Estudio comparativo de heurísticas bioinspiradas para optimización de redes MPLS

Javier Alejandro Carletto[1], José Luís Hernández[2] y Francisco Javier Díaz[3]

[1] Facultad de Ingeniería y Ciencias Agropecuarias – Universidad Nacional de San Luis
[2] Facultad de Ingeniería – Universidad Nacional de Rio Cuarto
[3] Facultad de Informática – Universidad Nacional de La Plata
javiercarletto@gmail.com

Resumen: La exigencia impuesta por el crecimiento de tráficos tan disímiles como voz, video, sonido, datos, etc. presentes en las redes convergentes actuales, ha conducido a la implementación de nuevas tecnologías para garantizar los anchos de banda requeridos. MPLS (conmutación de etiquetas multiprotocolo) se ha convertido en una tecnología eficaz cuando las demandas modernas ponen en riesgo de congestión a las redes que utilizan las técnicas tradicionales de conmutación, aún a cuando el problema de la selección de la mejor ruta y de la distribución óptima del tráfico siga existiendo, y exija nuevas propuestas de optimización del enrutamiento. La planificación conlleva a resolver un problema de optimización combinatorio cuyas características hacen inviable la utilización de métodos determinísticos, por lo que surgen otras alternativas como las heurísticas. Se presenta un análisis de estrategias bioinspiradas 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 diseñan cinco algoritmos inspirados en enjambres que permiten determinar una solución óptima explorando el espacio de búsqueda desde diferentes estrategias que brindan una solución de configuración fuera de línea, a este problema tradicional de la ingeniería de tráfico en redes con alta interconectividad. Se determina la aplicabilidad y los parámetros óptimos para distintas instancias, y se comparan los resultados.

Palabras Claves: MPLS, optimización, algoritmos bio-inspirados.

1 Introducción

Tradicionalmente los algoritmos de enrutamiento se han basado en minimizar el costo del ruteo, sin considerar el tráfico en sí, es decir sin optimizar el rendimiento de la red. Ejemplos de estos algoritmos son los utilizados en, RIP (Routing Información Protocol), basado en el método denominado vector de distancias y OSPF (Open Shortest Path First), representativo del enrutamiento por estado de enlaces. 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 otras rutas pueden permanecer sin uso [1]. Si bien estos algoritmos

tradicionales han funcionado bien 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 [2].

En redes dinámicas, el algoritmo de enrutamiento debe manejar un conjunto de funcionalidades básicas para manejar la congestión, control, encolamiento y tráfico generado por los usuarios. Su tarea principal 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. [1]

La evolución de las redes de comunicaciones permite que la infraestructura sea capaz de transmitir flujo de información multiservicio sobre una misma plataforma. Debido a las características específicas de cada tipo de tráfico, la red debe tratar a cada uno de ellos de manera diferencial para garantizar una calidad de servicio demandada [3]. El objetivo de la Ingeniería de Tráfico (Traffic Engineering, IE) es adaptar los flujos de tráfico a los recursos físicos, equilibrando su utilización [4].

Uno de los mecanismos de transporte de datos estándar, creado por la IETF y definido en el RFC 3031, es la conmutación de etiquetas multiprotocolo o MPLS, por sus siglas en inglés, MultiProtocol Label Switching [5]. En la transmisión, se genera una ruta entre los enrutadores de borde 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 [6]. Esta tunelización es una herramienta poderosa porque solo los enrutadores de entrada y salida necesitan conocer el contenido del tráfico transportado a través del túnel, mientras que, en el núcleo, el tráfico se puede enrutar explícitamente siguiendo las políticas de tráfico especificadas [7].

MPLS es escalable, está orientado a la conexión, es independiente de la tecnología de transporte de reenvío de paquetes, reduce la búsqueda de direcciones IP en las tablas de ruteo y minimiza la latencia [9]. La red MPLS puede decidir la mejor ruta para cada flujo, asignar múltiples servicios y tratar cada tráfico según los requisitos de QoS [7].

Las tareas de la ingeniería de tráfico consisten en seleccionar LSPs adecuados para un correcto equilibrio de carga en el sistema, disminuyendo la congestión y maximizando la utilización de los enlaces.

La distribución de la demanda puede ser controlada de forma tal que optimice la utilización y el desempeño a la vez que se satisfacen requerimientos de calidad de servicio y restricciones administrativas o de recursos [3]. 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, para determinados tamaños de instancias, buenas soluciones en tiempos razonables debido a su complejidad.

