Evaluando la calidad de metaheurísticas simples para el problema de Steiner generalizado.

Sergio Nesmachnow

Centro de Cálculo, Instituto de Computación,
Facultad de Ingeniería, Universidad de la República,
Montevideo, Uruguay,
E-mail: sergion@fing.edu.uy

Resumen - Este trabajo presenta la evaluación empírica de la calidad de resultados obtenidos por técnicas metaheurísticas simples para la resolución del problema de Steiner generalizado (GSP). Este problema modela el diseño de redes de comunicaciones de alta confiabilidad topológica exigiendo la existencia de un número variable de caminos disjuntos entre cada par de nodos terminales de comunicación. La solución del GSP se construye utilizando nodos intermedios para asegurar la redundancia de caminos, y tratando de minimizar el costo total. Se trata de un problema NP-difícil, para el que existen pocos algoritmos propuestos. Este trabajo presenta la resolución de instancias del GSP cuyas soluciones óptimas son conocidas, utilizando diferentes técnicas metaheurísticas simples codificadas sobre MALLBA, una biblioteca de propósito general para optimización combinatoria. Se compara la calidad de los resultados obtenidos por los diferentes algoritmos, reportándose promisorios resultados para algunas de las técnicas estudiadas.

Palabras clave - Metaheurísticas, problema de Steiner generalizado, redes de comunicaciones confiables.

Destinado al Wokshop de Agentes y Sistemas Inteligentes

1 Introducción

En los últimos años, el rápido desarrollo de la infraestructura de redes, del software y de servicios de Internet renovó el interés en problemas de diseño de redes de comunicaciones. Una cuestión importante consiste en hallar una topología de conexión de los nodos de una red cuyas propiedades aseguren la comunicación confiable de datos. Dado el continuo crecimiento en el tamaño de las redes, los problemas de optimización frecuentemente superan la capacidad de los algoritmos exactos tradicionales. En este contexto, las técnicas heurísticas y metaheurísticas se aplican al diseño de redes de comunicaciones confiables para resolver problemas de dimensiones reales en tiempos razonables.

Este trabajo presenta la aplicación de diversas metaheurísticas simples a la resolución de la clase de problemas de diseño de redes de comunicaciones confiables modelados bajo el denominado Problema de Steiner Generalizado (*Generalizad Steiner Problem* – GSP).

Dada una red de comunicaciones, el GSP propone diseñar una subred de mínimo costo que verifique requisitos prefijados de conexión entre pares de nodos distinguidos denominados *terminales*. Usualmente, minimizar el costo de las conexiones se contrapone con el objetivo de maximizar las propiedades de confiabilidad de la red: un modelo que no agregue redundancia de caminos conducirá a una topología de árbol, que no es capaz de soportar ni un fallo en sus componentes. El GSP incorpora requisitos adicionales de conectividad para garantizar la confiabilidad en las comunicaciones que demandan los escenarios reales. El problema ha sido poco estudiado en el pasado; en nuestro grupo de trabajo hemos diseñado algoritmos evolutivos puros e híbridos, en sus modelos secuenciales y paralelos, para su resolución. Como no existen problemas de prueba estándares, se evaluaron los algoritmos sobre instancias generadas aleatoriamente, cuyos valores óptimos no son conocidos.

Este trabajo presenta un análisis empírico de la calidad de resultados de diversas metaheurísticas simples al resolver instancias del GSP cuyas soluciones óptimas son conocidas. Los casos de prueba fueron construidos replicando grafos pequeños para los cuales la solución puede hallarse en forma exhaustiva en tiempos de ejecución moderados.

El artículo se organiza del modo que se describe a continuación. La sección 2 presenta de modo general a las técnicas heurísticas y metaheurísticas. La sección 3 describe el problema de Steiner generalizado y sus variantes. Los algoritmos considerados en el estudio se describen en la sección 4. Los detalles de implementación de los algoritmos se ofrecen en la sección 5. En la sección 6 se presentan los casos de prueba diseñados, previo a discutir los experimentos y analizar los resultados obtenidos en la sección 7. Por último, en la sección final se formulan las conclusiones y líneas de trabajo futuro.

2 Técnicas heurísticas y metaheurísticas

Al tratar con problemas de optimización combinatoria NP difíciles, para los cuales la complejidad de los algoritmos conocidos aumenta de manera superpolinomial con el tamaño del problema, la aplicabilidad de los métodos exactos de resolución se encuentra limitada por el enorme tiempo y consumo de recursos computacionales que demandan.

Cuando las técnicas de resolución analítica, la programación matemática y los métodos de búsqueda exhaustiva no son aplicables, las *técnicas heurísticas* surgen como una alternativa viable para abordar problemas NP difíciles con espacio de soluciones de dimensión elevada. A diferencia de las técnicas exactas, las técnicas heurísticas no pueden garantizar a priori obtener la solución óptima del problema. En la mayoría de los casos ni siquiera pueden garantizar que la solución hallada tenga un cierto margen de error con respecto a la solución óptima. Pero en la práctica, numerosas técnicas heurísticas se comportan de manera sumamente satisfactoria para la resolución de complejos problemas de optimización,

y permiten obtener buenas soluciones aproximadas en tiempos razonables. En un nivel superior de abstracción, las *técnicas metaheurísticas* proporcionan esquemas genéricos para la resolución de problemas complejos. Estos enfoques genéricos pueden ser instanciados para producir algoritmos específicos que trabajan bajo un mismo lineamiento general.

