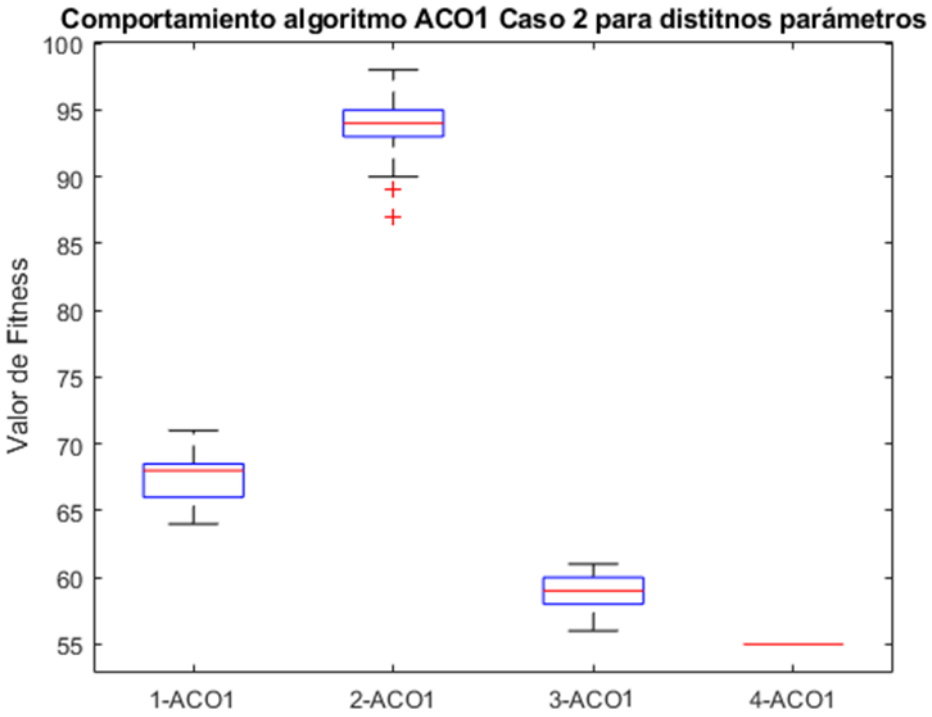


*Figura 5.24 – Ensayos ACO 1 Caso 2 para distintos parámetros.*

Mediante los ensayos se comprueba nuevamente, que el parámetro de influencia de feromona α, tiene que ser mayor al parámetro de influencia del valor heurístico β para lograr buenos resultados y que las variaciones del valor de $\tau_0$ no reflejan diferencias apreciables en los resultados.

Al contrario que en el caso 1, para este problema existen diferencias notables entre un ensayo y otro. Mas allá que la cantidad de iteraciones mejora los resultados, el factor determinante es α. De hecho 180 iteraciones son suficientes para los parámetros del ensayo 4 lo que puede determinarse fácilmente observando la siguiente figura.

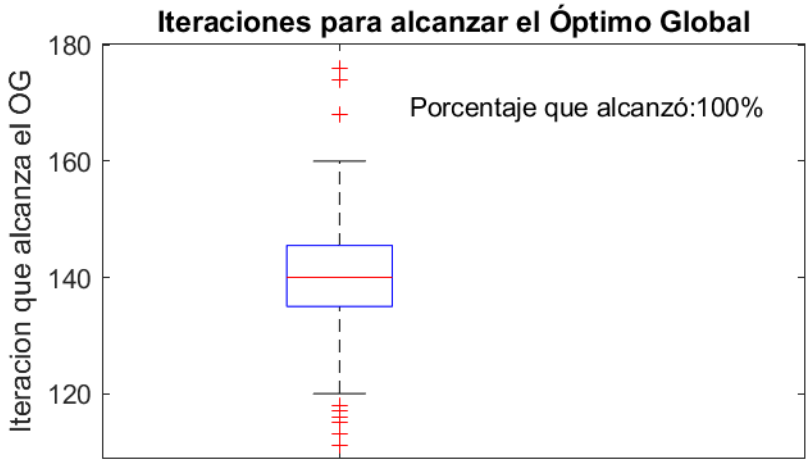


*Figura 5.25: Iteraciones para alcanzar el óptimo con algoritmo ACO 1 para el Caso 2*

En la tabla 5.18 se observan los parámetros óptimos y en la tabla 5.19 los valores estadísticos correspondiente a una ejecución de 100 ensayos con dichos valores.

| Cantidad de Hormigas | iteraciones | Parámetro Influencia feromona **α** | Parámetro Influencia valor heurístico **β** | Coeficiente de evaporación **ρ** | Valor de Feromona inicial $\tau_0$ |
|---|---|---|---|---|---|
| 20 | 180 | 3 | 1 | 0.01 | 0.1 |

*Tabla 5.18: Parámetros óptimos de configuración del algoritmo ACO 1 para el Caso 2*

| **Estadístico** | **Valor** |
|---|---|
| Mínimo | 55 |
| Efectividad | 100% |
| Media | 55 |
| Desviación Estándar | 0 |
| Varianza | 0 |
| Coeficiente de Variación | 0 |
| **Cuartiles** | |
| q1 | 55 |
| q2 | 55 |
| q3 | 55 |
| **Tabla de Frecuencias** | |
| Value Count Percent | |
| 55 100 100.00% | |

*Tabla 5.19: Datos estadísticos del algoritmo ACO 1 para el Caso 2*

Durante el desarrollo del algoritmo se producen estancamientos generando una evolución escalonada al igual que lo observado en el caso 1. La figura 5.26 muestra una evolución típica, en la cual se requirieron 130 iteraciones para alcanzar el óptimo global. Este ensayo corresponde al extremo inferior del diagrama de caja y bigote de la figura 5.25. No obstante, en otras ejecuciones, se han observado escalones con mayor cantidad de iteraciones, y con medianas del orden de 140 como se desprende de la figura 5.25

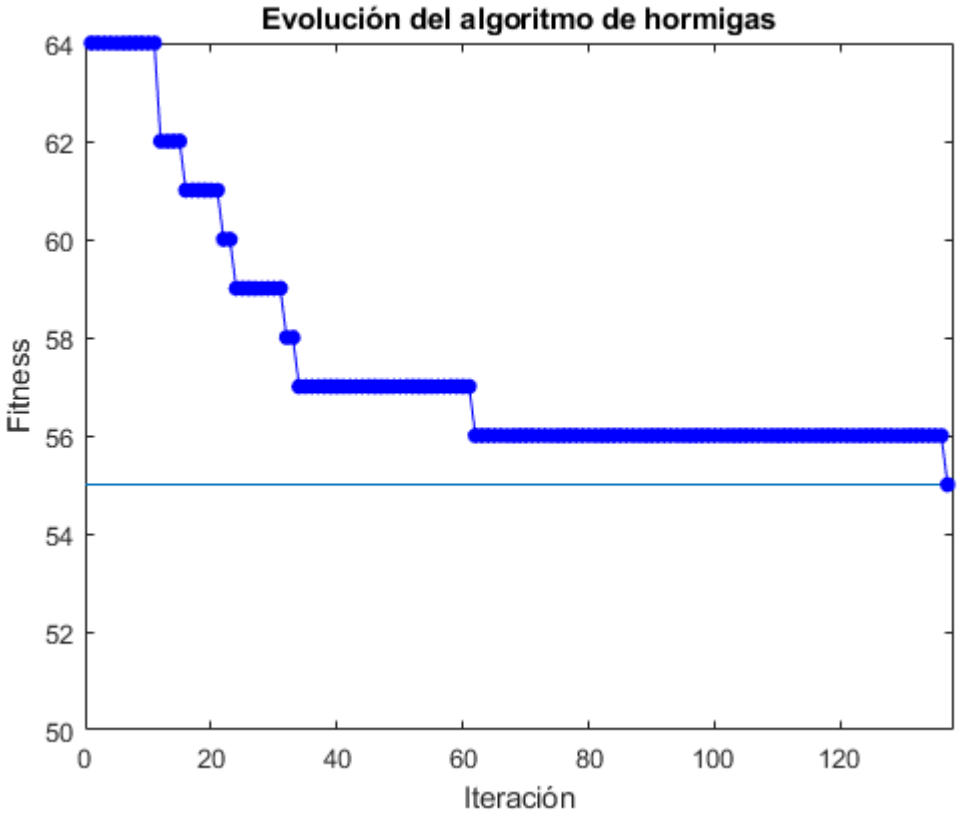


*Figura 5.26: Evolución del fitness para un ensayo del algoritmo ACO 1 para el Caso 2*

### 5.3.3 Ensayos del algoritmo ACO 2 – Redistribución de flujos (Caso 2)

Se realizaron varios ensayos con distintos parámetros para encontrar el mejor desempeño del algoritmo y, al igual que en la estrategia anterior, se registraron algunas diferencias importantes al modificar los valores del coeficiente de prioridad de feromona.

Como se explicó anteriormente, el factor β se establece en 1 para todos los ensayos y se detectó una fuerte dependencia del parámetro α como se muestra en la tabla 5.20.

| Ensayo | Cantidad de Hormigas | Iteraciones | Coeficiente de prioridad de feromona **α** | Coeficiente de evaporación $\boldsymbol{\rho}$ | Valor de Feromona inicial $\tau_0$ | Iteraciones para reiniciar | Efectividad |
|---|---|---|---|---|---|---|---|
| 1-ACO2 | 10 | 50 | -1 | 0.01 | 0.1 | 30 | 61 |
| 2-ACO2 | 20 | 100 | -1 | 0.01 | 0.1 | 100 | 95 |
| 3-ACO2 | 20 | 100 | 0 | 0.01 | 0.1 | 100 | 41 |
| 4-ACO2 | 20 | 100 | 0.001 | 0.01 | 0.1 | 100 | 44 |
| 5-ACO2 | 20 | 100 | 0.01 | 0.01 | 0.1 | 100 | 30 |
| 6-ACO2 | 20 | 100 | 0.08 | 0.01 | 0.1 | 100 | 24 |
| 7-ACO2 | 20 | 100 | 0.1 | 0.01 | 0.1 | 100 | 19 |
| 8-ACO2 | 20 | 100 | 0.2 | 0.01 | 0.1 | 100 | 12 |
| 9-ACO2 | 20 | 100 | 1 | 0.01 | 0.1 | 100 | 0 |
| 10-ACO2 | 30 | 50 | -1 | 0.01 | 0.1 | 30 | 95 |
| 11-ACO2 | 30 | 50 | -1.5 | 0.01 | 0.1 | 30 | 96 |

*Tabla 5.20: Parámetros de configuración de ensayos ACO 2 para el Caso 2*

Al contrario que en el caso 1, se encuentra mayor diferencia entre los ensayos, y, para los mejores parámetros encontrados (Ensayo 11-ACO2), vuelven a aparecer los mismos óptimos locales que para el caso 1 con valores 56 y 75.

Gráficamente puede observarse una particularidad de los ensayos en los diagramas de caja y bigote que muestra la figura 5.27.

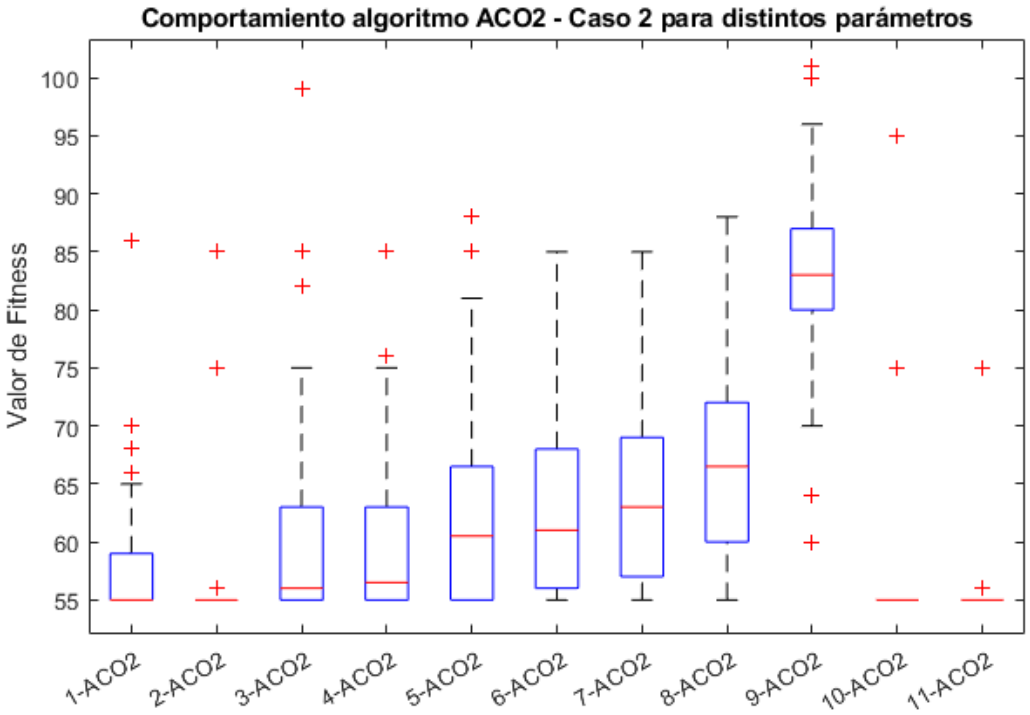


*Figura 5.27 – Ensayos ACO 2 Caso 1 para distintos parámetros.*

A medida que disminuye el valor de α, los resultados mejoran, lo que se observa en los ensayos 2 a 8. Incluso, valores negativos de α fueron los responsables de la generación de los mejores resultados. Esto se atribuye a que la lógica con la que se diseñó el algoritmo puede llevar a estancamientos en óptimos locales, que impiden continuar la evolución. Un valor de α negativo invierte la probabilidad aumentando el campo exploratorio del algoritmo, y evitando la necesidad de reinicializar la feromona. De hecho, si bien la iteración de *reset* fue configurada en 30, puede observarse en la figura 5.28, que las iteraciones necesarias para alcanzar el óptimo global no superan las 27 incluyendo los casos atípicos.

La mejora continua que puede observarse en la figura 5.30, apoya el hecho de que la determinación de un buen juego de parámetros que gobiernan la evolución del algoritmo, puede evitar la necesidad de la utilización de una estrategia extrema de reinicio.

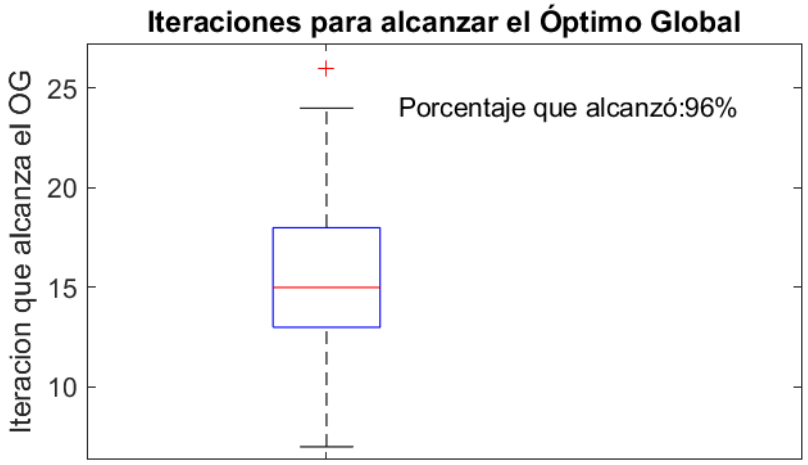


*Figura 5.28: Iteraciones para alcanzar el óptimo con algoritmo ACO 2 para el Caso 2*

La mejor combinación de parámetros hallada se corresponde con el ensayo 11-ACO2, aunque los ensayos 10-ACO2 y 2-ACO2 presentan resultados estadísticos similares los que podrían preferirse ya que al tener un 30% menos de hormigas virtuales, el algoritmo sería más rápido. Los parámetros óptimos y los estadísticos correspondientes se muestran en las tablas 5.21 y 5.22 respectivamente:

| Cantidad de Hormigas | Iteraciones | Coeficiente de prioridad de feromona **α** | Coeficiente de prioridad del valor heurístico ***β*** | Coeficiente de evaporación ***ρ*** | Valor de Feromona inicial $\tau_0$ | Iteraciones para reiniciar |
|---|---|---|---|---|---|---|
| 30 | 30 | -1.5 | 1 | 0.01 | 0.1 | 30 |

*Tabla 5.21: Parámetros óptimos de configuración del algoritmo ACO 2 para el Caso 2*

| **Estadístico** | **Valor** |
|---|---|
| Mínimo | 55 |
| Efectividad | 96% |
| Media | 55.42 |
| Desviación Estándar | 2.8 |
| Varianza | 7.9 |
| Coeficiente de Variación | 0.051 |
| **Cuartiles** | |
| q1 | 55 |
| q2 | 55 |
| q3 | 55 |
| **Tabla de Frecuencias** | |
| Value Count Percent<br>55 95 96.00%<br>56 2 2.00%<br>75 2 2.00% | |

*Tabla 5.22: Datos estadísticos del algoritmo ACO 2 para el Caso 2*

La distribución es muy similar a la del caso 1, donde las soluciones que no llegan al óptimo global quedan atrapadas en los dos óptimos locales que se dan con mayor frecuencia.

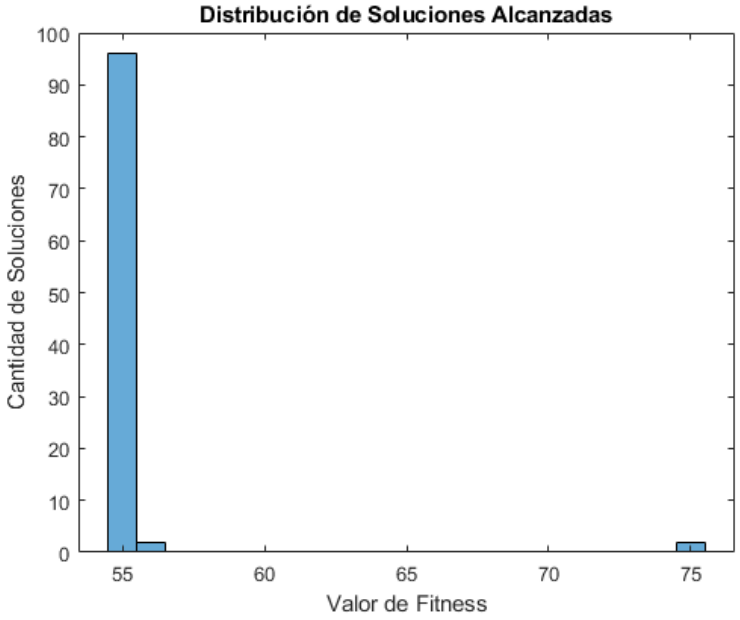


*Figura 5.29: Distribución de soluciones del algoritmo ACO 2 para el Caso 2*

La evolución del algoritmo presenta un comportamiento similar al caso 1. Una evolución relativamente continua con pocos o nulos estancamientos.