Uno de los principales tópicos de investigación y desarrollo en el área de Ingeniería de Trafico en MPLS es el ruteo basado en restricciones y 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. Se ha catalogado a CBR como un problema NP-completo y se han propuesto múltiples algoritmos heurísticos para abordarlo [10].

Las herramientas de optimización basadas en aplicaciones de inteligencia artificial han obtenido un reconocimiento importante en la resolución de problemas de optimización. Estas técnicas 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 ([11]; [8]). Estos métodos son agrupados dentro de las estrategias heurísticas y metaheurísticas.

En este trabajo se propone resolver el problema de asignación de recursos frente a situaciones de múltiples requerimientos en una red MPLS, para lo cual se analizan estrategias heurísticas y metaheurísticas bioinspiradas, donde el objetivo es minimizar el costo de enrutamiento, cumpliendo con restricciones de capacidad de los enlaces y de demandas específicas, para una planificación de la red fuera de línea.

2 Definición del problema

El modelado de la red fue presentado en [6], y los aspectos más importantes de describen a continuación.

La red MPLS, es 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.

Cada requerimiento sobre la red, es una solicitud de ancho de banda asegurado entre dos nodos, y será especificado mediante una terna $t=(S_t, D_t, d_t)$ donde S_t es el nodo origen, D_t el nodo destino y d_t la demanda de flujo requerida, en unidades/seg, que se desea transportar entre esos nodos. El conjunto de los caminos entre S_t y D_t se denomina P_t.

El costo total de un camino p para una solicitud t se puede calcular como la sumatoria de los costos de los enlaces utilizados, y debe respetarse que la sumatoria de las porciones de tráfico enrutadas por distintos caminos debe igualar al flujo demandado.

El costo total del sistema está dado por la sumatoria de los costos de los caminos multiplicados por la porción de flujo enrutados por ellos, por lo que la formulación matemática del problema se puede representar por:

$$min\left\{\sum_{t\in T}\sum_{l=1}^{L_t} costo_{p,t} \,.\, x_{p,t}\right\} \tag{1}$$

s.a.

$$\sum_{p=1}^{L_t} x_{p,t} = d_t \quad \forall t \in T \tag{2}$$

$$\sum_{t\in T}\sum_{l=1}^{L_t} x_{p,t} \quad a_{p,t,e} < Cap_e \tag{3}$$

La ecuación 2 exige el cumplimiento de la demanda mientras la ecuación 3 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 transporte.

El problema planteado es encontrar la distribución de tráfico que minimice el costo de transporte, para un conjunto de requerimientos determinados sobre una topología de red. 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 [12].

2.1 Representación de una solución

Una solución estará representada por un conjunto de vectores: uno por cada uno de los requerimientos de tráfico impuestos sobre la red. 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 $L(t_i), \forall i=1:r$.

Para ilustrar la representación supóngase un flujo de datos requerido de 10 *u/seg* entre el nodo 1 y 2 en la infraestructura de la Fig. 1. El flujo requerido puede ser dividido entre los caminos, representados por las flechas, de distintas maneras. Por ejemplo: [3, 2, 2], [6, 4, 0], [10, 0, 1].
Frente a diversas solicitudes sobre la red, la solución final puede representarse como:

$$x = [(x_{p1,t1}, x_{p2,t1}, \dots x_{pL_{t1},t1}), (x_{p1,t2}, x_{p2,t2}, \dots x_{pL_{t2},t2}) \dots (x_{p1,tr}, x_{p2,tr}, \dots x_{pL_{tr},tr})] \tag{4}$$

De esta manera, *x* tendrá tantos vectores como requerimientos se tengan, y cada vector tendrá tantas componentes como caminos alternativos existan para cada requerimiento. No cualquier combinación de las componentes del vector *x* 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.1).

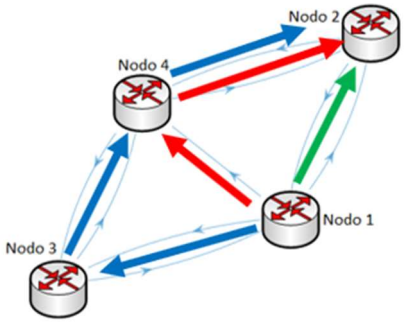

Fig. 1. Caminos posibles para un requerimiento de tráfico entre Nodo 1 y Nodo 2.