Este trabajo presenta técnicas metaheurísticas basadas en trayectoria (Simulated Annealing y Variable Neighbourhood Search) y basadas en población (Algoritmos Evolutivos) para la resolución del GSP. Hemos explorado también Colonias de Hormigas, cuyos resultados se exponen en un trabajo aparte. Las metaheurísticas consideradas son "simples", en el sentido de que no utilizan información específica del problema en su mecanismo de exploración –salvo, obviamente, la evaluación de la función a optimizar, utilizada para guiar la búsqueda de los algoritmos–. Los mecanismos de exploración del espacio de soluciones son completamente estocásticos, y quedan definidos por operadores aleatorios.

3 El problema de Steiner generalizado

3.1 Formulación del GSP

La siguiente formulación del GSP se basa en la presentada en el compendio de problemas de optimización NP de Kahn y Crescenzi [6].

Considérense los siguientes elementos:

- Un grafo no dirigido $G = (V,E)$, siendo V el conjunto de nodos y E el conjunto de aristas que representan a los enlaces bidireccionales de comunicación entre nodos.
- Una matriz de costos C asociados a las aristas del grafo G.
- Un subconjunto fijo del conjunto de nodos *terminales* $T \subseteq V$, de cardinalidad $n_T = |T|$, tal que $2 \leq n_T \leq n$, siendo $n = |V|$ la cardinalidad del conjunto de nodos V.
- Una matriz n_T x n_T simétrica $R = r_{ij}$ con $i,j \in T$, que indica los requisitos de conectividad –cantidad de caminos disjuntos– entre pares de nodos terminales i y j.

El GSP plantea encontrar un subgrafo $G_T \subseteq G$ de costo mínimo, tal que todo par de nodos $i,j \in T$, sean r_{ij} arista-conexos en G_T, es decir que existan r_{ij} *caminos disjuntos*, que no comparten aristas, entre los nodos i y j en G_T. Sobre los nodos no terminales no se plantean requisitos de conectividad; son llamados *nodos de Steiner*, y pueden formar parte o no de la solución óptima, de acuerdo a la conveniencia de utilizarlos.

3.2 Variantes de problemas de Steiner, complejidad y resolución

La complejidad del problema de Steiner obedece a la generalidad de su planteo, al permitir requisitos variables de conectividad entre pares de nodos terminales. Ciertas variantes simplifican estos requisitos: la subclase de *problemas de k-conexión* exigen un número fijo k de caminos disjuntos entre pares de nodos terminales. El caso más simple exige sólo un camino entre pares de nodos; como la solución a este problema tiene topología de árbol, se conoce como *problema del árbol de Steiner*.

El GSP pertenece a la clase de problemas NP difíciles [6]. El propio problema del árbol de Steiner, que plantea las restricciones más simples de conectividad, es NP-completo [7]. Este hecho hace difícil su resolución mediante algoritmos exactos cuando aumenta el tamaño del problema. Las técnicas heurísticas se utilizan como alternativas para abordar instancias complejas y hallar buenas soluciones en tiempos razonables.

Aunque varias heurísticas han sido empleadas para resolver variantes simples del problema (problemas del *árbol de Steiner* o de *k-conexión*), no existen antecedentes de resolución del problema generalizado. En nuestro grupo de trabajo hemos diseñado algoritmos evolutivos [9,10] para resolver el problema de Steiner generalizado, tomando como base la idea de mantener la codificación y los operadores tan simples como fuera posible.

4 Algoritmos Considerados en el estudio

Esta sección presenta las metaheurísticas consideradas en el estudio: Algoritmos Genéticos (AG) en su formulación clásica y en su variante CHC, Simulated Annealing (SA), Búsqueda de Vecindades Variables (*Variable Neiborghood Search* – VNS) y dos técnicas híbridas que combinan AG con SA y VNS respectivamente.

4.1 Algoritmo Genético

La formulación clásica de un Algoritmo Genético se basa en el esquema de un Algoritmo Evolutivo (AE) de la Figura 1 **Error! Reference source not found.**. El AG define operadores de selección, cruzamiento y mutación para aplicar a la población en cada generación. Los Algoritmos Genéticos están ampliamente difundidos por su versatilidad para resolver problemas de optimización. El AG diseñado se basa en el presentado en trabajos previos por el autor [9,10].

```
Inicializar(P(0))
generacion = 0
Evaluar(P(0))
mientras (no CriterioParada) hacer
  Padres = Seleccion(P(generacion))
  Hijos = Operadores de Reproduccion(Padres)
  NuevaPop = Reemplazar(Hijos,P(generacion))
  generacion ++
  P(generacion) = NuevaPop
retornar Mejor Solucion Hallada
```

Figura 1: Esquema de un Algoritmo Evolutivo.

4.2 Algoritmo CHC

CHC es una variante del AG clásico propuesta por Eshelman [4], que utiliza una estrategia de selección elitista y el operador de cruzamiento uniforme (HUX), que intercambia la mitad de los bits diferentes entre dos individuos padre. Sólo permite el cruce entre individuos que difieran bit a bit una cierta distancia, inicializada en 1/4 del largo del cromosoma y disminuida en 1 en cada generación en que no se generan descendientes. No utiliza mutación, sino que introduce diversidad mediante un mecanismo de reinicialización, aplicado al detectar la convergencia. La Figura 2 presenta un esquema del algoritmo CHC.

```
Inicializar(P(0))
generacion = 0
distancia = LargoCromosoma/4
Evaluar(P(0))
mientras (no CriterioParada) hacer
  Padres = Seleccion(P(generacion))
  Hijos = HUX(Padres)
  Evaluar(Hijos))
  NuevaPop = Reemplazar(Hijos,P(generacion))
  Si (NuevaPop == P(generacion)) entonces
     distancia --
  generacion ++
  P(generacion) = NuevaPop
  Si (distancia == 0) entonces
     Reinicializacion(P(generacion))
     distancia = LargoCromosoma/4
retornar Mejor Solucion Hallada
```

Figura 2: Esquema del algoritmo CHC.