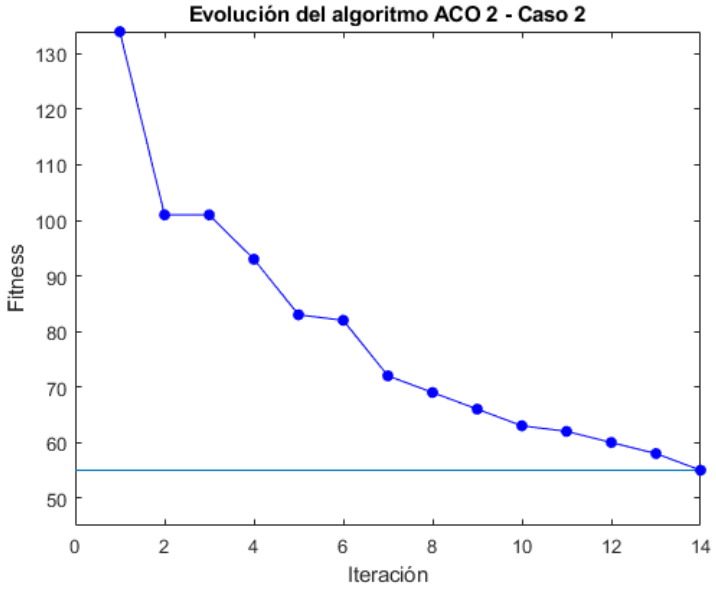


*Figura 5.30: Evolución típica del fitness para un ensayo del algoritmo ACO 2 para el Caso 2*

## 5.3.4 Ensayos del algoritmo de murciélagos (BA) (Caso 2)

Con la experiencia del caso anterior, se determinaron los parámetros adecuados para un desempeño aceptable del algoritmo con pocos ensayos. La tabla 5.23 muestra los parámetros utilizados y la eficiencia alcanzada por el algoritmo bajo análisis con cada configuración de parámetros. La similitud de los valores hace suponer, a proiri, una característica de robustez aceptable para el algoritmo seleccionado. La figura 5.31 muestra la comparación gráfica.

| Ensayo | Cantidad de Murciélagos | Frecuencia Mínima $f_{min}$ | Frecuencia máxima $f_{max}$ | Factor de Inercia $w$ | Volumen inicial $A_0$ | Constante de Disminución de volumen α | Constante aumento del pulso γ | Eficiencia |
|---|---|---|---|---|---|---|---|---|
| 1-BA | 100 | 0 | 7 | 0 | 1.2 | 0.9 | 0.5 | 93% |
| 2-BA | 100 | 0 | 6 | 0 | 1.5 | 0.9 | 0.5 | 96% |
| 3-BA | 100 | 0 | 5 | 0 | 1.5 | 0.9 | 0.5 | 98% |
| 4-BA | 100 | 0 | 6 | 0 | 1.5 | 0.9 | 0.5 | 98% |
| 5-BA | 100 | 0 | 7 | 0 | 1.5 | 0.9 | 0.5 | 97% |
| 6-BA | 100 | 0 | 7 | 0 | 1.5 | 0.99 | 0.5 | 98% |
| 7-BA | 150 | 0 | 7 | 0 | 1.2 | 0.9 | 0.5 | 97% |

*Tabla 5.23: Parámetros de configuración de ensayos BA para el Caso 2*

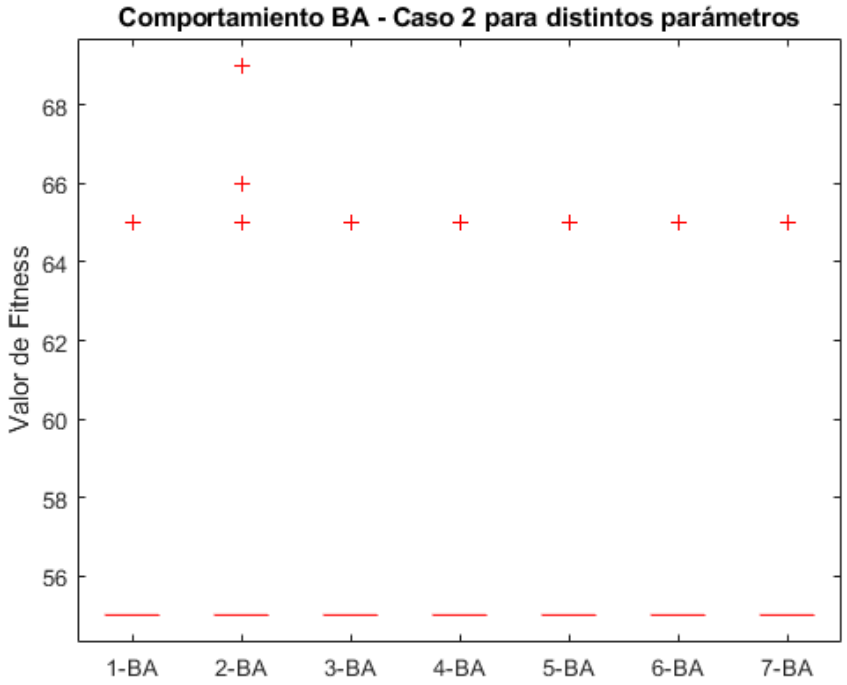


*Figura 5.31 – Ensayos BA Caso 2 para distintos parámetros.*

El algoritmo alcanza idénticos valores estadísticos para los ensayos 3-BA, 4-BA y 6-BA significando no existir diferencias con valores $f_{max}$=5, 6 y 7. Para los valores de $f_{max}$= 7 no existen variaciones importantes al modificar el volumen inicial $A_0$. Para los valores ensayados, las variaciones de la constante de disminución de volumen α y la constante de aumento del pulso γ no producen mejoras en los resultados obtenidos al igual que en el caso anterior. Idéntica situación pasa con la constante de aceleración *w* cuyo valor óptimo es cero.

En todos los ensayos, el algoritmo fue víctima de un óptimo local, difícil de sortear, con un valor de 65.

La cantidad de iteraciones necesarias para alcanzar el óptimo global con los parámetros determinados en la tabla 5.24, se muestran en la figura 5.32. Puede observarse que 50 iteraciones son suficientes para el alcance del óptimo global en el problema planteado, es decir, unas 30 iteraciones más que para el problema del caso 1.

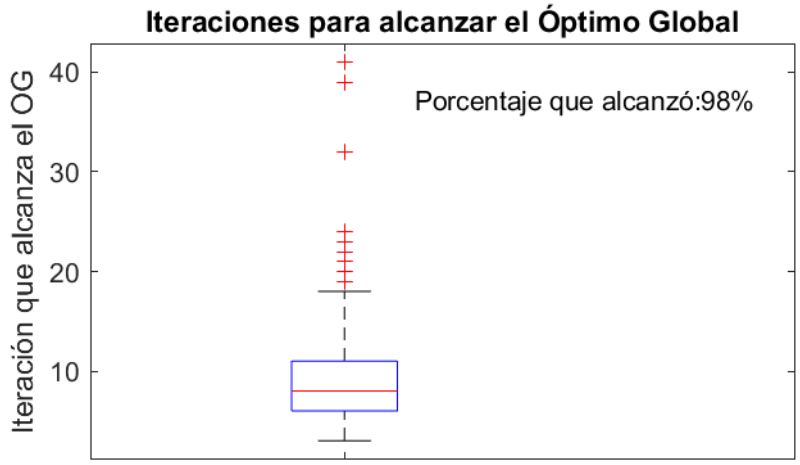


*Figura 5.32: Iteraciones para alcanzar el óptimo con algoritmo BA para el Caso 2*

La tabla 5.24 muestra los parámetros óptimos de configuración y la 5.25 los estadísticos correspondientes.

| Cantidad de Murciélagos | Iteraciones | Frecuencia Mínima $f_{min}$ | Frecuencia máxima $f_{max}$ | Factor de Inercia $w$ | Volumen inicial $A_0$ | Constante de Disminución de volumen $\alpha$ | Constante aumento del pulso $\gamma$ |
|---|---|---|---|---|---|---|---|
| 100 | 50 | 0 | 7 | 0 | 1.5 | 0.99 | 0.5 |

*Tabla 5.24: Parámetros óptimos de configuración del algoritmo BA para el Caso 2*

Con dichos parámetros los resultados estadísticos para 100 ensayos son:

| **Estadístico** | **Valor** |
|---|---|
| Mínimo | 55 |
| Efectividad | 98% |
| Media | 55.2 |
| Desviación Estándar | 1.41 |
| Varianza | 1.98 |
| Coeficiente de Variación | 0.025 |
| **Cuartiles** | |
| q1 | 55 |
| q2 | 55 |
| q3 | 55 |
| **Tabla de Frecuencias** | |
| Value Count Percent<br>55 98 98.00%<br>65 2 2.00% | |

*Tabla 5.25: Datos estadísticos del algoritmo BA para el Caso 2*

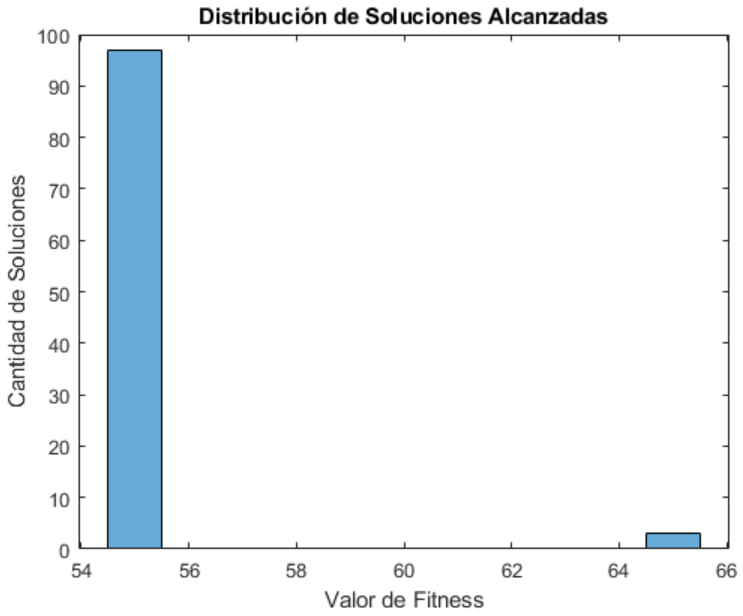


*Figura 5.33: Distribución de soluciones del algoritmo BA para el Caso 2*

La evolución del algoritmo continúa teniendo características similares, con pendientes pronunciadas y pocos o nulos estancamientos.

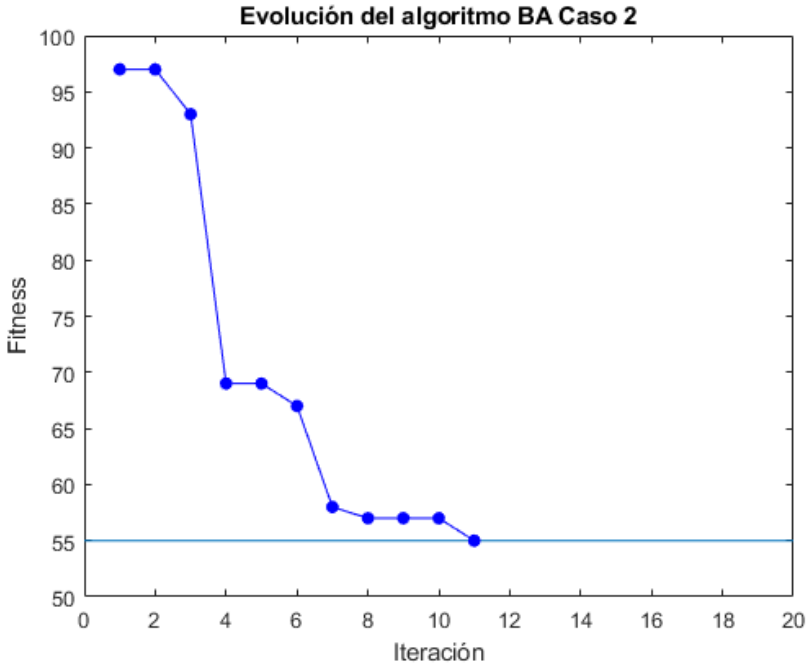


*Figura 5.34: Evolución típica del fitness para un ensayo del algoritmo BA para el Caso 2*

Para el caso 2 tampoco es necesario un post-optimizador por lo que se omiten las pruebas con el algoritmo BPA.

### 5.3.5 Comparación del desempeño los algoritmos para el Caso 2

Para el tamaño de la red del caso 2, los resultados comparativos resultan similares al caso 1. El desempeño de los algoritmos podría considerarse estadísticamente iguales en cuanto a la obtención de resultados, y su comparación simple y la diferencia son solo algunos casos atípicos.

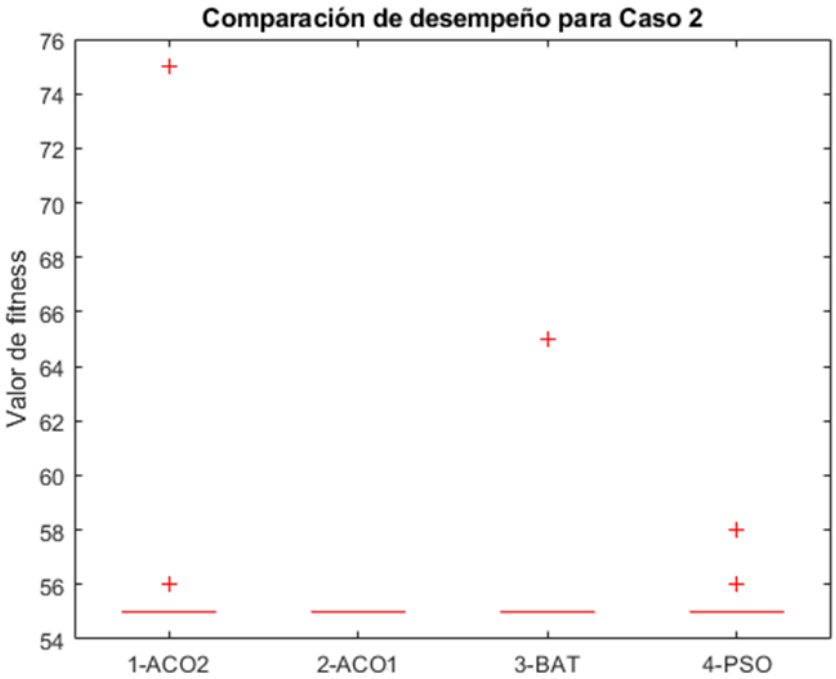


*Figura 5.35: Comparativa desempeño de los algoritmos para el Caso 2*

Siguiendo el mismo análisis que en el caso 1, se realiza un análisis de tiempos donde radica la mayor diferencia, lo que se observa en la tabla 5.26.

| Algoritmo | PSO | ACO 1 | ACO 2 | BA | BPA |
|---|---|---|---|---|---|
| **Porcentaje de efectividad** | 94% | 100% | 96% | 98% | - |
| **Tiempo medio para obtener óptimo** | 107.55 | 546.17 | 2.77 | 3.98 | - |
| **Tiempo mediano para obtener el óptimo** | 66.90 | 552.95 | 2.52 | 2.97 | - |
| **Tiempo medio para obtener una solución** | 414.19 | 933.53 | 5.29 | 20.11 | - |
| **Tiempo mínimo para obtener el óptimo** | 5.49 | 429.10 | 1.22 | 0.96 | - |
| **Tiempo máximo para obtener el óptimo** | 418.48 | 716.08 | 8.80 | 21.84 | - |

*Tabla 5.26: Comparativa del tiempo empleado por los algoritmos Caso 2.*

La figura 5.36 muestra la comparación de los tiempos necesarios para alcanzar el óptimo global para 100 ensayos del algoritmo. El algoritmo ACO 1 claramente presenta los mayores tiempos para encontrar la mejor solución, y en este caso, PSO también presenta tiempos altos.

En la figura 5.37 se cambia la escala para observar el comportamiento de las otras dos estrategias. En este caso los mejores tiempos se encontraron con el algoritmo ACO 2 seguido por BA, con valores similares.

Nuevamente el algoritmo distribución de flujos ACO 1 entrega soluciones con un balance de cargas mayor a las otras estrategias.

Como era esperable, se produce un incremento de los tiempos en todos los algoritmos siendo más notable en el algoritmo PSO, el que presenta un mayor crecimiento debido a la gran cantidad de partículas necesarias para obtener resultados aceptables.

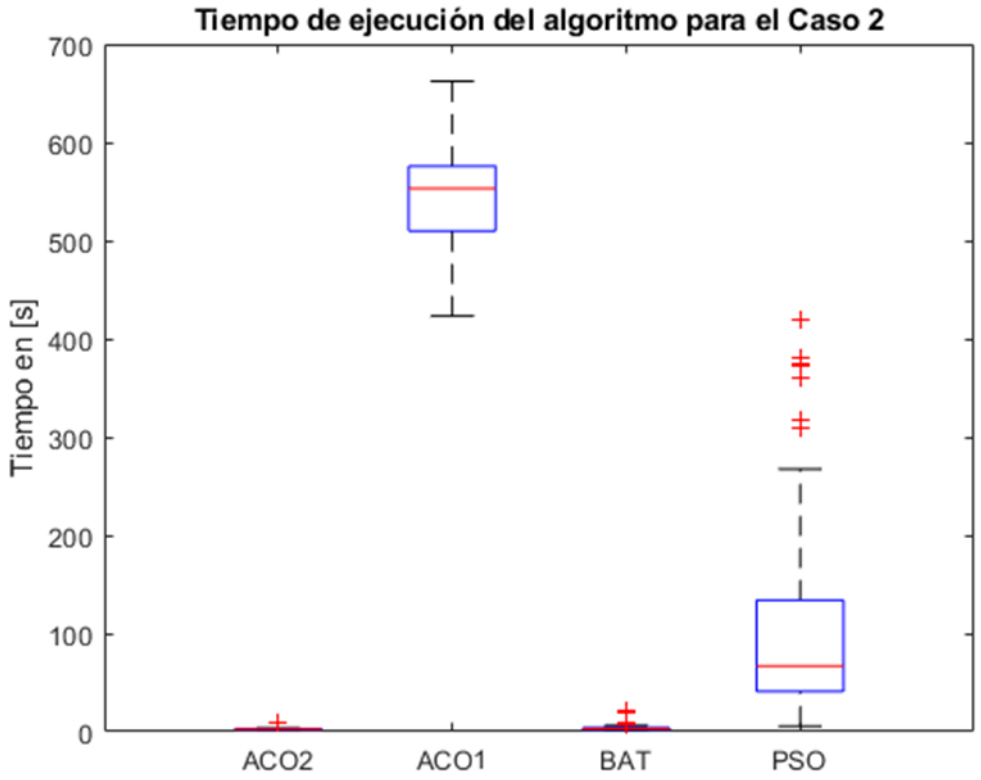


*Figura 5.36: Comparativa de tiempo de ejecución para encontrar el óptimo global (escala 0 a 700) Caso 2*

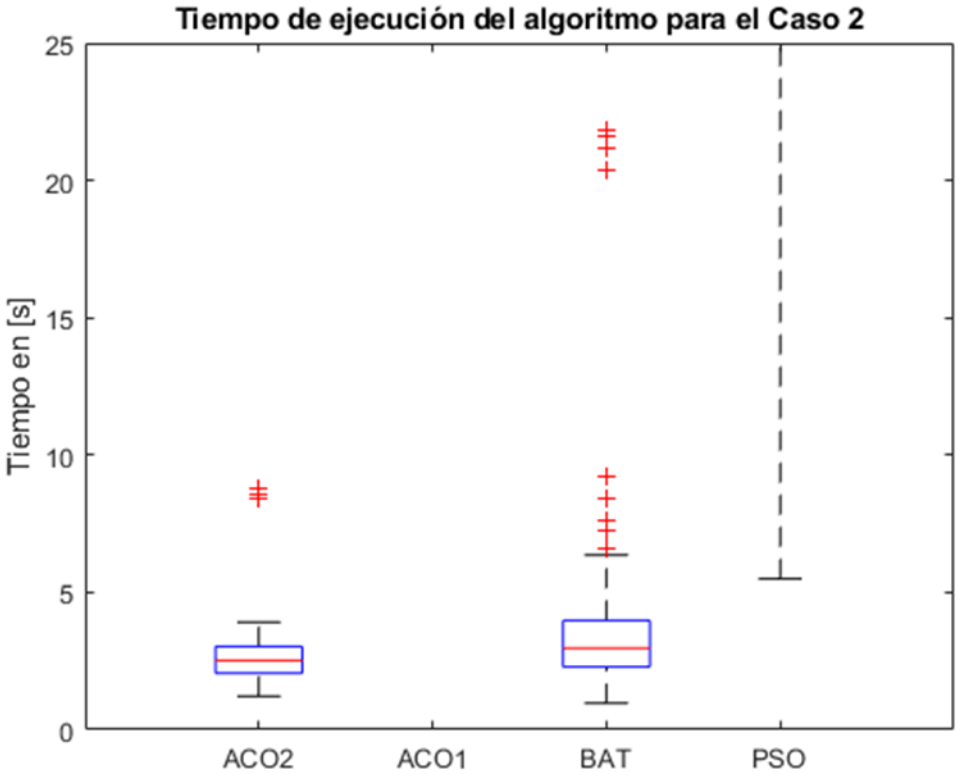


*Figura 5.37: Comparativa de tiempo de ejecución para encontrar el óptimo global (escala 0 a 25) Caso 2*

## 5.4 Experimentación con CASO 3 – Red de 8 nodos

Para la red planteada como Caso 3 (figura 5.3), las posibilidades de asignación de flujo crecen enormemente y con ellos los tiempos empleados por los algoritmos, que comienzan a ser significativos.