Considérese el siguiente esquema de requerimientos sobre la red de la Fig. 1: t_1=(1, 2, 10), t_2=(2, 3, 10) y t_3=(1, 3, 20). Para este caso el vector solución es un vector de tres subvectores con tres componentes cada uno, cuya cantidad de variaciones a evaluar es 999810 posibilidades. Si en este ejemplo se incrementa el requerimiento t_1 a 20 *u/seg* la cantidad de variaciones posibles a evaluar asciende a 3042900, lo que triplica al espacio de búsqueda original.

3 Estrategias Algorítmicas

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 a través de movimientos con alguna componente aleatoria. La diferencia entre las metaheurísticas está en la forma en que proponen el siguiente movimiento, dentro del espacio de soluciones [13]

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 [6].

Según [14] 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, generalmente, comparten información entre múltiples agentes, por lo que la autoorganización, la coevolución y el aprendizaje durante las iteraciones pueden ayudar a proporcionar una alta eficiencia. 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.

En este trabajo, para resolver el problema planteado, se han desarrollado cinco algoritmos inspirados en inteligencia de enjambre, basados en el conocimiento del comportamiento biológico de pájaros, insectos y murciélagos, cuyas estrategias se detallan a continuación.

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

La optimización por enjambre o cúmulo de partículas (Particle Swarm Optimization PSO) es un método de optimización estocástica desarrollado por Kennedy y Eberhart (1995) [15].

La abstracción del mecanismo descripto tras la observación del comportamiento de los individuos de ciertas especies que demuestran una inteligencia social y colectiva para conseguir sus objetivos inspiró un sinfín de variaciones al algoritmo, pero lo interesante del modelo clásico planteado es que cada partícula se mueve con ecuaciones

muy básicas y de bajo costo computacional por lo que la complejidad del proceso recae en la evaluación de la aptitud o fitness de cada partícula en cada iteración.

Para la optimización del problema planteado el vector solución **x** se encuentra representado por la posición de la partícula. Es decir, la posición está representada por un vector n dimensional. La actualización de la posición en cada iteración viene determinada por las ecuaciones 5 y 6

$$x_i^t = x_i^{t-1} + v_i^t \tag{5}$$

$$v_i^t = w * v_i^{t-1} + c_1 * r_1^t * \left(p_i^{best} - x_i(t-1)\right) + c_2 * r_2^t * \left(G_{Best} - x_i(t-1)\right) \tag{6}$$

Donde: v_i^t es la velocidad de la partícula *i* en el instante *t*, *w* es el factor de aceleración, 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* y G_{Best} representa la mejor posición conocida por el enjambre.

Dada la naturaleza no continua del espacio de búsqueda, es necesario realizar una reparación de la solución, ya que los elementos del vector solución no pueden ser negativos y se considerarán solamente valores enteros.

Finalmente, el cálculo del fitness de cada solución incluye una función de penalización que altera el valor de aptitud de las soluciones que no cumplan con la restricción de la capacidad de los enlaces.

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

Durante el proceso de exploración del territorio, 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, depositan una sustancia química denominada feromona (una sustancia que puede "olerse") [16]. 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, lo que permitirá encontrar el camino más corto hacia la fuente de alimento.

En la formulación matemática, la técnica básicamente consiste en la toma de decisiones mediante una ecuación de probabilidad que toma en cuenta tanto el valor de feromona depositado como un término heurístico que se define en función del problema. Inspirados en ACO, se desarrollan dos algoritmos, el primero de ellos construye una solución a través de la asignación de flujos en los caminos posibles, y el segundo parte de una solución aleatoria y la optimiza redistribuyendo los flujos generados aleatoriamente.

Algoritmo ACO 1 – Distribución de flujos basado en ACO. Dado un conjunto de requerimientos T, cada "hormiga" tomará al azar un requerimiento por vez para hacer

la asignación. 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 del rastro de feromona. Es decir que el *tour* de la hormiga se constituirá en la solución buscada.

Para este caso, la información heurística de cada camino η(p) se representa con 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:

$$\eta(\mathrm{p}) = \frac{CapU_p(\mathrm{p})}{Costo(p)} \tag{7}$$

Siendo $CapU_p(p)$ la capacidad útil del camino p, $\mathrm{Costo(p)}$ costo del camino p.

La utilización de la información heurística de cada camino η(p) 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.

La feromona τ, se representará 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 τ_0 que constituye un parámetro del algoritmo y su actualización se realizará sumando al valor de feromona del camino p en el instante *t*, el valor x_p correspondiente a la cantidad de flujo asignado al camino p dividido por el valor de aptitud de la solución construida.