4.3 Simulated Annealing

Simulated Annealing define una búsqueda local basada en una generalización del método Monte Carlo para hallar la configuración más estable de un sistema físico de *n* partículas [8]. SA trabaja con una solución candidata para el problema (análoga al estado de un sistema físico) y con una función objetivo (análoga a la función de energía) para la que se quiere hallar el mínimo global (análogo al estado más estable). La exploración se lleva a cabo mediante un *operador de transición*. SA puede aceptar peores soluciones que la actual, de acuerdo a un parámetro temperatura (*T*) que no tiene un análogo en el problema de optimización, por lo cual definir un *esquema de enfriamiento* para evitar mínimos locales es un arte. Los otros parámetros del método (valor inicial de *T*, número de iteraciones en cada paso y largo de la cadena de Markov) usualmente se determinan empíricamente. La Figura 3 presenta un esquema para el algoritmo SA.

```
Inicializar(T)
paso = 0;
Sol = SolucionInicial()
Valor = Evaluar(Sol)
repetir
  repetir
    paso ++
    NuevaSol = Generar(Sol,T)      // Transición
    NuevoValor = Evaluar(NuevaSol)
    si Aceptar(valor,NuevoValor,T)
       Sol = NuevaSol; Valor = NuevoValor;
  hasta ((paso mod LargoCadenaMarkov)==0)
  T = Actualizar(T)
hasta (CriterioParada)
retornar Sol
```

Figura 3: Esquema del algoritmo Simulated Annealing.

4.4 Búsqueda de Vecindades Variables

La Búsqueda de Vecindades Variables aplica una búsqueda local por vecindades, definiendo un mecanismo para el cambio sistemático del entorno en el que se buscan nuevas soluciones [11]. El VNS presentado en este trabajo sigue un esquema *general*, utilizando dos conjuntos de estructuras de vecindad, uno para la fase aleatoria de *agitación*, que genera una solución al azar y otro para la búsqueda local, que utiliza un mecanismo de descenso en vecindades variables (VND). El esquema del algoritmo se presenta en la Figura 4.

```
Inicializar estructuras de entorno (N_k y N_j)
paso = 0;
Sol = SolucionInicial()
Valor = Evaluar(Sol)
repetir
    k = 1
    repetir
        Sol'=Agitar(Sol) // Generar solución en entorno N_k de Sol
        NuevaSol=VND(Sol',N_j) // Aplica VND con entornos N_j
        NuevoValor = Evaluar(NuevaSol)
        si NuevoValor < Valor
            Sol = NuevaSol; Valor = NuevoValor;
            k = 1
        sino
            k ++
    hasta k==k_max
hasta (CriterioParada)
retornar Sol
```

Figura 4: Esquema del algoritmo Búsqueda de Vecindades Variables.

4.5 Algoritmos Híbridos

Las técnicas de hibridación proponen la inclusión de conocimiento para mejorar un método de búsqueda [2]. Puede instrumentarse a nivel de codificación y/o a través de operadores específicos (*híbridos fuertes*) o combinando diversos algoritmos (*híbridos débiles*).

En este trabajo hemos utilizado esta última alternativa, combinando el esquema evolutivo del AG con la búsqueda local de los algoritmos SA y VNS. Se definen dos algoritmos híbridos, AG+SA y AG+VNS respectivamente, donde el método de búsqueda local se incorpora como operador interno del AE, aplicándose luego de los operadores evolutivos.

5 Implementación

5.1 Codificación del problema

Se utilizó una codificación binaria simple basada en aristas para representar grafos que son soluciones factibles del GSP. Una solución se representa como un arreglo de bits (indexado entre 0 y $|E|$-1); cada bit indica la presencia o ausencia de una arista del grafo original. La Figura 5 presenta un grafo de ejemplo y su codificación en la representación propuesta: los nodos terminales se indican con color oscuro, los nodos de Steiner con color claro, las aristas presentes en la solución con líneas llenas y las no presentes con líneas punteadas.

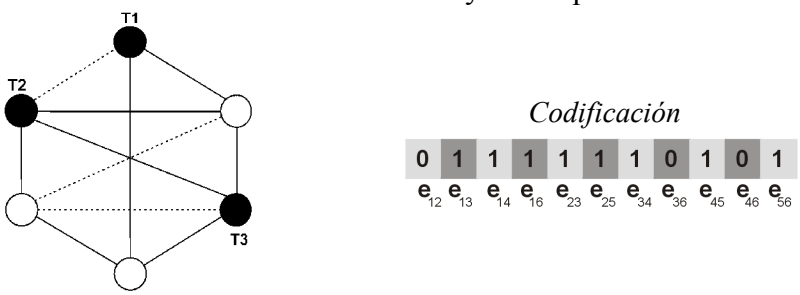


Figura 5: Codificación binaria basada en aristas.

Se trabajó descartando las soluciones no factibles generadas en el proceso de búsqueda [3,9,10]. Esta decisión simplifica los algoritmos, evita cuantificar la distancia al conjunto de soluciones factibles, y/o definir un modelo adecuado de penalización. El chequeo de factibilidad tiene dos componentes: una heurística descarta soluciones si los grados de algún terminal son menores que el máximo requerimiento de conexión para ese nodo. En caso contrario, se utiliza una variante del algoritmo de Ford–Fulkerson [5] para hallar caminos entre pares de nodos terminales, considerando uno como fuente y el otro como pozo. Asumiendo la capacidad de las aristas unitaria, el flujo máximo entre fuente y pozo coincide con el número máximo de caminos disjuntos entre los nodos terminales. Si éste es menor que el requerimiento correspondiente, la solución no es factible y se descarta.