Con la experiencia de los casos anteriores, se determinó el correcto funcionamiento de los algoritmos y se describió la influencia de los parámetros por lo que éstos pueden ser estimados, para redes similares con unas pocas ejecuciones.

Para este caso, como se determinó en la introducción, el vector solución está compuesto por 638 componentes, un número considerablemente mayor que en los casos anteriores, lo que trae aparejado tiempos de cómputo mayores. La experimentación se centrará en la comparación de ensayos de los algoritmos que mejor se desempeñan en una red de estas características, es decir, los métodos más rápidos.

La técnica de distribución de flujos basada en colonia de hormigas (Algoritmo ACO 1) no se tomará en cuenta, debido a su tiempo excesivo de ejecución. Algunos ensayos arrojaron tiempos de ejecución del orden de 7.5 h por ensayo, lo que llevaría más de un mes realizar un ensayo de 100 ejecuciones. El descartar este algoritmo con estas instancias no resta importancia al algoritmo, ya que tuvo resultados interesantes para las redes chicas, y realizando un balanceo de cargas que, si bien no fue objeto de esta tesis, para ciertas situaciones podría ser deseable. Los tiempos de ejecución podrían mejorarse notablemente, si se codifican en lenguajes que operen con mayor celeridad, compilados y paralelizables cuyo estudio se propondrá para trabajos futuros.

### 5.4.1 Ensayos del algoritmo PSO (Caso 3)

El algoritmo PSO también mostró tiempos significativos comparados con las otras técnicas en la ejecución de los ensayos para el caso 2, aunque considerablemente más chicos que el algoritmo ACO 1.

Con la experiencia de los casos anteriores, al tener instancias mayores del problema, la lógica indicaría que se deberían incrementar aún más la cantidad de partículas y la cantidad de iteraciones, lo que transformaría la técnica en inviable al igual que ACO 1 debido al tiempo de ejecución.

A efectos de no descartar el algoritmo y entender si ajustando los parámetros podría entregar resultados satisfactorios, se realizan los ensayos con parámetros pequeños para ver su funcionamiento en tiempos razonables, sobre la red de este caso.

El ensayo se realiza con los siguientes parámetros:

| Ensayo | Cantidad de Partículas | Iteraciones | Factor de Aprendizaje individual *C1* | Factor de Aprendizaje Social *C2* | Factor de Inercia *w* |
|---|---|---|---|---|---|
| 1-PSO | 100 | 150 | 8 | 0.3 | 1 |

*Tabla 5.27: Parámetros de configuración ensayo PSO para el Caso 3*

El resultado del ensayo se observa en el siguiente gráfico de caja y bigote y los resultados estadísticos se plasman en la tabla 5.28

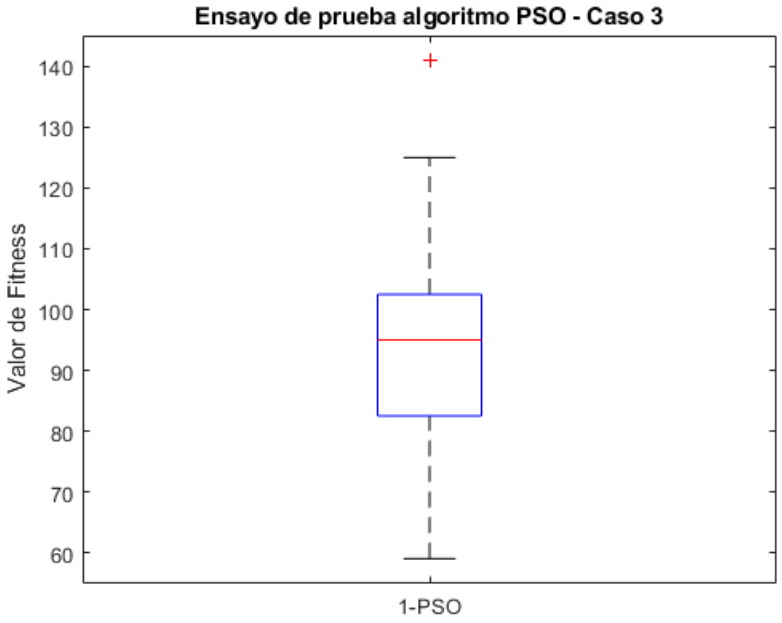


*Figura 5.38 – Ensayos PSO Caso 3 para distintos parámetros.*

| Estadístico | Valor |
|---|---|
| Mínimo | 59 |
| Efectividad | 0% |
| Media | 92.83 |
| Desviación Estándar | 14.99 |
| Varianza | 224.8 |
| Coeficiente de Variación | 0.16 |

| Cuartiles | |
|---|---|
| q1 | 82 |
| q2 | 95 |
| q3 | 102 |

| Tabla de Frecuencias | | |
|---|---|---|
| Value | Count | Percent |
| 59 | 1 | 1.00% |
| 60 | 2 | 2.00% |
| 65 | 1 | 1.00% |
| 69 | 1 | 1.00% |
| 70 | 3 | 3.00% |
| 71 | 1 | 1.00% |
| 73 | 1 | 1.00% |
| 74 | 1 | 1.00% |
| 75 | 6 | 6.00% |
| 77 | 1 | 1.00% |
| 78 | 2 | 2.00% |
| 80 | 5 | 5.00% |
| 85 | 7 | 7.00% |
| 86 | 2 | 2.00% |
| 88 | 2 | 2.00% |
| 90 | 7 | 7.00% |
| 91 | 1 | 1.00% |
| 92 | 2 | 2.00% |
| 93 | 1 | 1.00% |
| 94 | 1 | 1.00% |
| 95 | 9 | 9.00% |
| 96 | 2 | 2.00% |
| 97 | 1 | 1.00% |
| 98 | 2 | 2.00% |
| 99 | 1 | 1.00% |
| 100 | 9 | 9.00% |
| 101 | 1 | 1.00% |
| 102 | 2 | 2.00% |
| 103 | 1 | 1.00% |
| 104 | 1 | 1.00% |
| 105 | 4 | 4.00% |
| 106 | 1 | 1.00% |
| 107 | 1 | 1.00% |
| 109 | 1 | 1.00% |
| 110 | 8 | 8.00% |
| 113 | 1 | 1.00% |
| 114 | 3 | 3.00% |
| 115 | 2 | 2.00% |
| 125 | 1 | 1.00% |
| 141 | 1 | 1.00% |

*Tabla 5.28: Datos estadísticos del algoritmo PSO para el Caso 3*

La distribución de soluciones queda representada en el siguiente histograma, donde se observa la dispersión de las soluciones.

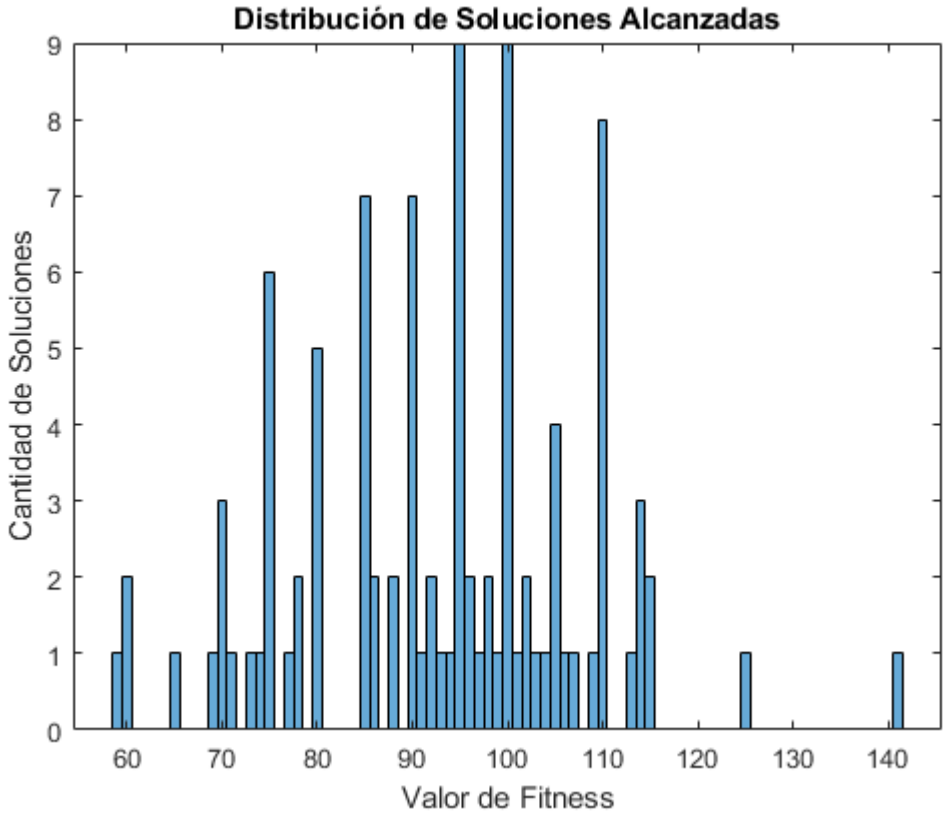


*Figura 5.39: Distribución de soluciones del algoritmo PSO para el Caso 3*

Los resultados obtenidos, si bien tienen un valor de aptitud aceptable, están lejos de un óptimo global. El algoritmo no tuvo la capacidad de encontrar la mejor solución del problema planteado con los parámetros dados, pero de todos modos ajustando los parámetros a costo de incrementar los tiempos se podría mejorar su rendimiento.

Aún para estos parámetros los tiempos empleados fueron altos. La figura 5.40 muestra la distribución de tiempos para los 100 ensayos realizados. Cada ejecución ronda los 20 minutos en promedio.

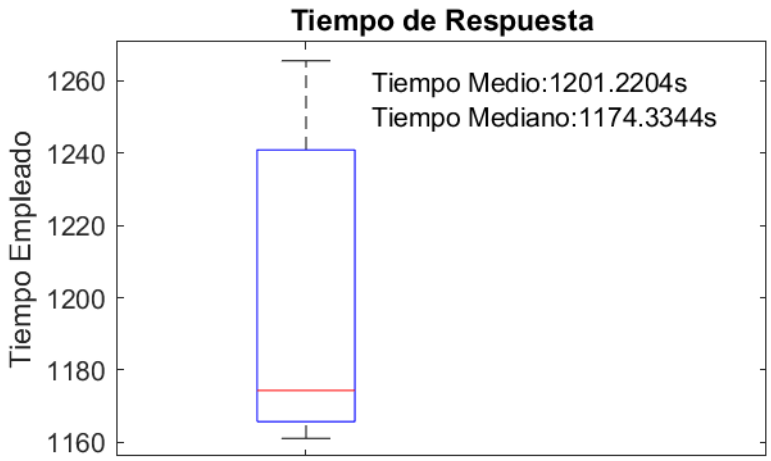


*Figura 5.40: Tiempo de respuesta para 100 ensayos del algoritmo PSO para el Caso 3*

En función del comportamiento del algoritmo PSO para los casos 1 y 2, donde la cantidad de iteraciones pasaron de 120 a 250 y la cantidad de partículas pasaron de 20 a 400 al agregar

dos nodos a la red, podría estimarse que al menos debería trabajarse con unas 1500 partículas y unas 500 iteraciones para comenzar a realizar algunas pruebas que arrojen mejores resultados.

Los tiempos para este tipo de parámetros se consideran inviables para el hardware y software utilizado por lo que se propondrá para futuros ensayos.

### 5.4.2 Ensayos del algoritmo ACO 2 – Redistribución de flujos (Caso 3)

Para el caso de ACO 2 se procedió de la misma forma, se ejecutó el algoritmo con algunos parámetros distintos, pero con una colonia de hormigas pequeña y pocas iteraciones, que, si bien se entiende que no serán suficientes para las dimensiones del caso, permitirá entender el funcionamiento del algoritmo en tiempos razonables.

Los ensayos se realizaron con los siguientes parámetros:

| Ensayo | Cantidad de Hormigas | Iteraciones | Coeficiente de prioridad de feromona $\alpha$ | Coeficiente de evaporación $\rho$ | Valor de Feromona inicial $\tau_0$ | Iteraciones para reiniciar | Efectividad |
|---|---|---|---|---|---|---|---|
| 1-ACO2 | 20 | 100 | -1 | 0.01 | 0.1 | 100 | 16% |
| 2-ACO2 | 20 | 100 | -1 | 0.01 | 1 | 100 | 0% |
| 3-ACO2 | 20 | 100 | -2 | 0.007 | 0.1 | 100 | 17% |
| 4-ACO2 | 20 | 100 | -2 | 0.01 | 0.1 | 100 | 14% |
| 5-ACO2 | 20 | 100 | -2 | 0.09 | 0.1 | 100 | 6% |
| 6-ACO2 | 20 | 100 | -2 | 0.1 | 0.1 | 100 | 12% |

*Tabla 5.29: Parámetros de configuración de ensayos ACO 2 para el Caso 1*

Si bien la efectividad es baja, todos los ensayos se realizaron con parámetros del algoritmo que garantizaban una ejecución rápida.

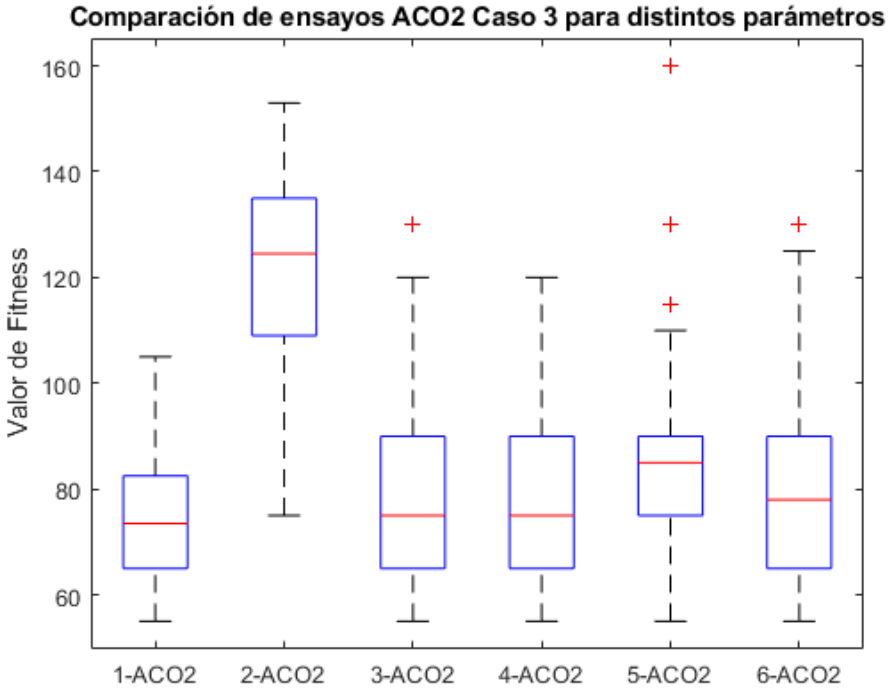


*Figura 5.41 – Ensayos ACO 2 Caso 3 para distintos parámetros.*

El ensayo 3-ACO2 tiene una efectividad de una unidad porcentual menor al 1-ACO2, pero los valores estadísticos de este ensayo son mejores, y se muestran en la siguiente tabla:

| Estadístico | Valor |
|---|---|
| Mínimo | 55 |
| Efectividad | 16% |
| Media | 74.17 |
| Desviación Estándar | 13.52 |
| Varianza | 182.83 |
| Coeficiente de Variación | 0.1823 |
| **Cuartiles** | |
| q1 | 65 |
| q2 | 73.5 |
| q3 | 82.5 |
| **Tabla de Frecuencias** | |

| Value | Count | Percent |
|---|---|---|
| 55 | 16 | 16.00% |
| 56 | 2 | 2.00% |
| 59 | 1 | 1.00% |
| 60 | 1 | 1.00% |
| 61 | 1 | 1.00% |
| 65 | 9 | 9.00% |
| 66 | 1 | 1.00% |
| 68 | 2 | 2.00% |
| 69 | 1 | 1.00% |
| 70 | 11 | 11.00% |
| 71 | 1 | 1.00% |
| 73 | 4 | 4.00% |
| 74 | 1 | 1.00% |
| 75 | 6 | 6.00% |
| 76 | 3 | 3.00% |
| 77 | 2 | 2.00% |
| 78 | 2 | 2.00% |
| 79 | 1 | 1.00% |
| 80 | 6 | 6.00% |
| 81 | 3 | 3.00% |
| 82 | 1 | 1.00% |
| 83 | 1 | 1.00% |
| 84 | 1 | 1.00% |
| 85 | 1 | 1.00% |
| 86 | 1 | 1.00% |
| 87 | 2 | 2.00% |
| 88 | 1 | 1.00% |
| 89 | 1 | 1.00% |
| 90 | 5 | 5.00% |
| 91 | 2 | 2.00% |
| 94 | 4 | 4.00% |
| 96 | 1 | 1.00% |
| 100 | 1 | 1.00% |
| 102 | 1 | 1.00% |
| 105 | 3 | 3.00% |

*Tabla 5.30: Datos estadísticos del algoritmo ACO 2 para el Caso 3*

La distribución de soluciones se representa en la figura 5.42, donde se observan alguno óptimos locales fuertes con valores 65 y 70.

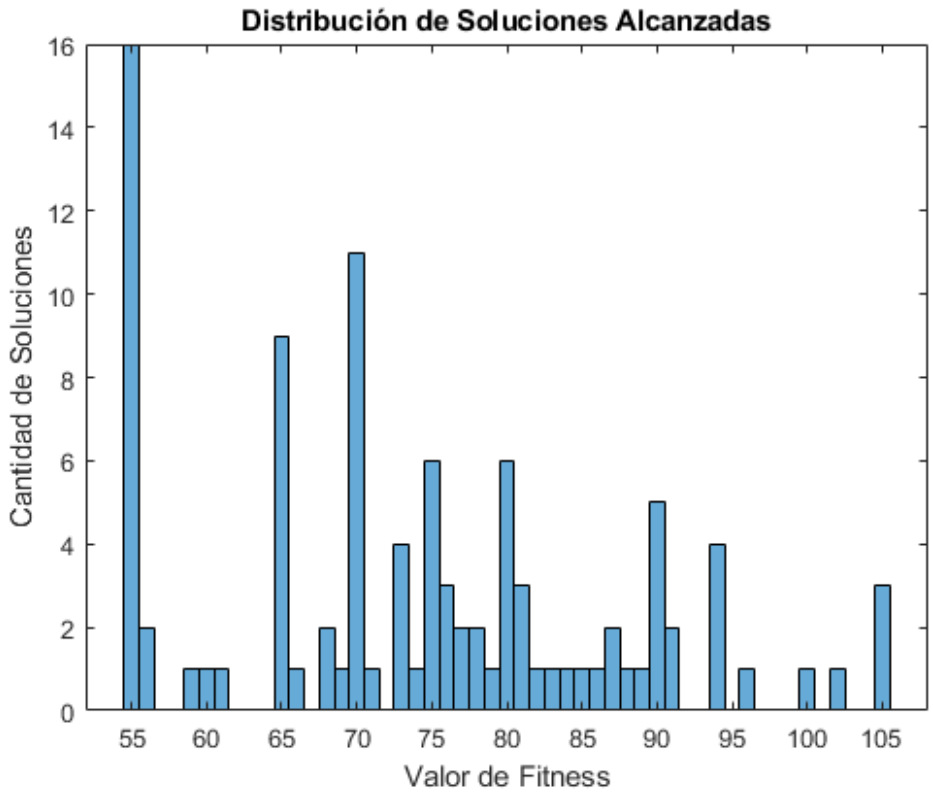


*Figura 5.42: Distribución de soluciones del algoritmo ACO 2 para el Caso 3*

El algoritmo aportaría mejores soluciones al aumentar la cantidad de iteraciones y el tamaño de la colonia de hormigas a costa de aumentar el tiempo de ejecución. Para estos parámetros, el tiempo empleado por el algoritmo se observa en la figura 5.43.