Algoritmo ACO 2 – Redistribución de flujos basado en ACO. Este algoritmo tiene un enfoque distinto al anterior. Partiendo de una solución generada aleatoriamente (un vector solución), 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 flujo, a otras se le sumará y a otras permanecerán inalteradas, obteniendo una nueva solución.

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, donde 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.

$$\tau = \begin{bmatrix} \tau_{x1}^{-1} & \tau_{x2}^{-1} & \tau_{x3}^{-1} & \tau_{xn}^{-1} \\ \tau_{x1}^{0} & \tau_{x2}^{0} & \tau_{x3}^{0} & \dots \tau_{xn}^{0} \\ \tau_{x1}^{+1} & \tau_{x2}^{+1} & \tau_{x3}^{+1} & \tau_{xn}^{+1} \end{bmatrix} \tag{8}$$

Es decir, τ_{xi}^{-1} corresponde a la feromona para restar 1 a la componente xi; τ_{xi}^{0} corresponde a la feromona para no alterar a la componente xi y τ_{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:

$$\eta(1,j) = \begin{cases} 1 \; \forall \; j \; donde \; x(j) \; \neq 0 \\ 0 \; \forall \; j \; donde \; x(j) = 0 \end{cases} \quad (9)$$

$$\eta(2,j) = 1 \; \forall \; j \quad (10)$$

$$\eta(3,j) = \begin{cases} 1 \; \forall \; j \; donde \; x(j) < demanda \\ 0 \; \forall \; j \; donde \; x(j) = demanda \end{cases} \quad (11)$$

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

Esta información heurística garantiza que ningún elemento de la solución sea negativo, y que ninguna componente del vector solución supere las unidades de flujo demandadas por un requerimiento, aunque 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.

La actualización de la matriz de feromona se realizará de la misma forma que el algoritmo anterior.

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, por lo que 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.

3.3 Optimización por Algoritmo de Murciélagos (BA – Bat Algorithm)

Dentro de este conjunto de metaheurísticas basado en población se encuentra el algoritmo de murciélago (Bat Algorithm, BA), una técnica propuesta en [17], basado en el comportamiento de un conjunto de estos pequeños animales en búsqueda de alimento utilizando su capacidad de ecolocalización (EL). Podría considerarse como una evolución del algoritmo PSO, donde la actualización de la posición (vector solución) se realiza ajustando los valores de frecuencia, volumen y velocidad.

$$f_i = f_{min} + (f_{max} - f_{min})\,\beta \quad (12)$$

$$v_i^t = v_i^{t-1} + (x_i^t - x^*) f_i \quad (13)$$

$$x_i^t = x_i^{t-1} + v_i^t \quad (14)$$

Siendo x_i la posición en el espacio de búsqueda, v_i es la velocidad y f_i la frecuencia de los pulsos emitidos, mientras que β es un vector aleatorio en el rango (0,1) y x* es la mejor solución obtenida. Los detalles del algoritmo fueron descriptos en [6].

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

Es común que los métodos heurísticos y meta-heurísticos no se utilicen en forma única para optimizar un problema. Luego de la experimentación realizada con los algoritmos descriptos, pudo observarse que el algoritmo de murciélago tenía la capacidad de llegar a soluciones óptimas actuando con mayor rapidez que sus pares, pero cuando los espacios de búsqueda toman un tamaño considerable, crece la probabilidad de quedar atrapado en óptimos locales.

Por otra 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 hibridizar el algoritmo inspirado en murciélagos con el algoritmo inspirado en hormigas para potenciar las características de cada uno.

Técnicamente 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 un algoritmo inspirado en colonia de hormigas. 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
    α_{BA} *y* γ
Definir Parámetros ACO:
    Cantidad de hormigas
    Número de iteraciones ACO
    α_{aco} - β - ρ - τ_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 ε 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 Experimentación y Resultados

Para la experimentación de los algoritmos desarrollados, se consideran cuatro casos de redes formadas por 4, 6, 8 y 10 nodos (Fig. 1 y 2) sobre las cuales establecieron los requerimientos: t_1=*(1, 2, 10),* t_2=*(2, 3, 10) y* t_3=*(1, 3, 20).*

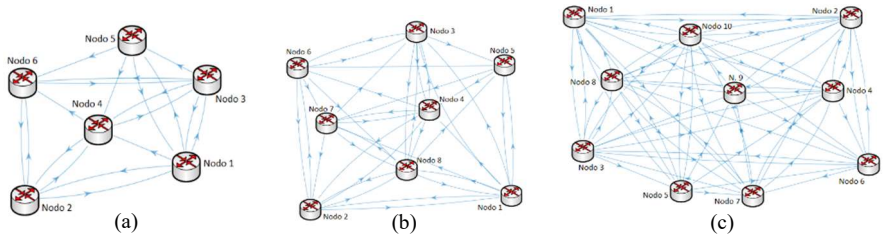

Fig. 2. Redes utilizadas para la experimentación con los algoritmos: (a) de 6 nodos, (b) red de 8 nodos, (c) red de 10 nodos.