5.2 Función a optimizar

La función a optimizar evalúa al costo del diseño de la red de comunicaciones representada por una solución, sumando de los costos de cada arista, tal como presenta la Ecuación 1.

$$f = \sum_{i=0}^{|E|-1} C(i)$$

Ecuación 1: Función a optimizar en el GSP.

5.3 Operadores y parámetros

Los operadores y parámetros utilizados por los algoritmos diseñados corresponden a:

- SA: Transición que invierte los valores de cinco aristas, esquema proporcional de decaimiento para la temperatura ($T_k = \alpha.T_{k-1}$ con $\alpha = 0.99$).

- AG: Selección proporcional, cruzamiento de dos puntos, mutación de inversión.
- CHC: Cruzamiento uniforme, selección elitista, reinicialización basada en la transición propuesta para SA, que se aplica iterativamente hasta obtener una solución factible.
- VNS: Vecindad definida por distancia de Hamming en la codificación binaria.

No se realizaron experimentos de configuración de parámetros para cada algoritmo, trabajándose con valores derivados de los trabajos previos [9,10].
Se definió un criterio de parada de esfuerzo prefijado en 2000 generaciones para los AE y 10000 iteraciones para SA y VNS. En los AE se utilizó una población de 120 individuos, inicializada mediante eliminación aleatoria de hasta 5% de las aristas originales, aplicada hasta generar soluciones factibles. La probabilidad de cruzamiento se fijó en 0.9 y la de mutación en 0.01. En CHC el proceso de reinicialización involucra el 35% de la población.
Los parámetros de búsqueda del VNS corresponden a k_{max}=3 para la fase de agitación, y j=3 para la búsqueda local que aplica VND.
En los algoritmos híbridos se aplica el operador interno (VNS ó SA) con probabilidad 0.01 en cada generación. Para GA+SA se utilizó una cadena de Markov corta (de largo 10) y 20 iteraciones, para reducir el costo computacional de aplicar SA como operador interno al AG.

5.4 La biblioteca MALLBA

Los algoritmos se implementaron sobre MALLBA, una biblioteca que proporciona esqueletos de software que implementan algoritmos de optimización [1]. Los detalles de los algoritmos, la interacción con el problema y el paralelismo se implementan mediante clases C++ que abstraen a las entidades involucradas en cada método de resolución:

- Las clases *provistas* implementan cada algoritmo, independizándose del problema. Las principales clases provistas son `Solver` (el algoritmo) y `SetUpParams` (fija parámetros).
- Las clases *requeridas* especifican la información relevante del problema a resolver. Cada esqueleto incluye las clases requeridas `Problem` y `Solution` que encapsulan las entidades dependientes del problema necesarias para los métodos de resolución.

La biblioteca MALLBA está disponible en **http://neo.lcc.uma.es/mallba/easy-mallba**.

5.5 Plataforma de ejecución

Para la ejecución de los algoritmos se utilizó un computador Intel Pentium III a 2.4 GHz, con 512 Mb RAM y sistema operativo SuSE Linux 8.0.

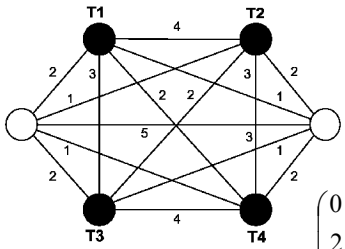
6 Problemas de prueba

El GSP ha sido poco estudiado, y no se han diseñado instancias de prueba estándar. En trabajos previos se usó un generador aleatorio de problemas para evaluar los algoritmos diseñados [9, 10]. Al no conocer las soluciones óptimas de las instancias utilizadas, es posible comparar algoritmos, pero no estimar la distancia de los resultados a los óptimos. Este trabajo plantea el análisis empírico de la calidad de resultados de metaheurísticas simples al resolver instancias del GSP con soluciones óptimas son conocidas. Los casos de prueba se construyeron replicando grafos cuyas soluciones pueden hallarse en tiempos de ejecución moderados utilizando un algoritmo de búsqueda exhaustiva. Las réplicas tienen una estructura separada en componentes conexas, pero este hecho es indiferente para los algoritmos, que considerarán al grafo replicado como un único problema de estructura desconocida, al no utilizar operadores de búsqueda dependientes del problema. Queda claro que estos casos de prueba no constituyen escenarios realistas, y que no formulan el peor caso en cuanto a la complejidad de las instancias planteadas. Sin embargo, permiten estimar la calidad de resultados obtenida por las metaheurísticas consideradas.

Las características de los grafos sin replicar se presentan en la Tabla 1, indicando el número total de nodos, de nodos terminales, de aristas y el grado de conectividad promedio (número de aristas sobre el número de aristas del grafo completo).
Las Figuras 6, 7, 8 y 9 presentan los problemas de prueba.