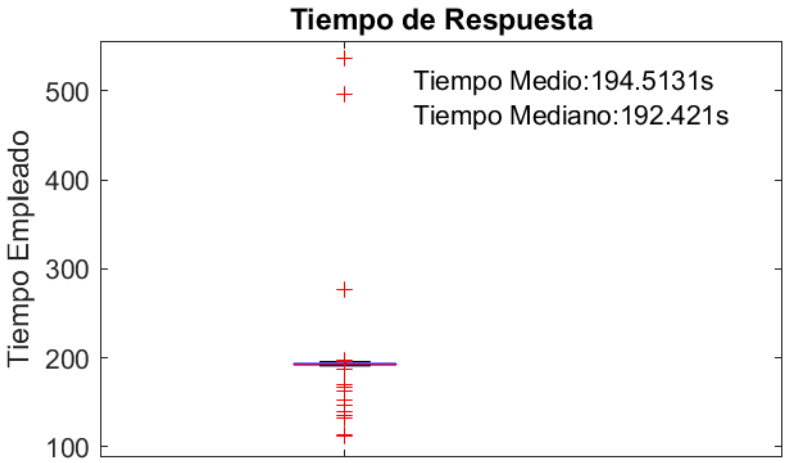


*Figura 5.43: Tiempos de respuesta del algoritmo ACO 2 para el Caso 3*

## 5.4.3 Ensayos del algoritmo de murciélagos (BA) (Caso 3)

En función de ensayos previos y a efectos de determinar el mejor valor de volumen inicial para este caso, se realizaron dos ensayos con los parámetros de la tabla 5.31 y cuyos resultados se exponen gráficamente en la figura 5.44.

| Ensayo | Cantidad de Murciélagos | Frecuencia Mínima $f_{min}$ | Frecuencia máxima $f_{max}$ | Factor de Inercia $w$ | Volumen inicial $A_0$ | Constante de Disminución de volumen α | Constante aumento del pulso γ | Eficiencia |
|---|---|---|---|---|---|---|---|---|
| 1-BA | 100 | 0 | 5 | 0 | 1.2 | 0.9 | 0.5 | 44% |
| 2-BA | 100 | 0 | 5 | 0 | 1.5 | 0.9 | 0.5 | 46% |

*Tabla 5.31: Parámetros de configuración de ensayos BA para el Caso 3*

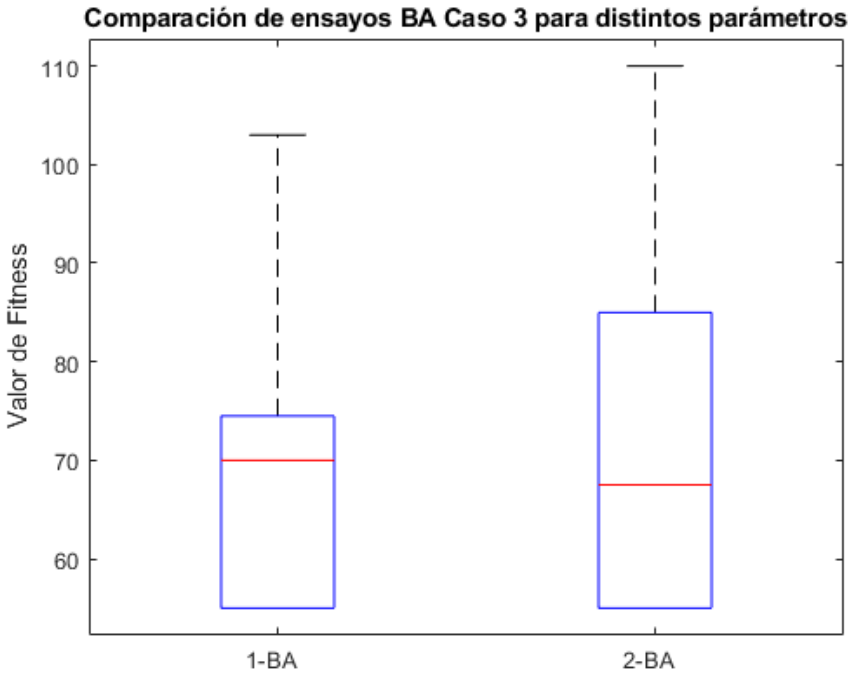


*Figura 5.44 – Ensayos BA Caso 3 para distintos parámetros.*

Mayores valores de volumen inicial $A_0$ arroja mejor efectividad del algoritmo. Sin embargo, los valores estadísticos son mejores el valor de volumen inicial menor, lo que es visible en la figura anterior. No obstante, la diferencia de efectividad es ínfima, y estadísticamente podríamos considerar que los resultados son iguales.

El algoritmo se ejecuta con buena velocidad para estos parámetros, y se determina que la cantidad de iteraciones no es un parámetro de influencia determinante en la obtención de resultados. Como se muestra en la figura 5.45, salvo algún caso atípico, 40 iteraciones son suficientes, por lo que el desempeño del algoritmo debe mejorarse ajustando los otros parámetros, por ejemplo, el tamaño de la población, aunque el incremento de este valor, traiga aparejados mayores tiempos de ejecución.

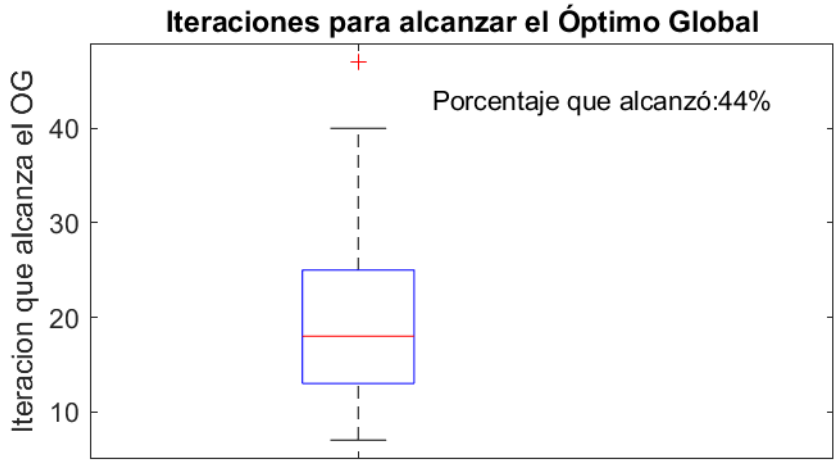


*Figura 5.45: Iteraciones para alcanzar el óptimo con algoritmo BA para el Caso 3*

Los parámetros estadísticos para el ensayo 1-BA se muestran en la siguiente tabla:

| Estadístico | Valor |
|---|---|
| Mínimo | 55 |
| Efectividad | 44% |
| Media | 67.93 |
| Desviación Estándar | 14.12 |
| Varianza | 199.36 |
| Coeficiente de Variación | 0.21 |
| **Cuartiles** | |
| q1 | 55 |
| q2 | 70 |
| q3 | 74.5 |
| **Tabla de Frecuencias** | |

| Value | Count | Percent |
|---|---|---|
| 55 | 44 | 44.00% |
| 65 | 4 | 4.00% |
| 70 | 26 | 26.00% |
| 74 | 1 | 1.00% |
| 75 | 1 | 1.00% |
| 80 | 4 | 4.00% |
| 82 | 1 | 1.00% |
| 85 | 5 | 5.00% |
| 90 | 8 | 8.00% |
| 95 | 1 | 1.00% |
| 99 | 1 | 1.00% |
| 100 | 3 | 3.00% |
| 103 | 1 | 1.00% |

*Tabla 5.32: Datos estadísticos del algoritmo BA para el Caso 1*

Se observa un óptimo global muy fuerte con valor 70 que suma el 26% de las soluciones.

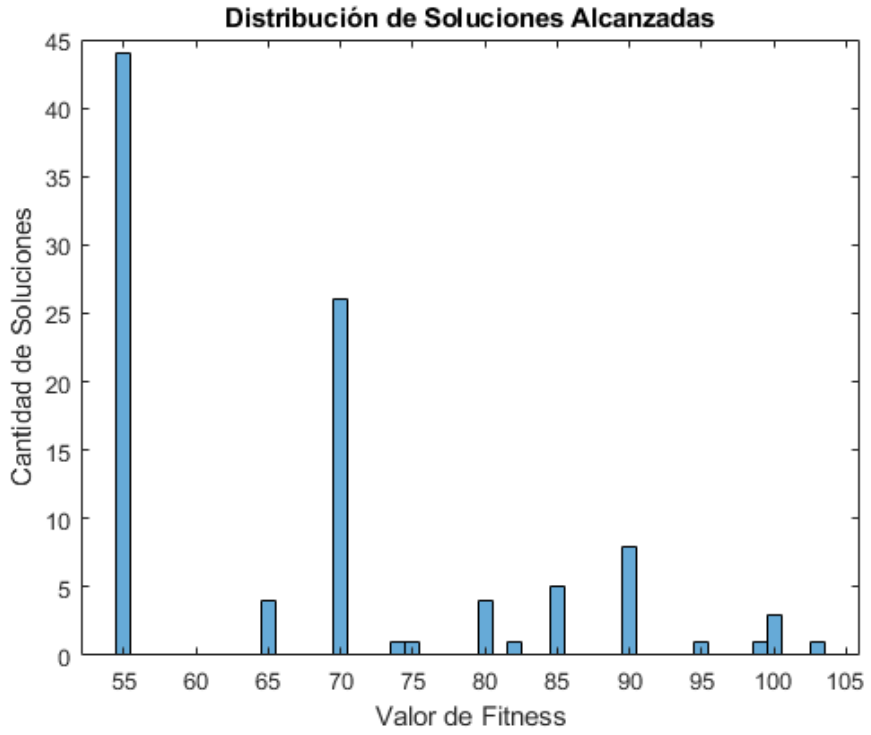


*Figura 5.46: Distribución de soluciones del algoritmo BA para el Caso 3*

La evolución del algoritmo tiene similitud con los avances descriptos para los casos anteriores. El avance posee pendientes pronunciadas y normalmente no se producen estancamientos. Los estancamientos se producen en los óptimos locales que hacen que el algoritmo no evolucione hacia la solución global. Una evolución típica para un ensayo que alcanza la solución global en 14 iteraciones se muestra a continuación.

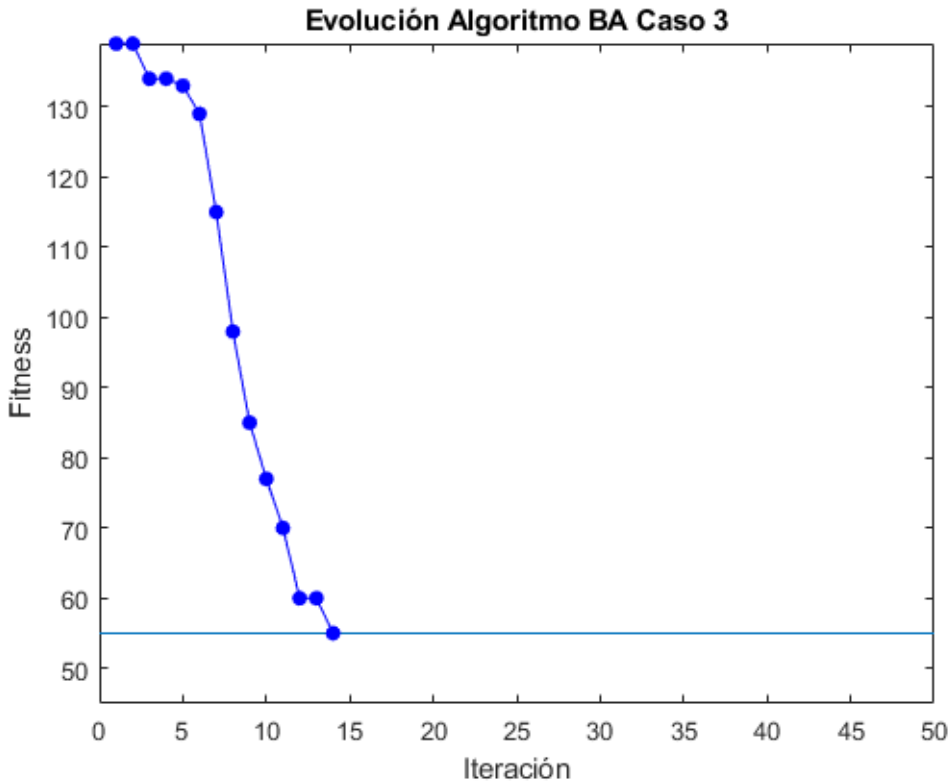


*Figura 5.47: Evolución típica del fitness para un ensayo del algoritmo BA para el Caso 3*

### 5.4.4 Ensayos del algoritmo híbrido murciélago - hormiga (BPA) (Caso 3)

Para esta instancia del problema planteado, y debido a la baja efectividad del mejor de los algoritmos ensayado, se procede al ensayo del algoritmo BPA.

El algoritmo BPA, fue expuesto en la sección 4.5, y es un híbrido entre el algoritmo BA y el ACO 2. Técnicamente el algoritmo encuentra una solución con el mecanismo de murciélagos y luego intenta optimizar dicha solución redistribuyendo los flujos a partir de la solución encontrada por BA, que de por sí, es una solución aceptable.

El algoritmo BPA se configurará con parámetros subóptimos para las técnicas individuales, pero que arrojaron resultados satisfactorios para esta hibridación logrando mejores tiempos en su ejecución. Es decir, que la hibridación permite bajar los parámetros de población de los enjambres y número de iteraciones, lo que dará como resultado final un tiempo inferior al que se conseguiría con las técnicas individuales configurada con los parámetros óptimos.

La figura siguiente expone la comparativa de los resultados obtenidos de la experimentación de BA y las dos instancias de ensayos de BPA con las que se logra alcanzar excelentes resultados.

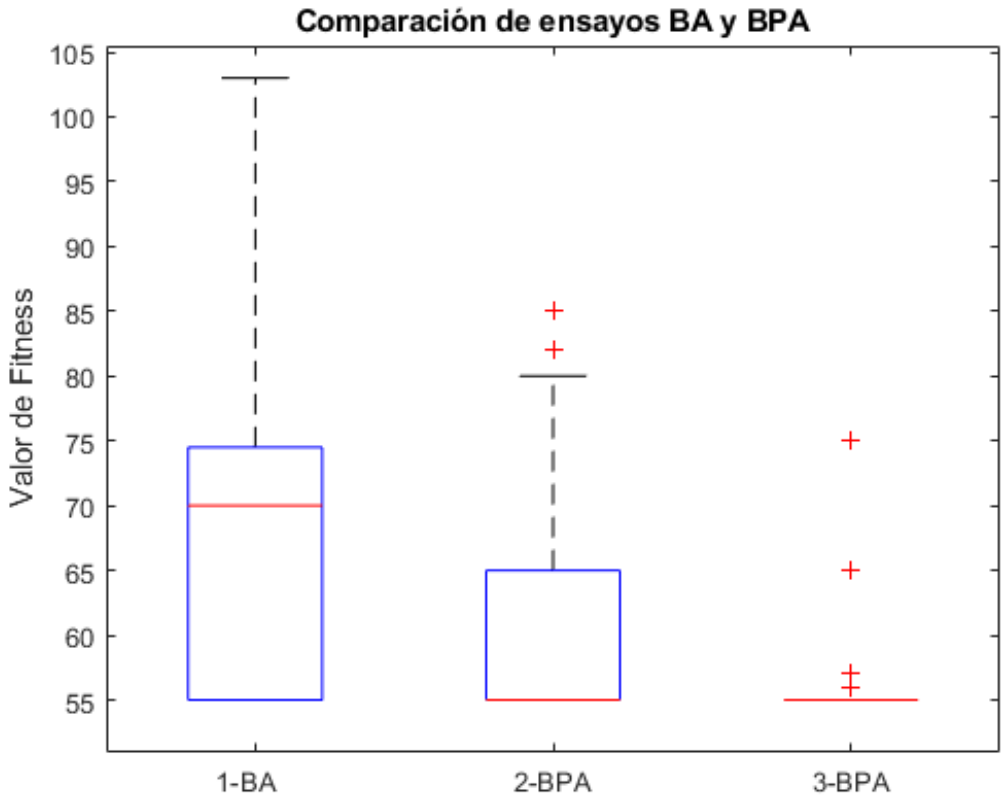


*Figura 5.48 – Ensayos BA y BPA Caso 3 para distintos parámetros.*

Los parámetros de los ensayos anteriores se muestran en la siguiente tabla

| | Ensayo | 1-BA | 2-BPA | 3-BPA |
|---|---|---|---|---|
| Parámetros de Murciélagos | Cantidad de Murciélagos | 100 | 100 | 100 |
| | Iteraciones | 50 | 50 | 50 |
| | Frecuencia Mínima $f_{min}$ | 0 | 0 | 0 |
| | Frecuencia máxima $f_{max}$ | 5 | 5 | 5 |
| | Factor de Inercia $w$ | 0 | 0 | 0 |
| | Volumen inicial $A_0$ | 1.2 | 1.2 | 1.2 |
| | Constante de Disminución de volumen **α** | 0.9 | 0.9 | 0.9 |
| | Constante aumento del pulso **γ** | 0.5 | 0.5 | 0.5 |
| Parámetros de Hormigas | Cantidad de Hormigas | NC | 20 | 20 |
| | Iteraciones | NC | 100 | 300 |
| | Coeficiente de prioridad de feromona **α** | NC | -1 | -1 |
| | Coeficiente de prioridad información heurística β | NC | 1 | 1 |
| | Coeficiente de evaporación $\rho$ | NC | 0.01 | 0.01 |
| | Valor de Feromona inicial $\tau_0$ | NC | 0.1 | 0.1 |
| | Iteraciones para reiniciar | NC | 100 | 100 |

*Tabla 5.33: Parámetros de configuración de ensayos BA y BPA para el Caso 3*

El ensayo 3-BPA alcanzó el 94% de efectividad. El 46% fue alcanzado directamente por la estrategia inspirada en murciélagos, y el pos-optimizador ACO 2, redistribuyó los flujos, ampliando el 46% a 94% de efectividad.