Para todos los casos se determinan experimentalmente los mejores parámetros de configuración para cada algoritmo. Para los casos 1 y 2 los mejores parámetros encontrados se detallan en la tabla 1 y para los casos 3 y 4 en la tabla 2.

Table 1. Parámetros redes de 4 y 6 nodos.

Alg.	**N**	**Partículas**	**Iter.**	***C1***	***C2***	***w***
PSO	4	20	120	5	1	1
PSO	6	400	250	8	0.5	1

		Hormigas	Iter	α	β	ρ		τ_0	
ACO 1	4	50	250	2	0.5	0.009		0.1	
ACO 1	6	20	180	3	1	0.01		0.1	
		Hormigas	**Iter**	**α**	**β**	**ρ**	**τ_0**	**Iter. reset**	
ACO 2	4	5	50	-1	1	0.01	0.1	30	
ACO 2	6	30	30	-1.5	1	0.01	0.1	30	
		Murc.	**Iter**	**f_{min}**	**f_{max}**	**w**	**A_0**	**α**	**γ**
BA	4	40	20	0	6	0	1.2	0.99	0.5
BA	6	100	50	0	7	0	1.5	0.99	0.5

Table 2. Parámetros algoritmo BPA Redes de 8 y 10 nodos.

Nodos	Murc	It	f_{min}	f_{max}	A_0	α	γ	Hor	Iter	α	β	ρ	τ_0	Iter.reset
8	100	50	0	5	1.2	0.9	0.5	5	50	-1	1	0.01	0.1	30
10	100	50	0	5	1.2	0.9	0.5	30	3000	-1	1	0.01	0.1	100

Para la red de 4 nodos (Fig 1), el vector solución está compuesto por 3 vectores de 3 componentes cada uno. Los algoritmos desarrollados tienen un desempeño similar, diferenciándose en el tiempo empleado para encontrar las soluciones como puede apreciarse en la Tabla 3. El algoritmo BA es el que arriba con mayor celeridad a las soluciones y con muy baja varianza. Le sigue el algoritmo ACO2 PSO quien presenta una dispersión mayor, y finalmente el algoritmo ACO que exhibe tiempos de convergencia mayores que el resto de las estrategias, a cambio de un mayor balanceo de cargas.

Tabla 3. Comparativa del tiempo empleado Red 4 nodos.

Algoritmo	PSO	ACO	ACO2	BA	BPA
Porcentaje de efectividad	100%	95%	96%	100%	-
Tiempo Medio	0.34	0.24	275.33	0.085	-
Tiempo Mediano	0.251	0.13	209.78	0.067	-
Tiempo Medio	2.25	0.96	583.63	0.884	-
Tiempo Mínimo	0.0138	0.036	95.68	0.0017	-
Tiempo Máximo	1.7	2.5	1201.4	0.7	-

Para la red de 6 nodos, la solución está dada por un vector compuesto por 3 vectores de 16, 9 y 11 componentes cada uno, y los resultados son similares al ensayo anterior, aunque comienzan a ser notables las diferencias en cuanto a los tiempos de cómputo. El algoritmo ACO claramente presenta los mayores tiempos para encontrar la mejor solución, y en este caso, PSO también presenta tiempos altos. En este caso los mejores tiempos se encontraron con el algoritmo ACO2 seguido por BA, con valores similares.

Como es de esperar, 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.

El resumen de estos experimentos se muestra en la Tabla 4.

Table 4. Comparativa del tiempo empleado Red 6 nodos

Algoritmo	PSO	ACO	ACO2	BA	BPA
Porcentaje de efectividad	94%	100%	96%	98%	-
Tiempo Medio	107.55	546.17	2.77	3.98	-
Tiempo Mediano	66.90	552.95	2.52	2.97	-
Tiempo Medio	414.19	933.53	5.29	20.11	-
Tiempo Mínimo	5.49	429.10	1.22	0.96	-
Tiempo Máximo	418.48	716.08	8.80	21.84	-

Para el caso de la red de 8 nodos, el tamaño del vector solución se incrementa considerablemente, y queda formado por 3 vectores y 289, 170, 179 componentes cada uno. Las alternativas de asignación de flujo crecen enormemente y con ellos los tiempos empleados por los algoritmos, que comienzan a ser significativos.