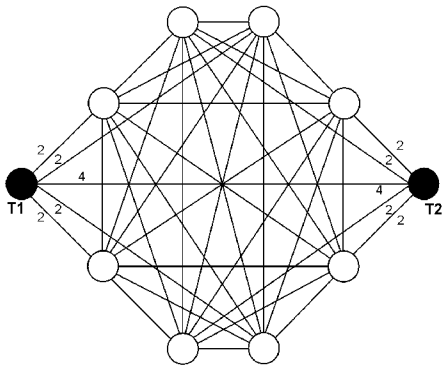
Instancia	*Nodos*	*Terminales*	*Aristas*	*Grado conectividad promedio*
grafo_1	6	4	15	1.000
grafo_2	10	2	37	0.561
grafo_3	16	6	25	0.077
grafo_4	15	5	26	0.248

Tabla 1: Características de los grafos sin replicar.



$$R = \begin{pmatrix} 0 & 2 & 3 & 4 \\ 2 & 0 & 2 & 3 \\ 3 & 2 & 0 & 2 \\ 4 & 3 & 2 & 0 \end{pmatrix}$$

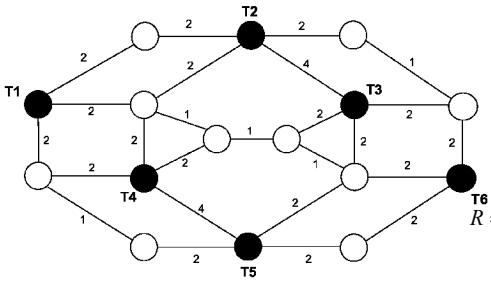
Figura 6: grafo_1.



Los costos de las aristas no etiquetados tienen valor 1.

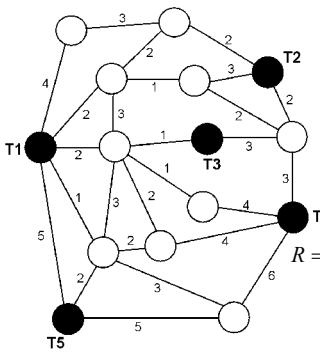
$$R = \begin{pmatrix} 0 & 4 \\ 4 & 0 \end{pmatrix}$$

Figura 7: grafo_2.



$$R = \begin{pmatrix} 0 & 2 & 3 & 2 & 3 & 2 \\ 2 & 0 & 2 & 3 & 2 & 3 \\ 3 & 2 & 0 & 2 & 3 & 2 \\ 2 & 3 & 2 & 0 & 2 & 3 \\ 3 & 2 & 3 & 2 & 0 & 2 \\ 2 & 3 & 2 & 3 & 2 & 0 \end{pmatrix}$$

Figura 8: grafo_3.



$$R = \begin{pmatrix} 0 & 3 & 3 & 2 & 1 \\ 3 & 0 & 3 & 2 & 1 \\ 3 & 3 & 0 & 1 & 1 \\ 2 & 2 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 \end{pmatrix}$$

Figura 9: grafo_4.

Las instancias presentadas fueron replicadas entre 2 y 13 veces, creando instancias de mayor complejidad que permitieran estudiar la eficacia de los algoritmos considerados a medida que aumenta el tamaño del problema. Los detalles de las instancias replicadas de mayor dimensión (13 réplicas) se presentan en la Tabla 2.

instancia	*Nodos*	*Terminales*	*Aristas*
grafo_1	78	52	195
grafo_2	156	26	481
grafo_3	208	78	325
grafo_4	195	65	338

Tabla 2: Características de las réplicas más complejas generadas.

7 Resultados

Los resultados de los algoritmos para las instancias del GSP estudiadas se resumen en las Tablas 3, 4, 5 y 6. Se presentan promedios y mejores valores de "efectividad" respecto al óptimo para cada algoritmo sobre las 10 ejecuciones independientes realizadas para los tamaños (réplicas) de cada problema. Las información se resume en las Figuras 10, 11, 12 y 13, presentando gráficamente promedios y mejores resultados para cada algoritmo (se omiten los mejores para GA+SA al no diferir significativamente de GA+VNS).

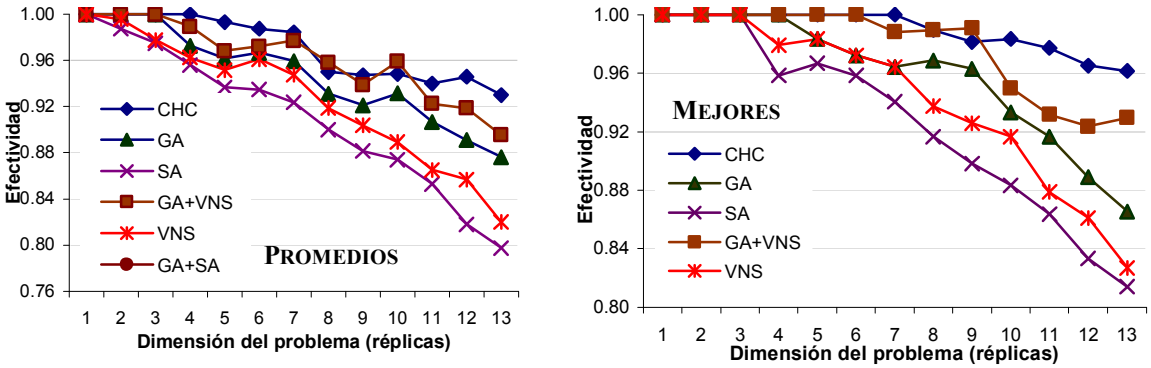


Figura 10: Calidad de resultados para la instancia grafo_1.