La tabla 5.34 muestra los valores estadísticos para este ensayo y la figura 5.49 la distribución de soluciones

| Estadístico | Valor |
|---|---|
| Mínimo | 55 |
| Efectividad | 94% |
| Media | 55.44 |
| Desviación Estándar | 2.43 |
| Varianza | 5.9257 |
| Coeficiente de Variación | 0.044 |
| **Cuartiles** | |
| q1 | 55 |
| q2 | 55 |
| q3 | 55 |
| **Tabla de Frecuencias** | |
| Value Count Percent | |
| 55 94 94.00% | |
| 56 2 2.00% | |
| 57 1 1.00% | |
| 65 2 2.00% | |
| 75 1 1.00% | |

*Tabla 5.34: Datos estadísticos del algoritmo BPA para el Caso 3*

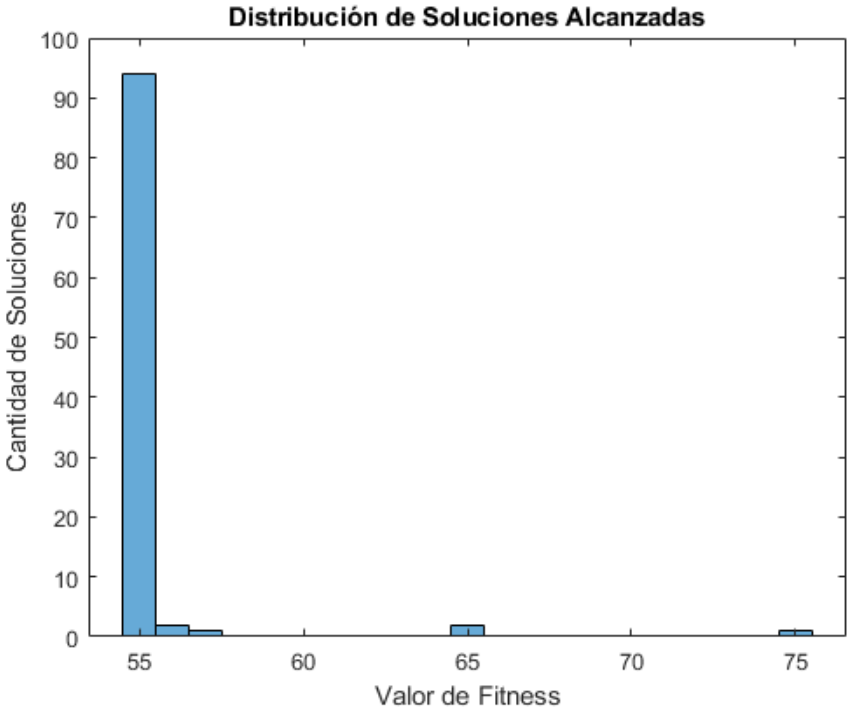


*Figura 5.49: Distribución de soluciones del algoritmo BPA para el Caso 3*

Los tiempos de respuesta del algoritmo se resumen en el siguiente gráfico. Como tiempo medio se necesitan en el orden de 14 minutos para obtener el óptimo, sin embargo, existen tiempos que si bien son atípicos ascienden hasta aproximadamente dos horas para obtener el resultado. Si bien estos tiempos son mayores al algoritmo BA para los parámetros ensayados, garantizan un mayor nivel de efectividad.

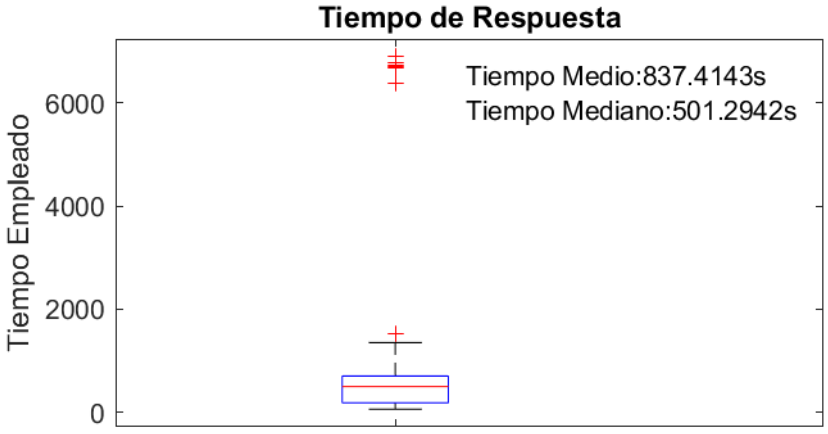


*Figura 5.50: Tiempos de respuesta del algoritmo BPA para el Caso 3*

El avance típico en la mejora del fitness refleja el resultado de la hibridación y se muestra en la figura 5.51

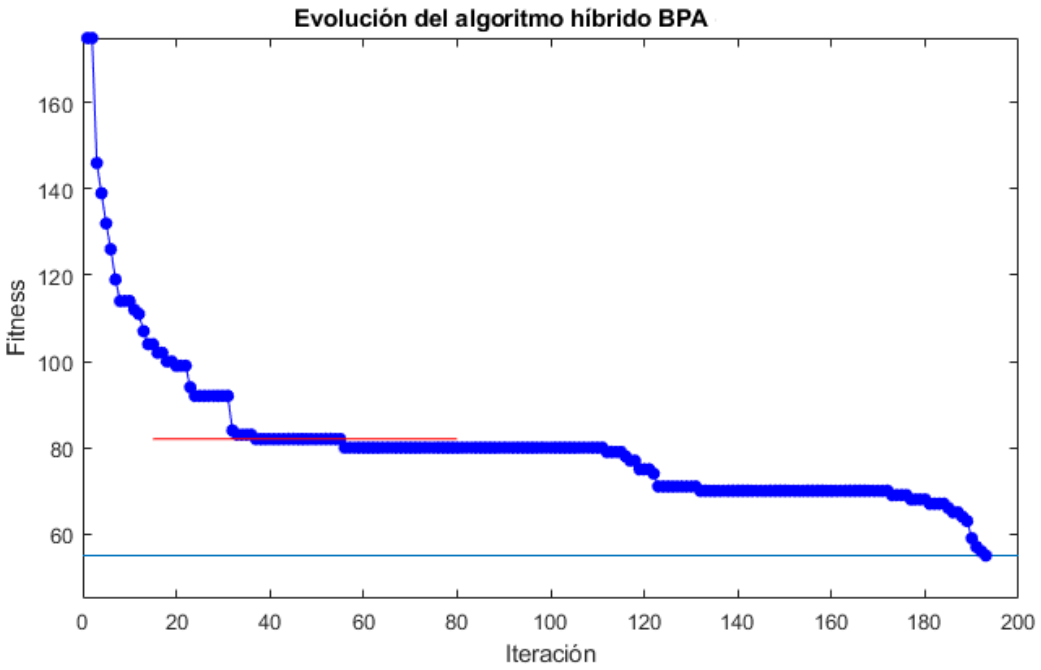


*Figura 5.51: Evolución típica del fitness para un ensayo del algoritmo BPA para el Caso 3*

La línea roja divide el accionar del enjambre de murciélagos en la parte superior y de la colonia de hormigas en la parte inferior. La porción BA registra una pendiente abrupta con pocos estancamientos típico de este algoritmo, mientas que la porción de post-optimización desde la iteración 50 en adelante, presenta el típico escalonamiento ACO 2.

### 5.4.5 Comparación del desempeño de los algoritmos para el Caso 3

La comparación de los algoritmos ya no resulta simple como los casos anteriores. La mayor diferencia continúa siendo el tiempo de ejecución, aunque en este caso, para los parámetros ensayados, se encuentra una diferencia sustancial en los valores estadísticos de las soluciones entregadas.

Debido al excesivo tiempo empleado por la estrategia ACO que distribuye los flujos, se omite su inclusión para este caso.

Realizando un balance entre el tiempo de ejecución y las soluciones alcanzadas, la comparación de los resultados con los parámetros configurados se observa en la siguiente figura.

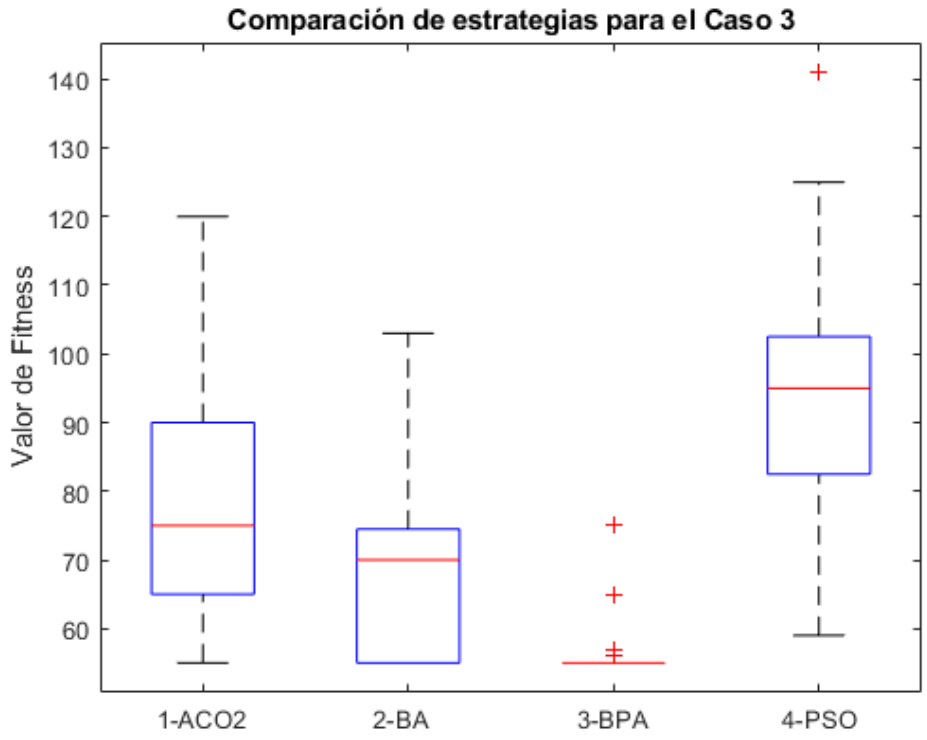


*Figura 5.52: Comparación de resultados de los diferentes algoritmos para el Caso 3*

Cabe destacar que probablemente existan parámetros que mejoren estos resultados, pero a costa de incrementar notablemente los tiempos de ejecución.

La comparación de los tiempos que demoran los algoritmos en entregar una solución se expone en la siguiente figura comparativa de cajas y bigotes.

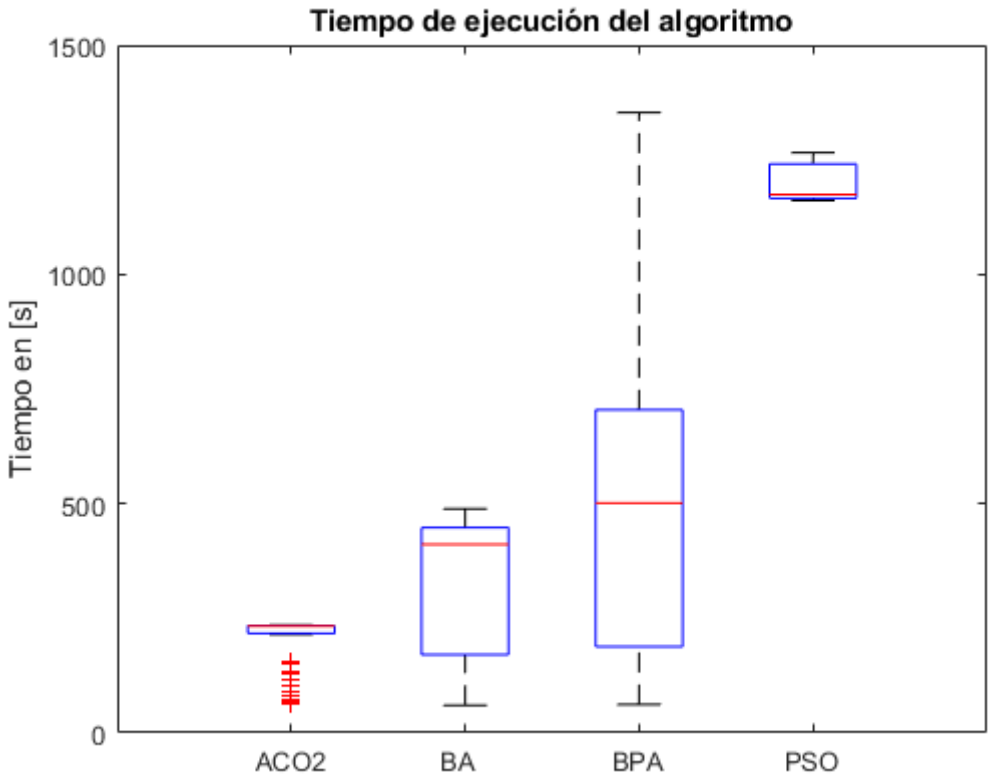


*Figura 5.53: Comparación del tiempo de ejecución de los algoritmos para el Caso 3*

El algoritmo PSO posee tiempos excesivos y una efectividad nula, por lo que escapa a la comparación. Con respecto a las otras tres estrategias, el tiempo empleado es directamente proporcional al porcentaje de efectividad. Se resumen los tiempos en la tabla 35 para un análisis comparativo.

| Algoritmo | PSO | ACO 1 | ACO 2 | BA | BPA |
|---|---|---|---|---|---|
| **Porcentaje de efectividad** | 0% | - | 14% | 44% | 94% |
| **Tiempo medio para obtener óptimo** | - | - | 211.6 | 323.9 | 837.4 |
| **Tiempo mediano para obtener el óptimo** | - | - | 232.8 | 411.0 | 501.3 |
| **Tiempo medio para obtener una solución** | 1265.5 | - | 236.3 | 477.7 | 6803.1 |
| **Tiempo mínimo para obtener el óptimo** | - | - | 64.9 | 59.9 | 62.0 |
| **Tiempo máximo para obtener el óptimo** | - | - | 235.0 | 488.7 | 6898.1 |

*Tabla 5.35: Comparativa del tiempo empleado por los algoritmos Caso 3.*

# 5.5 Experimentación con CASO 4 – Red de 10 nodos

La red del caso 4, una red de 10 nodos con una alta redundancia de interconectividad, plantea un problema de optimización ambicioso, donde las posibilidades combinatorias son realmente altas. La cantidad de caminos posibles para cada requerimiento se plantearon en la ecuación 5.9, y de acuerdo a la ecuación 3.27 la cantidad de posibilidades a evaluar es del orden de $2.33x10^{6047}$.

Las estrategias de PSO, ACO 1 y ACO 2 no serán evaluadas, ya que con el estudio de los casos anteriores podemos entender que no serán capaces de obtener buenos resultados en tiempos razonables. Se realizará la experimentación con el algoritmo BA y BPA

## 5.5.1 Ensayos del algoritmo de murciélagos (BA) (Caso 4)

El crecimiento exponencial en las posibilidades de solución hace que los tiempos de ejecución sufran un incremento importante. La experimentación con distintos parámetros resulta compleja por dichos tiempos. Para observar el desempeño del algoritmo, se ejecuta con parámetros que entregaron buenos resultados en la red del caso 3, lo que permitirá obtener resultados en tiempos aceptables.

La tabla 5.36 muestra los parámetros utilizados y la figura 5.54 el diagrama de caja y bigote para las soluciones alcanzadas.

| Ensayo | Cantidad de Murciélagos | Iter. | Frecuencia Mínima $f_{min}$ | Frecuencia máxima $f_{max}$ | Factor de Inercia $w$ | Volumen inicial $A_0$ | Constante de Disminución de volumen $\alpha$ | Constante aumento del pulso $\gamma$ | Ef. |
|---|---|---|---|---|---|---|---|---|---|
| 1-BA | 100 | 50 | 0 | 5 | 0 | 1.2 | 0.9 | 0.5 | 0% |

*Tabla 5.36: Parámetros de configuración de ensayos BA para el Caso 4*

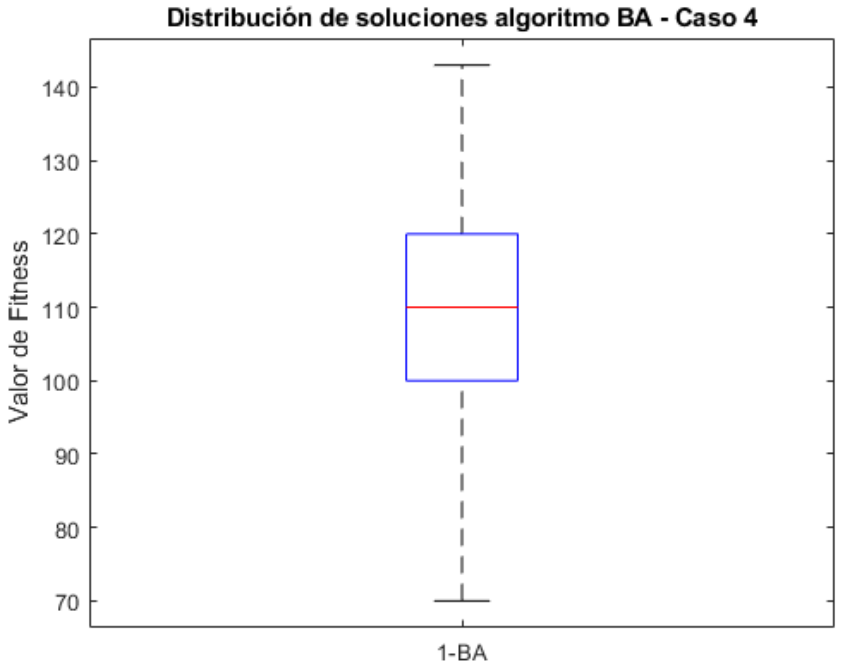


*Figura 5.54 – Ensayos BA Caso 4 para distintos parámetros.*

El algoritmo no fue capaz de alcanzar el óptimo global, aunque para esta instancia donde la complejidad del problema es excesivamente alta, las soluciones entregadas se consideran de buena calidad. Los valores estadísticos del ensayo se representan en la tabla 5.37.

| Estadístico | Valor |
|---|---|
| Mínimo | 70 |
| Efectividad | 0% |
| Media | 108.56 |
| Desviación Estándar | 16.49 |
| Varianza | 271.97 |
| Coeficiente de Variación | 0.15 |
| **Cuartiles** | |
| q1 | 100 |
| q2 | 110 |
| q3 | 120 |

**Tabla de Frecuencias**

| Value | Count | Percent |
|---|---|---|
| 70 | 5 | 5.00% |
| 85 | 8 | 8.00% |
| 90 | 4 | 4.00% |
| 95 | 3 | 3.00% |
| 100 | 17 | 17.00% |
| 105 | 2 | 2.00% |
| 110 | 23 | 23.00% |
| 114 | 1 | 1.00% |
| 115 | 2 | 2.00% |
| 120 | 16 | 16.00% |
| 123 | 1 | 1.00% |
| 125 | 5 | 5.00% |
| 128 | 1 | 1.00% |
| 129 | 2 | 2.00% |
| 130 | 4 | 4.00% |
| 134 | 1 | 1.00% |
| 135 | 3 | 3.00% |
| 141 | 1 | 1.00% |

| | | |
|---|---|---|
| 143 | 1 | 1.00% |

*Tabla 5.37: Datos estadísticos del algoritmo BA para el Caso 4*

Se observan óptimos locales fuertes con valor 100, 110 y 120 que acaparan más de la mitad de las soluciones. El 75% de las soluciones tienen un valor de aptitud menor que el óptimo local con valor 120. La figura 5.55 muestra la distribución de las soluciones gráficamente.

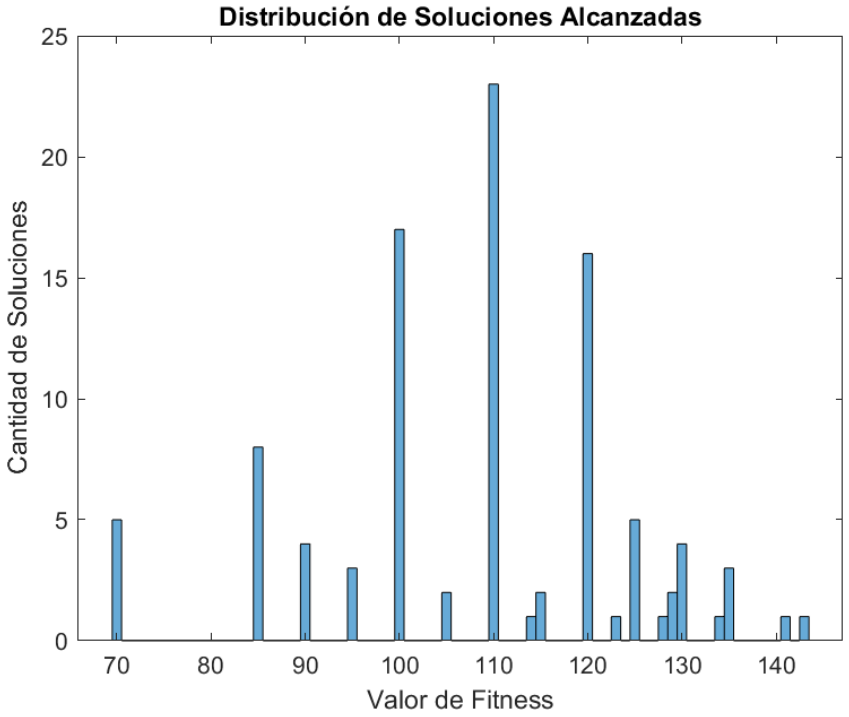


*Figura 5.55: Distribución de soluciones del algoritmo BA para el Caso 4*

Los tiempos de respuesta del algoritmo para toda la experimentación se expone en la figura 5.56. Los ensayos más rápidos tardaron 3 h 34 min, mientas que los más lentos tardaron 4 h 10m la media de los ensayos es de unas 3 h y 49m para obtener un resultado.