La técnica de distribución de flujos basada en colonia de hormigas (Algoritmo ACO) no resulta útil dados los tiempos que insume, y se utiliza el algoritmo híbrido BPA, capaz de mejorar las soluciones de las estrategias puras, alcanzando el óptimo en casi la totalidad de los casos y en un tiempo mediano que ronda los 8 minutos. La Fig. 3 establece la comparación de las estrategias para el caso de 8 nodos en relación a la calidad de la solución.

Finalmente se ensaya una red de 10 nodos, cuyo vector solución se conforma con 3 subvectores de 5807, 5025 y 4016 componentes respectivamente. Para esta red solo toman relevancia los algoritmos BA y BPA, ya que los otros no son capaces de entregar buenas soluciones en tiempos razonables. La comparación de estos algoritmos es evidente, debido a que el algoritmo BPA es una mejora por hibridación de BA, por lo que todas las soluciones de BPA tendrán mejores valores de aptitud que las soluciones BA a excepción de algunas soluciones atípicas.

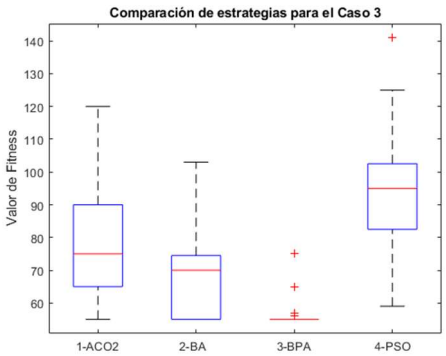

Fig. 3. Comparativa resultados de los ensayos para la red de 8 nodos.

Como se puede observar en la Fig. 4, BPA presentó 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, a cambio de un tiempo mediano elevado: 55 h.

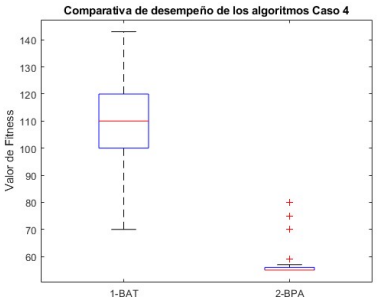

Fig. 4. Comparativa resultados de los ensayos para la red de 10 nodos.

Conclusiones

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 tamaño de la instancia, se incrementa la demanda de tiempo y la calidad de las soluciones obtenidas disminuye

Para instancias medias del problema (red de 6 nodos) el algoritmo ACO2 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 ACO, es el algoritmo que mayor tiempo demandado para encontrar las soluciones óptimas con una amplia diferencia. No obstante, a diferencia de otros algoritmos entrega soluciones con un mejor balance de cargas.

Para instancias grandes del problema (red de 8 nodos) se descarta el algoritmo ACO, PSO comienza a demandar tiempos que crecen considerablemente y las soluciones encontradas son de menor calidad. Los ACO2 y BA se comportan razonablemente bien, aunque con menores valores de efectividad. 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 alcanzar la mejor solución al problema planteado.

Una vez más, la combinación de heurísticas, adaptadas a la resolución del problema permite alcanzar mejor desempeño que el que logra cada estrategia aplicada por separado. La hibridización hace posible potenciar los aspectos más fuertes de cada una de ellas.

Referencias

1. 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)

2. 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
3. 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
4. Delfino, A.; Rivero, S.; San Martín, M. (2006) *"Ingeniería de tráfico en redes MPLS"* – I Congreso Regional de Telecomunicaciones MVD Tecom 2006.
5. 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.
6. Carletto, Javier A.; Carnero, Mercedes; Diaz, Javier; Hernandez, José Algoritmo Bioinspirado en Murciélagos Aplicado a Optimización en redes MPLS Congreso Latinoamericano de Ingeniería y Ciencias Aplicadas CLICAP 2022- San Rafael, Mendoza- Argentina 2022
7. 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.
8. 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.
9. 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
10. 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).
11. Gero, J. (1987) "*Artificial intelligence and engineering optimization. Engineering Optimization",* vol. 12, pp. 89-90.
12. 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.
13. 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
14. 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
15. 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.
16. 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..
17. 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