	CHC		GA		SA		VNS		GA+VNS		GA+SA	
#rep	Mejor	Prom.	Mejor	Prom.	Mejor	Prom.	Mejor	Prom.	Mejor	Prom.	Mejor	Prom.
1	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
2	1.000	1.000	1.000	1.000	1.000	0.987	1.000	0.995	1.000	1.000	1.000	1.000
3	1.000	1.000	1.000	1.000	1.000	0.975	1.000	0.977	1.000	1.000	1.000	1.000
4	1.000	1.000	1.000	0.972	0.958	0.956	0.979	0.962	1.000	0.989	1.000	0.977
5	1.000	0.993	0.983	0.961	0.966	0.936	0.983	0.951	1.000	0.968	0.984	0.963
6	1.000	0.987	0.972	0.966	0.958	0.934	0.972	0.961	1.000	0.972	0.973	0.966
7	1.000	0.984	0.964	0.959	0.940	0.923	0.964	0.947	0.988	0.977	0.967	0.953
8	0.989	0.950	0.968	0.931	0.916	0.900	0.937	0.918	0.989	0.958	0.966	0.946
9	0.981	0.947	0.963	0.921	0.898	0.881	0.925	0.903	0.990	0.938	0.964	0.923
10	0.983	0.948	0.933	0.931	0.883	0.874	0.916	0.889	0.950	0.959	0.933	0.929
12	0.977	0.940	0.916	0.906	0.863	0.853	0.878	0.865	0.931	0.922	0.916	0.901
13	0.965	0.946	0.888	0.891	0.833	0.818	0.861	0.856	0.923	0.918	0.923	0.912

Tabla 3: Resultados de los algoritmos para la instancia grafo_1.

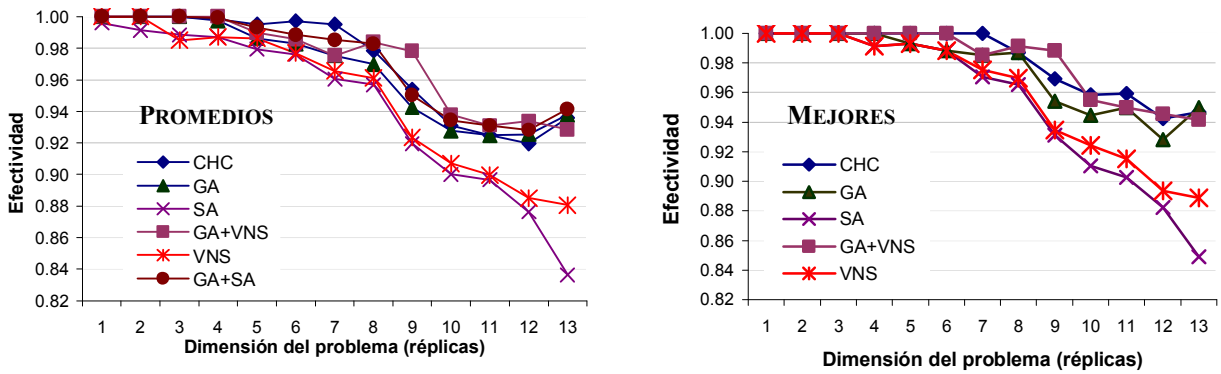


Figura 11: Calidad de resultados para la instancia grafo_2.

	CHC		GA		SA		VNS		GA+VNS		GA+SA	
#rep	Mejor	Prom.	Mejor	Prom.	Mejor	Prom.	Mejor	Prom.	Mejor	Prom.	Mejor	Prom.
1	1.000	1.000	1.000	1.000	1.000	0.995	1.000	1.000	1.000	1.000	1.000	1.000
2	1.000	1.000	1.000	1.000	1.000	0.991	1.000	1.000	1.000	1.000	1.000	1.000
3	1.000	1.000	1.000	1.000	1.000	0.988	1.000	0.985	1.000	1.000	1.000	1.000
4	1.000	0.998	1.000	0.997	0.991	0.987	0.991	0.987	1.000	1.000	1.000	0.999
5	1.000	0.995	0.993	0.986	0.993	0.979	0.993	0.986	1.000	0.989	1.000	0.993
6	1.000	0.997	0.988	0.982	0.988	0.976	0.988	0.977	1.000	0.985	0.994	0.988
7	1.000	0.995	0.985	0.975	0.970	0.960	0.975	0.965	0.985	0.975	0.995	0.985
8	0.987	0.978	0.987	0.969	0.965	0.956	0.969	0.961	0.991	0.983	0.987	0.982
9	0.969	0.954	0.954	0.942	0.931	0.919	0.934	0.923	0.988	0.978	0.954	0.950
10	0.959	0.931	0.944	0.927	0.910	0.900	0.924	0.906	0.955	0.937	0.944	0.934
12	0.959	0.925	0.949	0.924	0.902	0.896	0.915	0.899	0.949	0.931	0.956	0.931
13	0.942	0.919	0.928	0.925	0.882	0.876	0.893	0.885	0.945	0.933	0.933	0.928

Tabla 4: Resultados de los algoritmos para la instancia grafo_2.

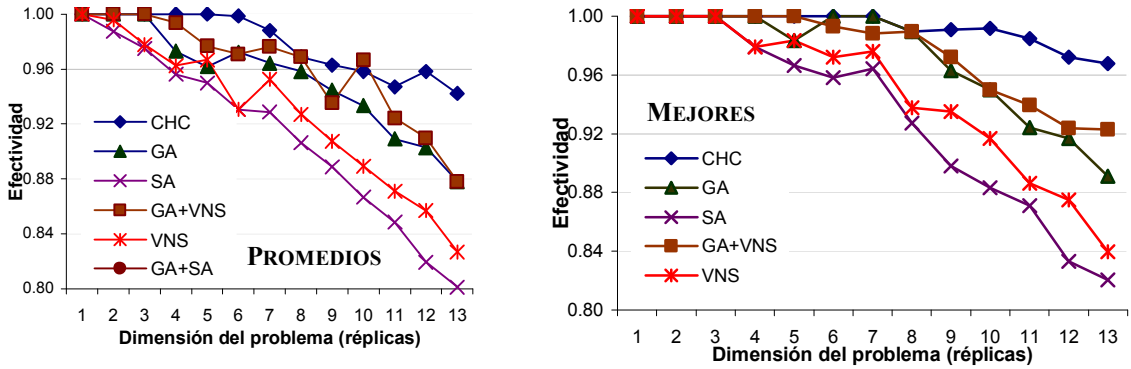


Figura 12: Calidad de resultados para la instancia grafo_3.