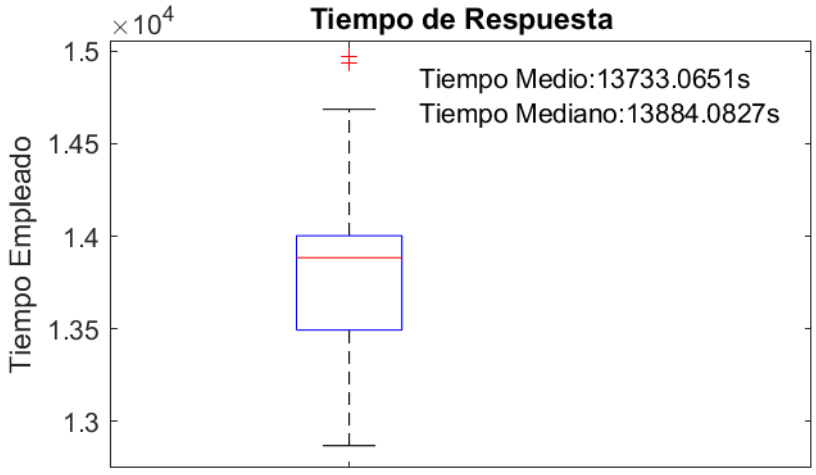


*Figura 5.56: Evolución típica del fitness para un ensayo del algoritmo BA para el Caso 3*

El crecimiento abrupto de tiempo para este caso, dificulta la realización de la misma cantidad de ensayos que en los casos anteriores, con la implementación aquí descripta. Es interesante notar que, para los mismos parámetros, el tiempo crece en forma considerable comparado con el caso 3, donde la media era de 14 minutos contra casi 3 h para este caso.

### 5.5.2 Ensayos del algoritmo híbrido murciélago - hormiga (BPA) (Caso 4)

Como se explicó con anterioridad, el algoritmo BPA tomas soluciones de BA y las optimiza mediante ACO 2. Para evitar los tiempos de ensayo mostrados en la sección anterior, se aplica directamente el post-optimizador a las soluciones del ensayo BA cuyas soluciones se muestran en la figura 5.54.

Los parámetros utilizados por el algoritmo BPA son los siguientes:

| | Ensayo | 1-BPA |
|---|---|---|
| Parámetros de Murciélagos | Cantidad de Murciélagos | 100 |
| | Iteraciones | 50 |
| | Frecuencia Mínima $f_{min}$ | 0 |
| | Frecuencia máxima $f_{max}$ | 5 |
| | Factor de Inercia $w$ | 0 |
| | Volumen inicial $A_0$ | 1.2 |
| | Constante de Disminución de volumen **α** | 0.9 |
| | Constante aumento del pulso **γ** | 0.5 |
| Parámetros de Hormigas | Cantidad de Hormigas | 30 |
| | Iteraciones | 3000 |
| | Coeficiente de prioridad de feromona **α** | -1 |
| | Coeficiente de prioridad información heurística β | 1 |
| | Coeficiente de evaporación $\rho$ | 0.01 |
| | Valor de Feromona inicial $\tau_0$ | 0.1 |
| | Iteraciones para reiniciar | 100 |

*Tabla 5.38: Parámetros de configuración de ensayos BPA para el Caso 4*

Los tiempos que utiliza el post-optimizador son elevados, llegando en algunos casos a 172 h lo que representa algo más de 7 días (Fig.5.57) Esto determinó que el ensayo se realice tomando una muestra de 34 soluciones BA a las que se le aplicó el post-optimizador ACO 2.

La figura 5.57 muestra los tiempos utilizados por el post-optimizador para los ensayos realizados.

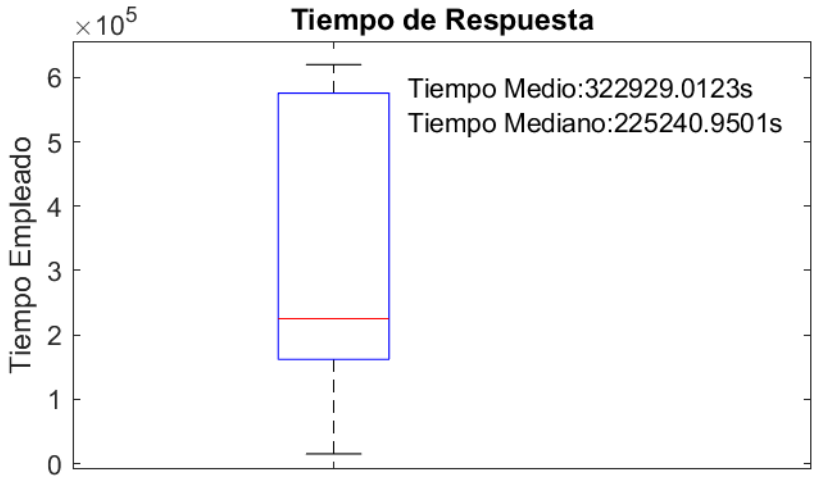


*Figura 5.57: Tiempos de ejecución algoritmo BPA para el Caso 4 para 10 ensayos*

Para determinar el tamaño de la muestra, se utilizaron herramientas estadísticas, donde se consideró el tamaño de la población (100), un nivel de confianza del 85%, y un error estimado de 10%. Con estos parámetros podemos inferir que los resultados obtenidos sobre la muestra son representativos de la población.

Las muestras ensayadas se tomaron en forma aleatoria y corresponden a diferentes valores de aptitud inicial correspondientes a las soluciones entregadas por el algoritmo BA. La tabla 5.39 se muestran los valores iniciales de fitness, la cantidad de muestras con dicho valor, y el valor porcentual de representación en la muestra a las cuales se les aplicó el post-optimizador.

| Fitness inicial (Solución BA) | Cantidad de muestras | Representación porcentual |
|---|---|---|
| 70 | 1 | 2.94% |
| 85 | 2 | 5.88% |
| 90 | 1 | 2.94% |
| 95 | 1 | 2.94% |
| 100 | 7 | 20.59% |
| 105 | 1 | 2.94% |
| 110 | 6 | 17.65% |
| 114 | 1 | 2.94% |
| 115 | 1 | 2.94% |
| 120 | 10 | 29.41% |
| 125 | 3 | 8.82% |

*Tabla 5.39: Distribución de valores de fitness intermedio algoritmo BPA Caso 4*

Considerando que el 87% de las soluciones entregadas por BA poseen un fitness menor a 125, por lo que la muestra aleatoria resulta representativa.

Los resultados de los ensayos se exponen en la Figura 5.58 y los valores estadísticos para las soluciones encontradas por el post-optimizador sobre la muestra se exponen en la tabla 5.40.

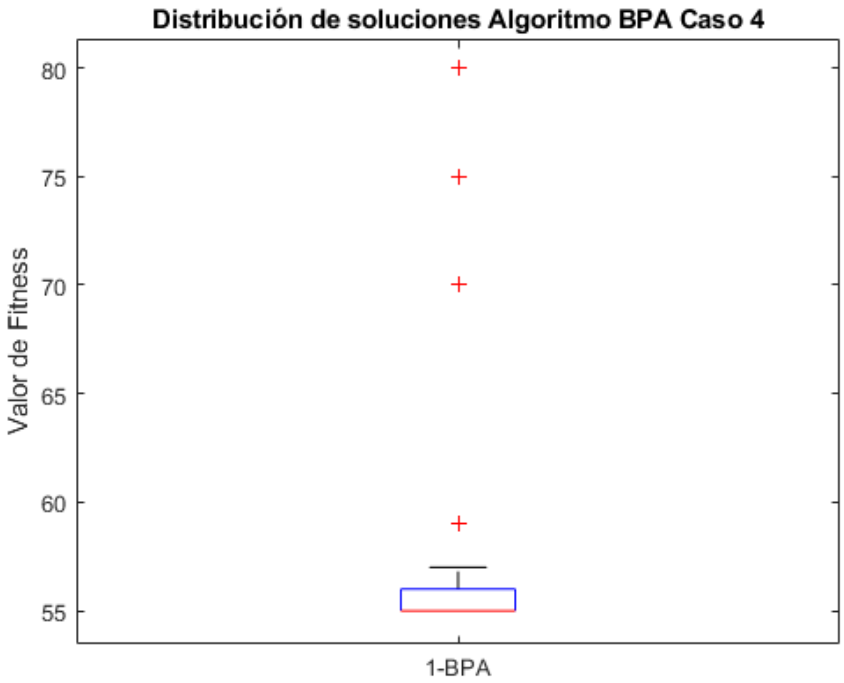


*Figura 5.58: Distribución de soluciones algoritmo BPA para el Caso 4 para 10 ensayos*

| Estadístico | Valor |
|---|---|
| Mínimo | 55 |
| Efectividad | 73.53% |
| Media | 57.88 |
| Desviación Estándar | 6.6 |
| Varianza | 43.62 |
| Coeficiente de Variación | 0.11 |
| **Cuartiles** | |
| q1 | 55 |
| q2 | 55 |
| q3 | 56 |

**Tabla de Frecuencias**

| Value | Count | Percent |
|---|---|---|
| 55 | 25 | 73.53% |
| 56 | 2 | 5.88% |
| 57 | 1 | 2.94% |
| 59 | 1 | 2.94% |
| 70 | 3 | 8.82% |
| 75 | 1 | 2.94% |
| 80 | 1 | 2.94% |

*Tabla 5.40: Datos estadísticos del algoritmo BPA para el Caso 4*

La distribución de las soluciones encontradas se observa en la siguiente gráfica

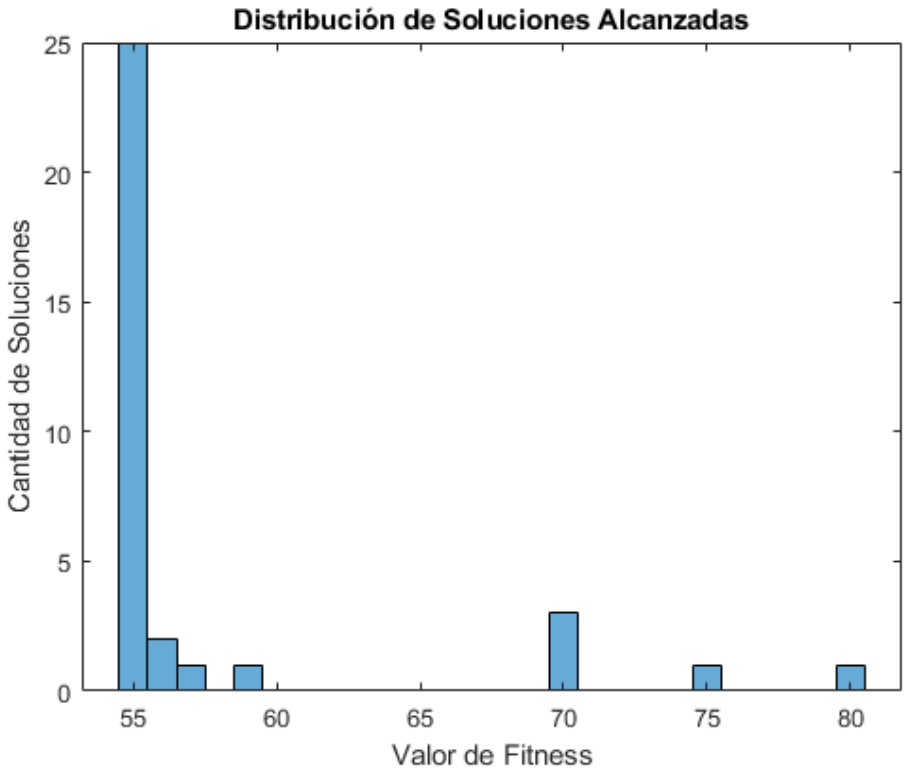


*Figura 5.59: Distribución de soluciones del algoritmo BPA para el Caso 4*

Se expone en las figuras 5.60 y 5.61 la comparativa de cajas y bigotes de los tiempos utilizados para la optimización (BA) y post-optimización (ACO 2) del algoritmo BPA en dos escalas diferentes. Los tiempos del algoritmo BPA se corresponde con la suma de estos tiempos, aunque se observa que el tiempo utilizado para optimización es solamente del orden de 2% del utilizado en la post optimización.

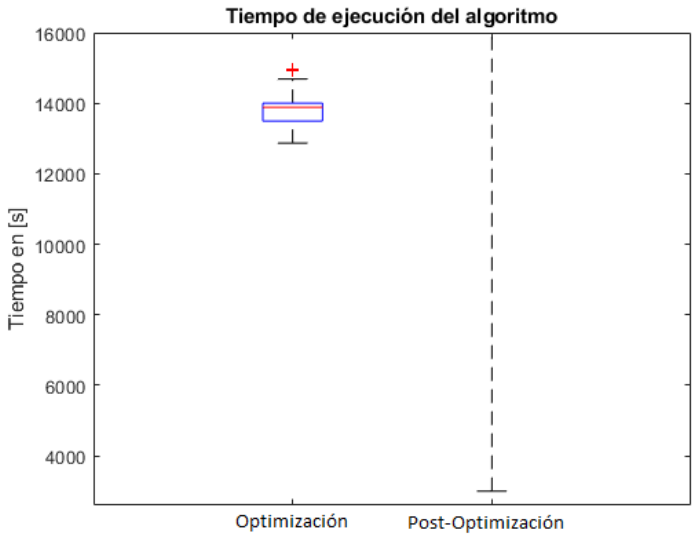


*Figura 5.60: Comparación de las etapas BPA para el Caso 4 escala 0-16000*

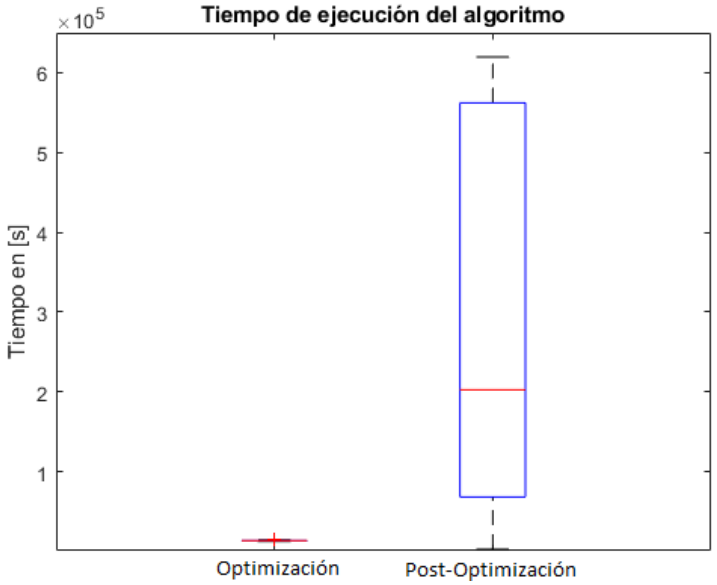


*Figura 5.61: Comparación de las etapas BPA para el Caso 4 escala 0-650000*

### 5.5.3 Comparación del desempeño de los algoritmos para el Caso 4

La comparación de los algoritmos es evidente, debido a que el algoritmo BPA es una mejora por hibridación de BA, e independientemente que se ha tomado solo una muestra del 34% de ensayos para realizar la post-optimización, todas las soluciones de BPA tendrán mejores valores de aptitud que las soluciones BA a excepción de algunas soluciones atípicas.

En el gráfico siguiente se observan los diagramas de caja y bigote de los ensayos de ambos algoritmos donde se refleja claramente esta situación.

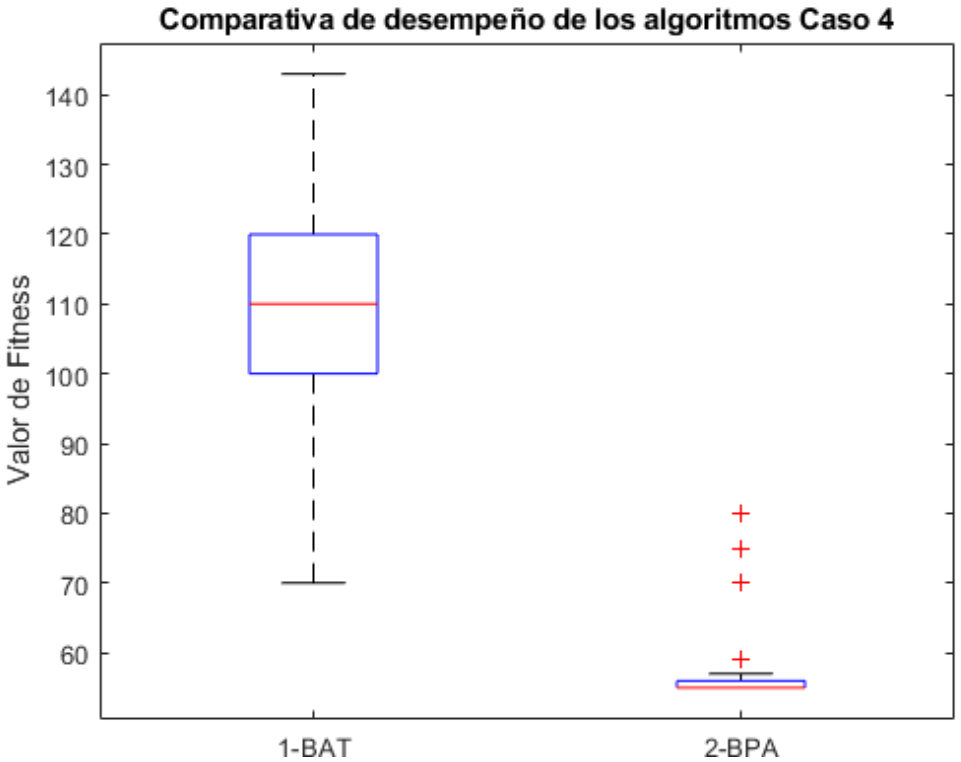


*Figura 5.62: Comparación de resultados de los diferentes algoritmos para el Caso 4*

La comparación de los tiempos que demoran los algoritmos en entregar una solución carece de sentido ya que el post-optimizador se ejecuta, como su nombre lo indica, en forma posterior a BA, por lo que los algoritmos no son comparables desde esa óptica.

Solo podría mencionarse que el algoritmo BA es capaz de entregar soluciones en tiempos medios de 3h 49m y el algoritmo BPA requería del mismo tiempo más el tiempo de post-optimización que en promedio asciende a 89h 42m lo que sumaría un tiempo total de 93h 31m.

Si bien los tiempos no son comparables entre sí, tampoco lo son las soluciones alcanzadas, BPA presenta un 73.53% de efectividad contra una efectividad nula del algoritmo BA, y una media en las soluciones de 57.88 lo que representa prácticamente la mitad de la media de las soluciones entregadas por el algoritmo BA.

## 5.6 Conclusiones del capítulo

Se pudo demostrar experimentalmente que los algoritmos tienen un buen desempeño para pequeñas instancias del problema, y en términos generales todos obtienen buenos resultados en tiempos aceptables.

A medida que crece el problema a resolver, los algoritmos comienzan a necesitar tiempos mayores para encontrar las soluciones deseadas, un comportamiento acorde a lo previsto.

De acuerdo a los ensayos realizados, para instancias medias del problema (red de 6 nodos) el algoritmo ACO 2 de redistribución de flujos presenta los mejores tiempos de ejecución seguido muy de cerca por el algoritmo BA y en tercer lugar PSO con tiempos algo mayores. Por su parte el algoritmo basado en colonia de hormigas que distribuye los flujos denominado algoritmo ACO 1, es el algoritmo que mayor tiempo demanda para encontrar las soluciones óptimas con una amplia diferencia. No obstante, a diferencia de otros algoritmos entrega soluciones con un balance de cargas mayor, lo que se debe a que, en su formulación, la probabilidad de selección de los caminos no está influida solamente por el valor de aptitud de soluciones previas, sino además por parámetros que están en función de la carga de los enlaces de la red.

Para instancias grandes del problema (red de 8 nodos) se descarta el algoritmo ACO 1, PSO comienza a demandar tiempos que crecen considerablemente y las soluciones encontradas son de menor calidad. Los ACO 2 y BA se comportan razonablemente bien, aunque con valores de efectividad bastante menores. Por su parte BPA, si bien necesita mayores tiempos de cómputo, consigue efectividades cercanas al 100%.

Para el caso de problemas muy grandes se ha ensayado el algoritmo BA que, aunque ofrece una eficiencia menor, es capaz de proporcionar, soluciones de buena calidad. Por su parte el algoritmo BPA, a costa de mayores tiempos de cómputo, logra los mejores resultados estadísticos al momento de alcanzar la mejor solución al problema planteado.

En términos generales las estrategias aplicadas a cada algoritmo han respondido satisfactoriamente para distintas instancias, y cada una tiene su ventaja o característica que podrían ser deseables frente a una situación de aplicación real. En todos los casos, las mejores soluciones vienen asociadas a costo de tiempo que se acrecienta a medida que aumenta la instancia del problema a resolver.

Respecto de los tiempos que se reportan en los casos de instancias de mayor tamaño, es importante destacar que los mismos surgen desde una implementación de prototipo, realizada en Matlab, intérprete, no compilado y no paralelizado. Basado en experiencias ajenas a la realización de esta tesis, al re-implementar estos prototipos en otro ámbito, se ha logrado incrementar la velocidad de procesamiento en un factor de 20:1. Esto aún sin hacer uso de la paralelización, una alternativa intrínsecamente utilizable en los diseños descritos.

# Capítulo 6
# Conclusiones

## 6.1 Conclusiones

Los nuevos requerimientos de tráfico de las redes modernas donde el histórico control basado en el mejor esfuerzo no resulta eficiente para evitar la congestión, y la implicancia de adecuar los modelos de tráfico mediante la aplicación de técnicas de optimización que ayuden a la consecución del objetivo de utilizar eficientemente los recursos de la red motivaron el desarrollo de este trabajo.

MPLS fue diseñado para atender las necesidades de las redes modernas y han recibido gran interés en los últimos años debido a las posibilidades, flexibilidad y eficiencia que brinda en ingeniería de tráfico.

Esta tesis se ha centrado, concibiendo a MPLS como una técnica orientada a la conexión, en el problema de ingeniería de tráfico fuera de línea. Luego de entender el funcionamiento, características y las ventajas indiscutibles de MPLS se realizó un modelado matemático de la red que permitió establecer mecanismos para abordar el problema de la asignación de recursos en situaciones de múltiples requerimientos en una red multicaminos.

Determinada la complejidad del problema, y descartando la utilidad de métodos deterministas que pudieran abordarlo, se sintetizó una taxonomía de estrategias heurísticas, se diseñaron e implementaron en un intérprete, cinco algoritmos diferentes, bioinspirados en enjambres, para resolver el problema de la asignación óptima de flujos en redes MPLS.

Se ha logrado obtener soluciones al problema de selección de los mejores LSPs para cada topología y demandas propuestas. Si bien éstas han resultado subóptimas en algunos casos, todas ellas han resultado ser buenas soluciones que cumplen las restricciones de demanda, capacidad de enlaces e, incluso realizando un buen balance de carga, para algunos casos de estudio. En tamaño de instancias pequeños el óptimo global es alcanzado prácticamente en la totalidad de los casos ensayados. Para redes de mayor tamaño se ha conseguido el mismo resultado en la mayoría de los ensayos. Para grandes instancias el porcentaje de obtención del óptimo global es menor, pero casi siempre se encuentra dicho valor en algunos de los ensayos realizados.

En general, las implementaciones, de metaheurísticas basadas en el comportamiento de seres vivos diseñadas para resolver un problema concreto, como en el caso de la presente tesis, necesitan la construcción de estrategias basadas en las metáforas del comportamiento de los organismos vivos, pero necesariamente adaptadas al problema que se resuelve. La hibridación de varios métodos y la inclusión de conocimiento específico del problema se deben combinar de manera tal que coadyuve a la consecución de los requerimientos que la ingeniería exige: obtener buenas soluciones en tiempos aceptables.

De los cinco algoritmos desarrollados, uno está inspirado en bandadas de pájaros (PSO), dos algoritmos están inspirados en el comportamiento de colonias de hormigas (ACO 1 y ACO 2)

uno inspirado en el comportamiento de los murciélagos y el último una hibridación entre el comportamiento de los murciélagos y las hormigas. La diferencia entre ellos recae en la forma en que se desplazan por el espacio de búsqueda, lo que resalta las ventajas de unos u otros dependiendo de las situaciones o instancias a resolver. La combinación entre ellos o su hibridación, resulta en una mejora significativa de los resultados obtenidos.

Experimentalmente se ratificó el funcionamiento de los algoritmos tanto para instancias pequeñas del problema, con excelente desempeño, como para instancias medianas, grandes y muy grandes, en las cuales se pusieron de manifiesto las ventajas de cada implementación. Mientras algunos entregaron soluciones en tiempos menores, otros tuvieron la ventaja de realizar un mejor balance de las cargas entre los caminos posibles.

El diseño de todos los algoritmos mantiene la misma estructura organizativa, parten de la misma representación de la red, la que puede modificarse en forma sencilla para utilizar otras métricas de costo o funciones objetivos diferentes para la optimización, simplemente cambiando la función de cálculo del fitness, por lo que los algoritmos son flexibles y adaptables a otras instancias.

Los algoritmos fueron diseñados y codificados, minimizando las estructuras utilizadas con el fin hacer un uso eficiente del hardware y en consecuencia maximizar su desempeño. No obstante, debido a las características de problema a resolver, fue inevitable utilizar diversas estructuras de información de gran tamaño, para simular el comportamiento de estos seres vivos en busca de una solución en forma algorítmica. Esto ocasiona que los tiempos de ejecución crezcan notablemente a medida que crecen las instancias a resolver, más aun considerando que los algoritmos se constituyen en prototipos que necesitarían recodificarse en lenguajes compilados y paralelizables, que mejorarían drásticamente los tiempos de ejecución para un funcionamiento óptimo para la puesta en producción.

El tema abordado es amplio y existen diversos aspectos que podrían ser analizados con mayor detalle, pero se puede concluir finalmente que, dado el alcance planteado y la complejidad demostrada del problema, los algoritmos desarrollados dan cumplimiento a los objetivos y expectativas delineados en la introducción de esta tesis, resolviendo un problema complejo mediante herramientas desarrolladas a tal fin que permitirán la planificación estratégica fuera de línea para un funcionamiento óptimo de una red MPLS.

Se considera un aporte importante de esta tesis al hecho de que la mayoría de las estrategias heurísticas descriptas en el capítulo 4 podrán adaptarse para la resolución de otros problemas de optimización de asignación de recursos con relativa facilidad. Además, no sólo se ha comprobado las ventajas de la hibridación, sino que esta constituye una potencialidad importante del algoritmo híbrido. Existe un sinfín de hibridaciones posibles entre métodos diversos que pueden mejorar aún más el desempeño de los algoritmos estudiados.

Por otra parte, el modelo matemático de la red puede ser reformulado en forma sencilla no solo utilizando distintas métricas que evalúen su costo, sino focalizando diferentes aspectos a optimizar. Podrían introducirse aspectos tales como funciones de equilibrio de carga específicos, mecanismos de control de congestión, políticas distintas para minimizar el costo de ruteo, distintas políticas de enrutamiento, etc.

Se ha formulado un problema de gran alcance, se han resuelto instancias que han permitido testear un conjunto de estrategias utilizadas en forma pura e hibridas. Se han logrado

encontrar en todos los casos buenas soluciones en tiempos aceptables constituyendo una propuesta válida para el campo de la ingeniería.

## 6.2 Trabajos a Futuro

A raíz de este trabajo pueden plantearse varias instancias de investigación y desarrollo en el futuro.

En primer lugar, se propone la incorporación de otras estrategias para ensayar nuevas hibridaciones, particularmente las relacionadas con la incorporación de búsqueda local como mecanismo de explotación de regiones promisorias del espacio de búsqueda.

Se aplicarán los algoritmos propuestos a la resolución de otro tipo de problemas de asignación de recursos redes, considerando diferentes funciones objetivo, que permitan incluir métricas más complejas y nuevos objetivos, incluso contrapuestos a los planteados en este trabajo dando lugar a un análisis conducente a la resolución de problemas de optimización combinatoria multiobjetivo.

Es posible también abordar el diseño de interfaces que permitan trasladar las soluciones obtenidas a los dispositivos de la red de manera sencilla y/o automática, por ejemplo, generando los archivos de configuración de los equipos de la red en producción. Finalmente, está claro que uso masivo de tecnología no se ha mermado, sino que, por el contrario, día a día surgen nuevos y variados servicios, marcando una firme tendencia a conectar todo a la red de redes. Esto trae aparejado el establecimiento de requerimientos cada vez más exigentes para las disciplinas relacionadas con la conectividad: es necesario, entre otros aspectos, asegurar la flexibilidad necesaria para la implementación de servicios de red de una manera determinista, dinámica y escalable, evitando al administrador de red gestionar dichos servicios a bajo nivel.

La alternativa que ha recibido mucha atención para cumplir con las exigencias citadas es SDN (Software Defined Networking), conjunto de técnicas que se proponen como un cambio disruptivo en el paradigma de las redes de datos y tienden a satisfacer las necesidades mencionadas.

Las técnicas descriptas en esta tesis son aplicables a SDN, La idea original de MPLS era la separación del plano de control del plano de datos, lo que para SDN resulta natural. Además, si bien los sistemas SDN hacen al cliente menos dependiente de un proveedor MPLS, estas tecnologías permiten al proveedor de servicios aumentar y extender las implementaciones MPLS existentes. En efecto, existen avances en configurar flujos de datos entre dispositivos finales con etiquetado MPLS en una arquitectura SDN. Esta combinación de los mecanismos MPLS y la arquitectura SDN permiten la automatización de la red y sus operaciones a través de un control centralizado lo que facilita su optimización y planificación.

El análisis y aplicación de los algoritmos desarrollados en esta tesis podrían fácilmente adaptarse a SDN, más aún cuando es entendible que las soluciones entregadas por los algoritmos diseñados serían configuradas directamente en un solo dispositivo que gobierna la red, el controlador central. Esto plantea un desafío interesante para continuar con la investigación, desarrollo e innovación de técnicas que permitan optimizar el funcionamiento de las arquitecturas de red del futuro, para garantizar métodos rentables y ágiles que aseguren la plena simbiosis con los avances permanentes y sostenidos en TIC a través de la red.

# Bibliografía


Abdulqadder, I.H.; Zou, D.; Aziz, I.T.; Yuan, B. y Li, W. (2018) *"SecSDN-Cloud: De feating Vulnerable Attacks Through Secure Software-Defined Networks"*, IEEE Access,2018,6, pp. 8292-8301.

Alkasassbeh, M.; Al-Naymat, G.; Alauthman, M. y Ednat, E. (2018*) **"**Optimizing Traffic Engineering in Software Defined Networking"*. Preprints 2018, 2018110486 (doi: 10.20944/ preprints 201811. 0486.v1).

Alwayn, V. (2001) *"Advanced MPLS Design and Implementation*". Old Tappan: Cisco Press, 2001.

Amat Rodrigo, J. (2019) "*Optimización con enjambre de partículas (Particle Swarm Optimization)"* by available under a Attribution 4.0 International (CC BY 4.0) at https://www.cienciadedatos.net/documentos/py02_optimizacion_pso

Ambika, B.J. y Banga, M. K. (2021) *"Energy-Efficient MPLS-MANET Using Ant Colony Optimization and Harmony Search Algorithm".* In: Mallick P.K., Bhoi A.K., Marques G., Hugo C. de Albuquerque V. (eds) Cognitive Informatics and Soft Computing. Advances in Intelligent Systems and Computing, vol 1317. Springer, Singapore. https://doi.org/10.1007/978-981-16-1056-1_17

Amorim, K. S. y Pavani, G. S. (2019) *"Routing and Restoration in IP/MPLS over Optical Networks by Means of Ant Colony Optimization"* ***IEEE Global Communications Conference (GLOBECOM),*** 2019, pp. 1-6, doi: 10.1109/GLOBECOM38437.2019.9013479.

Aslam, M. N. y Aziz, Y. (2008) *"Traffic Engineering with Multi-Protocol Label Switching".* Master Thesis. Computer Science Thesis no: MCS-2008-32 August 2008 - Department of Interaction and System Design, School of Engineering at Blekinge Institute of Technology

Awad, M.K.; El-Shafei, M.; Dimitriou, T.; Rafique, Y.; Baidas, M. y Alhusaini, A. (2017) *"Power-efficient routing for SDN with discrete link rates and size-limited flow tables: A tree-based particle swarm optimization approach"*, International Journal of Network Management, 2017,27, (5), pp. e1972.

Awduche D. y Malcolm J. (1999) *"Requirements for trafic engineering over MPLS"* RFC2702, IETF

Barabas, T.; Ionescu, D. y Veres, S.A. (2012) "*Traffic engineering algorithm for differentiated multicast services over MPLS networks"*, in Proc. 7th SACI, Timisoara, Romania.

Belzarena, P. (2003), *"Ingeniería de tráfico en línea en redes MPLS aplicando la teoría de grandes desviaciones".* Tesis Maestría en Ingeniería Eléctrica De La Universidad De La República Montevideo, Uruguay

Burgess, J. (2014) *"ONOS (Open Network Operating System)"*, Ingram Micro Advisor Blog, August 2008. Available at: http://www.ingrammicroadvisor.com/data-center/7-advantages-ofsoftware-defined-networking (Accessed: 5 June 2017) – Tomado de Terefenko 2018.

Casellas, R. (2002) *"MPLS trafic engineering"*, Ph.D. thesis, ENST, Paris.

Casellas, R.; Rougier, J.L.; y Kofman, D. (2002) *"Packet based load sharing schemes in MPLS networks"*, ECUMN'2002. Colmar (April 2002.).

Cassetti, C.; Mardente, G.; Mellia, M.; Manufo, M.; y Lo Cigno, R. (2003) *"Online routing optimization for MPLS-based IP networks"*. In Proceedings of IEEE Workshop on High Performance Switching and Routing, pages 215–220.

Chang, Z.X.; Heng, Z. y Yang, W.J.: (2017) *"Optimization of Resource Management for 5G"*, DEStech Transactions on Computer Science and Engineering, pp. 538- 543.

Chen, H. H. y Huang, S. K.: (2016) *"LDDoS Attack Detection by Using Ant Colony Op timization Algorithms"*, Journal of Information Science & Engineering, 2016,32, (4), pp. 995-1020.

Chen, P,; Huang H. y Dong. X. (2010) *"Iterated Variable Neiborhood descent algorithm for the capacited vehicle routing problem"*. Elsevier. Expert System with Applications, V 37: pp 1620-1627.

ComputerWorld (2003) *"MPLS: Aprendiendo del pasado"* https://www.networkworld.es/archive/ mpls-aprendiendo-del-pasado **(**Accedido 31/07/2020)

Cruz, H. y Viteri, V., (2007) *"Optimización de problemas combinatorios y multiobjetivo utilizando el método de colonia de hormigas (OCH)"*, Master's thesis, Escuela Politécnica Nacional - Quito, Apr. 2007.

Cruz, I.; Carossio, C.; Carnero, M. y Hernández, J.L. (2013) "O*ptimización de ruteo en redes de conmutación de etiquetas multiprotocolo"*. Asociación Argentina de Mecánica Computacional. Mecánica Computacional Vol. XXXII, págs. 2613-2621

Dale, M. J.; Ferra, H. L.; y Palmer, R. A. (2007) *"Fast MPLS network optimization using machine learning".* IEEE Region 10 Annual International Conference Proceedings TENCON.

Dayal, N. y Srivastava, S.: (2018*) "An RBF-PSO based approach for early detection of DDoS attacks in SDN".* In Proc. of IEEE 2018 10th International Conference on Communication Systems & Networks (COMSNETS), Bengaluru, India, Jan. 2018, pp. 17-24.

Delfino, A.; Rivero, S.; San Martín, M. (2006) *"Ingeniería de tráfico en redes MPLS"* – I Congreso Regional de Telecomunicaciones MVD Tecom 2006.

Deneubourg, J. y Pasteels, J., (1983) *"Probabilistic Behaviour in Ants: A Strategy of Errors?",* Journal of Theoretical Biology, vol. 105, pp. 259-271, Oct. 1983.

Depaux, F. (2011) *"Optimización de una Red de Datos IP/MPLS sobre SDH/DWDM usando Tabú Search. Caso de estudio: Red de Datos de un Operador de Telefonía Nacional"* Tesis para obtener el título de Magister en Ingeniería Matemática Instituto de Investigación: LPE-IMERL Componentes universitarios: Universidad de La República, Facultad de Ingeniería

DiCaro, G. D. y Dorigo, M., (1997) *"Antnet: A mobile agents approach to adaptive routing",* Tech. rep., 1997.

DiCaro, G. D. y Dorigo, M., (1998) *"AntNet: Distributed Stigmergetic Control for Communications Networks"* Research, Journal of Artificial Intelligence, pp. 317-365-

Di Stefano, A.; Cammarata, G.; Morana, G. y Zito, D.: (2015) *"A4sdn-adaptive alienated ant algorithm for software-defined networking".* In Proc. of 10th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), Krakow, Poland, Nov.2015, pp. 344-350.

Dobrijevic, O.; Santl, M.; Matijasevic, M. (2015) *"Ant colony optimization for QoE centric flow routing in software-defined networks"*. In Proc. of 11th International Conference on Network and Service Management (CNSM), Barcelona, Spain, Nov. 2015, pp. 274-278.

Domínguez Medina, C. H. (2011) *"Algoritmos bioinspirados para el encaminamiento de datos en redes inalámbricas de sensores"*. Tesis Maestría en Ciencias de la Computación - Instituto Politécnico Nacional. Centro de Investigación en Computación - México D.F., 26 de mayo del 2011

Donoso, Y.; Fabregat, R. y Marzo, J. (2004). *"Multi-Onjective Optimization Algorithm for multicast Routing with Traffic Engineering"* IEEE 3rd International Conference on Networking ICN"04 - Guadeloupe, French Caribbean.

Dorigo, M. (1992). *"Optimization, Learning and Natural Algorithms"*. Politecnico di Milano, Italy. PhD thesis.

Dorigo, M. y DiCaro, G., (1999) *"Ant Colony Optimization: A New Meta-Heuristic".* - In: Proceedings of the 1999 congress on Evolutionary computation-CEC99 (Cat. No. 99TH8406), vol. 2, pp. 1470–1477. IEEE.

Dorigo, M.; Birattari, M.; y Stutzle, T., (2006) *"Ant Colony Optimization Articial Ants as a Computational Intelligence Technique"*, IEEE Computational Intelligence Magazine vol. 1,pp. 28-39, Sept..

Egilmez, H. E.y Dane, S. T. (2012) *"OpenQoS: An OpenFlow controller design for 103 multimedia deliverywith end-to-end Quality of Service over Software-Defined Networks"*. Signal & Information Processing Association Annual Summit and Conference (APSIPA ASC), IEEE,1–8. Retrieved from http://ieeexplore.ieee.org/xpls/abs_all.jsp?a rnumber=6411795

El-Alfy, E. M. (2006) *"MPLS Network Topology Design Using Genetic Algorithms".* IEEE International Conference on Computer Systems and Applications, pp 1059-1065. doi: 10.1109/AICCSA.2006.205218.