	CHC		GA		SA		VNS		GA+VNS		GA+SA	
#rep	Mejor	Prom.	Mejor	Prom.	Mejor	Prom.	Mejor	Prom.	Mejor	Prom.	Mejor	Prom.
1	1.0000	1.0000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
2	1.0000	1.0000	1.000	1.000	1.000	0.987	1.000	0.995	1.000	1.000	1.000	1.000
3	1.0000	1.0000	1.000	1.000	1.000	0.975	1.000	0.977	1.000	1.000	1.000	1.000
4	1.0000	1.0000	1.000	0.972	0.958	0.956	0.979	0.962	1.000	0.989	1.000	0.977
5	1.0000	1.0000	0.983	0.961	0.966	0.936	0.983	0.951	1.000	0.968	0.984	0.963
6	1.0000	0.9986	0.972	0.966	0.958	0.934	0.972	0.961	1.000	0.972	0.973	0.966
7	1.0000	0.9881	0.964	0.959	0.939	0.923	0.964	0.947	0.988	0.977	0.967	0.953
8	0.9896	0.9688	0.968	0.931	0.916	0.900	0.937	0.918	0.989	0.958	0.966	0.946
9	0.9907	0.9630	0.963	0.921	0.898	0.881	0.925	0.903	0.990	0.938	0.964	0.923
10	0.9917	0.9583	0.933	0.931	0.884	0.874	0.916	0.889	0.950	0.959	0.933	0.929
12	0.9848	0.9470	0.916	0.906	0.861	0.853	0.878	0.865	0.931	0.922	0.916	0.901
13	0.9722	0.9583	0.888	0.891	0.833	0.818	0.861	0.856	0.924	0.918	0.923	0.912

Tabla 5: Resultados de los algoritmos para la instancia grafo_3.

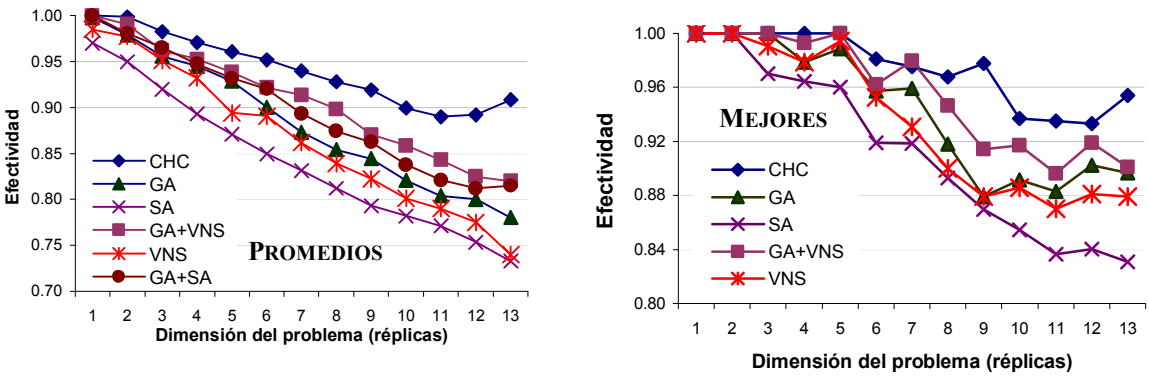


Figura 13: Calidad de resultados para la instancia grafo_4.

	CHC		GA		SA		VNS		GA+VNS		GA+SA	
#rep	Mejor	Prom.	Mejor	Prom.	Mejor	Prom.	Mejor	Prom.	Mejor	Prom.	Mejor	Prom.
1	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
2	1.000	0.999	1.000	1.000	1.000	0.987	1.000	0.995	1.000	1.000	1.000	1.000
3	1.000	0.983	1.000	1.000	1.000	0.975	1.000	0.977	1.000	1.000	1.000	1.000
4	1.000	0.971	1.000	0.972	0.958	0.956	0.979	0.962	1.000	0.993	1.000	0.977
5	1.000	0.961	0.983	0.961	0.966	0.950	0.983	0.966	1.000	0.976	1.000	0.965
6	0.981	0.952	1.000	0.972	0.958	0.930	0.972	0.930	0.993	0.970	1.000	0.972
7	0.976	0.940	1.000	0.964	0.952	0.928	0.976	0.952	0.988	0.976	0.976	0.964
8	0.968	0.928	0.989	0.958	0.916	0.906	0.937	0.927	0.989	0.968	0.979	0.958
9	0.978	0.919	0.963	0.944	0.898	0.888	0.935	0.907	0.972	0.935	0.944	0.923
10	0.937	0.900	0.950	0.933	0.883	0.866	0.916	0.889	0.950	0.966	0.933	0.933
11	0.935	0.890	0.924	0.909	0.871	0.848	0.886	0.871	0.939	0.924	0.916	0.916
12	0.933	0.892	0.916	0.902	0.833	0.819	0.875	0.856	0.923	0.909	0.923	0.916
13	0.953	0.908	0.896	0.780	0.830	0.733	0.839	0.826	0.922	0.878	0.907	0.884

Tabla 6: Resultados de los algoritmos para el grafo_4.