El-Sayed, M. El-Alfy,; Mujahid, S.N. y Selim S.Z., (2013) "*A Pareto-based hybrid multiobjective evolutionary approach for constrained multipath traffic engineering optimization in MPLS/GMPLS networks".* Journal of Network and Computer Applications 36 pp 1196-1207

Elwalid, A.; Jin, C.; Low, S. y Widjaja I. (2001) *"MATE: MPLS adaptive traffic engineering,"* Proceedings IEEE INFOCOM 2001. Conference on Computer Communications. Twentieth Annual Joint Conference of the IEEE Computer and Communications Society (Cat. No.01CH37213), 2001, pp. 1300-1309 vol.3, doi: 10.1109/INFCOM.2001.916625.

Felici, S. y Montañana, R. (2009) *"PRÁCTICA: Configuración de MPLS en modo trama".* Licencia Creative Commons Atribución-NoComercial-CompartirIgual 4.0 Internacional.

Fister, Jr.; Yang, X. S.; Fister, I.; Brest, J. y Fister, D., (2013) *"A brief review of nature-inspired algorithms for optimization".* Elektroteh. Vestnik/Electrotechnical Rev., vol. 80, no. 3, pp. 116–122, Jul. 2013

Fuentes, J. y Torruella G. C. (2011) *"MPLS MultiProtocol Label Switching".* Redes y Comunicaciones Ingeniería y Arquitectura La Salle. Universitat Ramon Llull

Gagnon, I.; April, A.; y Abran, A. (2020). *"A critical analysis of the bat algorithm"*. Engineering Reports. 2. 10.1002/eng2.12212.

Gao, C.; Wang, H.; Zhai, L.; Yi, S. y Yao, X.: (2016) *"Optimizing Routing Rules Space through Traffic Engineering Based on Ant Colony Algorithm in Software Defined Network"*. In Proc. of IEEE 28th International Conference on Tools with Artificial Intelligence (ICTAI), San Jose, USA, Nov.2016, pp. 106-112.

Gero, J. (1987) "*Artificial intelligence and engineering optimization. Engineering Optimization",* vol. 12, pp. 89-90.

Gerometa, O. A. (2012) *"MPLS Frame-Mode Básico".* Cuadernos EduBooks® ISBN:978-987-27966-1-7

Glesk, I.; Fouad, M. y Masood, M. (2018) *"Pareto Based Bat Algorithm for Multi Objectives Multiple Constraints Optimization in GMPLS Networks".* Collection of open chapters of books in transport research. URL https://www.scipedia.com/public/Glesk_et_al_2018a Vol. 2018, 93

Glover, F. (1989). *"Tabú Search – Part I".* ORSA Journal on Computing – Vol 1: pp 190-206.

Gonzales Cos, L. A. (2012) *"Optimización de enrutamiento en redes IP usando algoritmo Antnet calibrado por un algoritmo genético"* Tesis Magister en ingeniería informática - Pontificia Universidad Católica de Valparaíso Facultad de Ingeniería Escuela de Ingeniería Informática (http://opac.pucv.cl/pucv_txt/txt-3000/UCF3404_01.pdf)

Grampin, E.; Castro, A.; German, M,; Rodriguez, F.; Tejera, G. y Sanguinetti M. (2007). *"A PCE-based Connectivity Provisioning Management Framework".* Proceedings of IFIP/IEEE 5th Latin American Network Operations and Management Symposium.

Guo, A. y Yuan, C. (2021) *"Network Intelligent Control and Traffic Optimization Based on SDN and Artificial Intelligence".* Electronics 2021, 10, 700. https://doi.org/10.3390/ electronics 10060700

Hakiri, A.; Gokhale, A.; Berthou, P.; Schmidt, D. C. y Gayraud, T. (2014) *"Software Defined Networking: Challenges and research opportunities for Future Internet".* Computer Networks, 75, 453–471.http://doi.org/10.1016/j.comnet.2014.10.01 5

Hernández Camacho, T. (2015) "*Estudio de la ingeniería de tráfico en redes MPLS mediante casos de uso práctico con la herramienta VNX"* - Trabajo Final Máster Universitario en Ingeniería de Redes y Servicios Telemáticos - Universidad Politécnica de Madrid Escuela Técnica Superior de Ingenieros de Telecomunicación

Holland, J.H. (1975) *"Adaptation in Natural and Artificial Systems".* University of Michigan Press, Ann Arbor, Michigan; re-issued by MIT Press (1992).

Hopkin, M. (2004) *"Las hormigas tienen un sistema propio para regular el tráfico",* Nature News, vol. 10, pp. 1-6, Mar. 2004.

Hu, F. (ed.) (2014) *"Network innovation through Openflow and SDN: Principles and design".* Boca Raton, FL: Taylor & Francis

Iglesias de la Torre, D.; Álvarez Paliza, F.A.; Ramos Fleites, A (2019) "*Combinación de mecanismos MPLS en una arquitectura SDN"*¸ Revista Telemática. Vol. 18. No. 1, enero-abril, 2019, p.1-10 ISSN 1729-3804

Kennedy, J. y Eberhart, R.C. (1995). *"Particle swarm optimization"*. In Proceedings of the 1995 IEEE International Conference on Neural Networks, Perth, Australia, 1942–1948. Piscataway, NJ: IEEE Service Center.

Kirkpatrick, S.; Gelatt, C. D. y Vecchi, M. P. (1983). *"Optimization by Simulated Annealing".* Science, New Series 220 (4598): pp 671–680.

Know How - Digital Guide Ionos (2017) *"MPLS: estándar de transporte de datos en redes"* https://www.ionos.es/digitalguide/servidores/know-how/mpls-que-es-el-multiprotocol-label-switching/ accedido 31/07/2020

Krasnogor, N. y Smith, J. (2005) *"A tutorial for competent memetic algorithms: model, taxonomy, and design issues"*. IEEE Transactions on Evolutionary Computation, 9(5): 474–488.

Kuipers, F.A.; Korkmaz, T.; y Krunz, M. (2002) *"An overview of constraint-based path selection algorithms for QoS routing".* IEEE Communications Magazine, vol. 40, no. 12 (December 2002).

Latah, M. y Toker, L. (2018) *"Artificial Intelligence Enabled Software Defined Networking: A Comprehensive Overview".* Department of Computer Engineering, Ege University, 35100, Bornova, Izmir, Turkey arXiv:1803.06818v3 [cs.AI] 6 Nov 2018

Lemeshko, O.V.; Hailan, A.M. y Starkova, O.V. (2011) *"Multi-level traffic management in the MPLS-TE DiffServ network"*, CAD Systems in Microelectronics (CADSM), 11th International Conference The Experience of Designing and Application of, pp.118-120.

Li, K.; Bao, J.; Lu, Z.; Qi, Q.; Wang, J.: (2017) *"A PSO-based virtual SDN customiza tion for multi-tenant cloud services".* In Proceedings of the 11th International Conference on Ubiquitous Information Management and Communication, Beppu, Japan, Jan. 2017, pp. 1-6.

Lin, R. y Ye, Z. (2018) "*A bat algorithm for SDN network scheduling". EURASIP J.Wirel. Commun. Netw*., https://doi.org/10.1186/s13638-018-1145-y

Liu, J.; Zhang, H. y Guo, Z.: (2017) *"A Defense Mechanism of Random Routing Mutation in SDN".* IEICE Transactions on Information and Systems, 2017,100, (5), pp. 1046-1054.

Minei, I. y Lucek, J. (2011) *"MPLS-Enabled Applications. Emerging Developments and New Technologies"*. Third Edition. John Wiley & Sons, Ltd. ePDF ISBN: 9780470976166

Machesa, M.G.; Tartibu, L.K.; Tekweme, F.K. y Okwu, M.O. (2019*). "Evaluation of the Stirling heat engine performance prediction using ANN-PSO and ANFIS models".* In 2019 6th International Conference on SoftComputing and Machine Intelligence (ISCMI), 19Nov 2019, 217–222. IEEE. https://doi.org/10.1109/ISCMI47871.2019.9004406.

Martinez, J. F. (2018) *"SD-WAN vs. MPLS"* Gerente Comercial en Ingeniería Smart S.A. Publicada el 20 de abril de 2018 en https://www.linkedin.com/pulse/sd-wan-vs-mpls-juan-facundo-martinez/?trk=related_artice_SD-WAN%20vs.%20MPLS_article-card_title (Consultado 18/01/2021)

Masood, M.; Fouad, M.; Kamal, R.; Glesk, I. y Khan, I. U. (2019) *"An Improved Particle Swarm Algorithm for Multi-Objectives Based Optimization in MPLS/GMPLS Networks"* in IEEE Access, vol. 7, pp. 137147-137162, 2019, doi: 10.1109/ACCESS.2019.2934946.

Masood, M.; Fouad, M. y Glesk, I., (2017) "*A Pareto based approach with elitist learning strategy for MPLS/ GMPS networks",* in Proc. 9th CEEC, Colchester, UK,

Masood, M.; Fouad, M.; y Glesk, I. (2017). *"Proposing bat inspired heuristic algorithm for the optimization of GMPLS networks".* 1-4. 10.1109/TELFOR.2017.8249295.

Masood, M.; Fouad, M. y Glesk, I., (2018) "*Analysis of Artificial Intelligence-Based Metaheuristic Algorithm for MPLS Network Optimization",* 20th International Conference on Transparent Optical Networks (ICTON), pp. 1-4, doi: 10.1109/ICTON.2018.8473751.

Melendi, D.; Pañeda, X. G.; García, V. G.; García, R. y Neira, A. (2003) *"Métricas para el Análisis de Calidad en Servicios de Vídeo-Bajo-Demanda Reales"* Tercer congreso iberoamericano de telemática Art. 19. Uruguay 2003

Mira Telecomunicaciones (2019) *"MPLS – Multiprotocol Label Switching"*. Publicación web de divulgación. https://miratelecomunicacions.com/blog/sin-categorizar/mpls-multiprotocol-label-switching/ (accedido 31/07/2020)

Montemanni, R. y Smith, D. H.; (2009) *"Heuristic manipulation, tabu search and frequency assignment"*. Computers & Operations Research.

Naganathan, E. R.; Rajagopalan, S. y Narayanan, S. (2014) *"Protocol Label Switching Network Using Ant Colony Optimization Algorithm"*. ICT and Critical Infrastructure: Proceedings of the 48th - Springer International Publishing Book Series: Advances in Intelligent Systems and Computing Print ISBN: 978-3-319-03106-4 Electronic ISBN: 978-3-319-03107-1

Okwu, M. y Tartibu L. k., (2021) "*Metaheuristic Optimization: Nature-Inspired Algorithms Swarm and Computational Intelligence, Theory and Applications"*. Studies in Computational Intelligence 927, Springer, ISBN 978-3-030-61110-1, pp. 1-151

Onety, R.E.; Tadei, R.; Neto, O.M. y Takahashi, R.H.C., (2013) "*Multiobjective optimization of MPLS-IP networks with a variable neighborhood genetic algorithm".* Applied Soft Computing 13 pp. 4403-4412

O'Reilly, J. (2014) *"SDN Limitations".* Network Computing, October 2014. Available at: https://www.networkcomputing.com/networking/sdn-limitations/241820465 (Accessed: 5 June 2017) – Tomado de Terefenko 2018.

Osman, I.H. y Kelly, J.P. (1996) "*MetaHeuristics: Theory and Applications*", Boston USA Ed. Kluwer Academic.

Parsaei, M.R.; Mohammadi, R. y Javidan, R. (2017) *"A new adaptive traffic engineer ing method for telesurgery using ACO algorithm over Software Defined Networks",* European Research in Telemedicine/La Recherche Europeenne en Telemedecine, 2017,6, (3-4), pp. 173:180.

Poli, R. (2007) *"An analysis of publications on particle swarm optimisation applications".* Technical Report CSM-469, Department of Computer Science, University of Essex, UK.

Poli, R. (2008) *"Analysis of the publications on the applications of particle swarm optimisation".* Journal of Artificial Evolution and Applications, pages 1–10.

PremKumar, S. y Saminadan, V. (2017) *"Performance evaluation of smart grid communication network using MPLS".* Int. Conf. on Communication and Signal Processing (ICCSP), Chennai, India, 2017, pp. 2116–2120

Rajagopalan, S.; Naganathan, E.R. y Raj, P.H. (2011) *"Ant Colony Optimization Based Congestion Control Algorithm for MPLS Network".* In: Mantri A., Nandi S., Kumar G., Kumar S. (eds) High Performance Architecture and Grid Computing. HPAGC 2011. Communications in Computer and Information Science, vol 169. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-22577-2_29

Rajpurohit, J.; Sharma, T. K. y Abraham, A.; (2017) *"Glossary of Metaheuristic Algorithms".* Int. J. Comput. Inf. Syst. Ind. Manag. Appl., vol. 9,pp. 181–205.

Reber, A. (2015) *"On the Scalability of the Controller in Software-Defined Networking",* MSc in Computer Science, University of Liege, Belgium. Available at: http://www.student.montefiore.ulg.ac.be/~agirmanr/src/tfe-sdn.pdf (Accessed: 5 June 2017). – Tomado de Terefenko 2018.

Reeves, C.R. (2014) *"Fitness Landscapes".* In: Burke E., Kendall G. *"Search Methodologies",* pp 681-705. Springer, Boston, MA. https://doi.org/10.1007/978-1-4614-6940-7_22

Reis, J.; Rocha, M.; Phan, T. K.; Griffin, D.; Le, F. y Rio, M., (2019) "*Deep Neural Networks for Network Routing",* International Joint Conference on Neural Networks (IJCNN), 2019, pp. 1-8, doi: 10.1109/IJCNN.2019.8851733.

Ridwan, M. A.; Radzi, N. A.; Wan S. H. M.; Abdullah, F.; Jamaludin, M.Z. y Zakaria, M. N. (2019). *"Recent trends in MPLS Networks: Technologies, Applications and Challenges".* IET Communications. 14. 10.1049/iet-com.2018.6129.

Rodríguez Herlein, D. R.; Talay, C. A.; González, C. N. y Marrone, L. (2020) *"Explorando las redes definidas por software (SDN)"* XXII Workshop de Investigadores en Ciencias de la Computación (WICC 2020, El Calafate, Santa Cruz). Red de Universidades con Carreras en Informática. XXII Workshop de Investigadores en Ciencias de la Computación (WICC 2020, El Calafate, Santa Cruz). ISBN: 978-987-3714-82-5 Páginas: 100-104

Rojas Calero, A. M. y Pachón Á. (2017) *"Software Defined Networks"* Dirección de Tecnología. Institución Universitaria Antonio José (Consultado 20/01/2021) disponible en: https://silo.tips/download/software-defined-networks-15

Rosen, E.; Viswanathan, A.; y Callon, R. (2001) *"Multiprotocol Label Switching Architecture",* RFC 3031, DOI 10.17487/RFC3031, January 2001, https://www.rfc-editor.org/info/rfc3031.

Shuguftha Naveed, S. y Vinay Kumar, (2014) *"MPLS Traffic Engineering - Fast Reroute",* International Journal of Science and Research (IJSR), https://www.ijsr.net/get_abstract.php?paper_id=20132188, Volume 3 Issue 5, May 2014, 1796 -1801

Sarsembagieva, K.; Gardikis, G.; Xilouris, G. y Kourtis, A. (2012) *"A fast route planning algorithm for MPLS-TE",* Telecommunications and Multimedia (TEMU), 2012 International Conference on , pp.142-146.

Sathya, R. y Thangarajan, R. (2015) *"Efficient anomaly detection and mitigation in soft ware defined networking environment".* In Proc. of 2nd International Conference on Electronics and Communication Systems (ICECS), Coimbatore, India, Feb. 2015, pp. 479-484.

Sylwester, K. y Adam, K. (2008) *"Evolutionary algorithms in MPLS network designing",* Information Technology. IT 2008. 1st International Conference on, vol.1, no.4, pp. 18-21.

Tetzlaff, T., Gaudiani, A; Rojas Paredes, A.; Encinas, E.; Fassio, E.; Trigila, M.; González, R. y Bertaccini, D., (2021) *"Metaheurísticas, búsqueda estocástica y cómputo eficiente en optimización aplicada".* XXIII Edición del Workshop de Investigadores en Ciencias de la Computación - Chilecito: UNdeC, 2021. ISBN 978-987-24611-3-3 pp 124-128

Terefenko, D. (2018). *"A Comparison of Multiprotocol Label Switching (MPLS) and OpenFlow Communication Protocols"*. 10.13140/RG.2.2.14457.98404.

Torres-Treviño, L.M. (2021). *"A 2020 taxonomy of algorithms inspired on living beings behavior". ArXiv, abs/2106.04775***.** Cornell University

Umar, S. y Rashid, T. A. (2021) "*Critical analysis: Bat algorithm-based investigation and application on several domains".* CoRR, abs/2102.01201, 2021.

Walshaw, C. (2004). *"Multilevel refinement for combinatorial optimization problems".* Annals of Operations Research, 131: pp 325–372.

Walsh, T. y Cherukuri, R. (2005) *"Two Reference Models for MPLS Control Plane Iinterworking",* MPLS/ FR Alliance Technical Committee document MPLS 2005.050.00, March 2005

Wang, C.; Zhang, G.; Xu, H. y Chen, H. (2016) *"An ACO-based Link Load-Balancing Algorithm in SDN"*. In: Proc. of 7th International Conference on Cloud Computing and Big Data (CCBD), Macau, China, nov., pp. 214-218.

Wang, D.; Tan, D. y Liu, L. (2017) *"Particle swarm optimization algorithm: an overview"* Springer-Verlag Berlin Heidelberg 2017 Soft Comput DOI 10.1007/s00500-016-2474-6

Wang, F.; Liu, B. y Zhang, L., (2017) *"Dynamic routing and spectrum assignment based on multilayer virtual topology and ant colony optimization in elastic software defined optical networks*", Optical Engineering, ,56, (7), pp. 076111.

Wang, Z. y Crowcroft, J. (1996) *"Quality of service routing for supporting multimedia applications",* IEEE JSAC (Sept. 1996).

Xiong, Y.; Shi, J.; Lv, Y. y Rouskas, G.N. (2017) *"Power-Aware Lightpath Management for SDN-Based Elastic Optical Networks"*. In Proc. of 26th International Conference on Computer Communication and Networks (ICCCN), Vancouver, Canada, July 2017, pp. 1-9.

Yang, X. S., (2010). *"A new metaheuristic bat-inspired algorithm".* In Nature Inspired Cooperative Strategies for Optimization (NICSO 2010) (pp. 65–74). Springer, Berlin, Heidelberg

Yang, X. S., ( 2018) *"Nature-Inspired Algorithms and Applied Optimization"*, vol. 744.London: Springer, Cham.

Yang, X. S. (2010). *"A new metaheuristic bat-inspired algorithm".* In Nature Inspired Cooperative Strategies for Optimization (NICSO 2010), 65–74. Berlin, Heidelberg: Springer.

Yang, X. S. (2011). *"Bat algorithm for multi-objective optimisation".* International Journal of Bio-Inspired Computation 3 (5): 267–274.

Yi, S.; Wang, H.; Yao, X.; Gao, C. y Zhai, L. (2016) *"Maximizing Network Utilization for SDN Based on WiseAnt Colony Optimization"*. In Proc. of IEEE 18th International Conference on High Performance Computing and Communications; IEEE 14th International Conference on Smart City; IEEE 2nd International Conference on Data Science and Systems (HPCC/SmartCity/DSS), Dec. 2016, Sydney, Australia, pp. 959- 966.

Yousif, S. T. (2021) *"Congestion Management in MPLS Network Based on Hybrid Particle Swarm Algorithm"* Journal of Al-Qadisiyah for Computer Science and Mathematics Vol. 13(1) 2021, pp Comp. 109–115 DOI: https://doi.org/10.29304/jqcm.2021.13.1.786