CHC		GA		SA		VNS		GA+VNS		GA+SA	
Mejor	**Prom.**	**Mejor**	**Prom.**	**Mejor**	**Prom.**	**Mejor**	**Prom.**	**Mejor**	**Prom.**	**Mejor**	**Prom.**
> 0.95	> 0.9	> 0.88	> 0.80	> 0.83	> 0.71	> 0.83	> 0.85	> 0.92	> 0.87	> 0.90	> 0.88

Tabla 7: Resultados comparativos.

Las técnicas basadas en población muestran su ventaja para la resolución de las cuatro instancias estudiadas: CHC, AG y las dos técnicas híbridas obtienen sistemáticamente los mejores resultados, superando a VNS y SA. Con los sencillos operadores propuestos, las técnicas basadas en trayectoria son menos competitivas, mostrando problemas para manejar la complejidad del GSP. La búsqueda exhaustiva en vecindades de VNS le permite superar a SA, que muestra los peores resultados para todos los casos estudiados. Se observa que potenciar al AG con VNS ó SA como operador interno ayuda a alcanzar mejores resultados en promedio, potenciando la búsqueda al introducir una nueva fuente de diversidad en el mecanismo evolutivo.

El algoritmo CHC presentó sistemáticamente los mejores resultados para las instancias estudiadas, siendo superado solamente en casos puntuales por el híbrido GA+VNS. Este hecho confirma resultados anteriores [9,10] donde CHC se mostró como la mejor alternativa para resolver el GSP. Tomando ventaja de sus operadores que proporcionan mayor diversidad en la población, CHC define una búsqueda robusta y efectiva.

Los mejores resultados de las metaheurísticas estudiadas se ubicaron por encima del 80% respecto al óptimo del problema para las instancias de mayor tamaño. CHC siempre superó el 95% respecto al óptimo, mientras que AG y los híbridos alcanzaron resultados en el entorno del 90-95%. Estos resultados muestran que aún utilizando operadores simples, que no incorporan información específica alguna respecto al problema, las técnicas evolutivas son capaces de alcanzar soluciones de calidad elevada para el conjunto de problemas estudiados. Si bien las instancias no son representativas de casos reales, ni presentan una complejidad característica de otros problemas, la alta eficacia obtenida alienta a considerar como precisos los resultados previamente obtenidos por estas técnicas para instancias generadas aleatoriamente y casos realistas.

Aunque la evaluación del desempeño no consistió un objetivo del trabajo, se analizaron los tiempos de ejecución de los algoritmos. Los tiempos promedio de las 10 ejecuciones independientes para resolver la instancia más compleja abordada (13 réplicas) se presentan en la Tabla 8. Las metaheurísticas basadas en trayectoria resuelven rápidamente el problema, al trabajar con una única solución, mientras que las basadas en población demoran un tiempo entre tres y cuatro veces mayor. La hibridación del mecanismo del AG con las técnicas basadas en trayectoria incrementa de modo notorio los tiempos de ejecución del algoritmo para resolver las instancias consideradas.

problema	*CHC*	*GA*	*SA*	*VNS*	*GA+VNS*	*GA+SA*
grafo_1	10.0	14.7	3.2	5.9	25.7	21.3
grafo_2	15.2	21.9	5.2	9.0	35.7	29.1
grafo_3	13.4	19.8	4.5	8.6	32.5	26.6
grafo_4	14.7	21.0	4.9	8.8	34.9	28.9

Tabla 8: Tiempos de ejecución promedio (minutos).

8 Conclusiones y Trabajo Futuro

Este trabajo analiza la calidad de resultados obtenidos por técnicas metaheurísticas al resolver instancias del GSP con soluciones óptimas conocidas. Las metaheurísticas trabajan con el mayor nivel de simplicidad algorítmica, sin incluir información dependiente del problema, salvo la función a optimizar. Aunque las instancias no son realistas, ni pueden considerarse con la complejidad característica de otros problemas, la alta calidad de los resultados obtenidos contribuye a potenciar los resultados previamente obtenidos por estas técnicas para instancias generadas aleatoriamente y casos de la vida real.

Las técnicas basadas en población se mostraron muy efectivas para la resolución de las cuatro instancias estudiadas. Con los sencillos operadores propuestos, las técnicas basadas en trayectoria son menos competitivas. El algoritmo CHC confirmó resultados previos para instancias aleatorias, mostrándose como una técnica muy prometedora para la resolver el GSP. Este método logra resultados con una distancia relativa al óptimo menor que el 5% para las instancias más complejas estudiadas. Adicionalmente, alcanza individuos muy adaptados en pocas generaciones y su tiempo de ejecución total es moderado. La hibridación permite mejorar los resultados, al introducir una nueva fuente de diversidad en el mecanismo evolutivo. Como contrapartida, la inclusión de operadores internos más complejos incrementa notoriamente los tiempos de ejecución de los algoritmos.

El estudio ha dejado aspectos inconclusos que plantean líneas de trabajo futuro. Como primer paso, debe afrontarse el estudio teórico que permita diseñar instancias de prueba representativas de problemas realistas. Respecto a los algoritmos, debe estudiarse la contribución de los modelos de hibridación, considerando el costo computacional que demandan. Asimismo, debe analizarse la influencia del operador de reinicialización en los resultados del algoritmo CHC. El trabajo no afrontó aspectos referentes al desempeño computacional de los algoritmos. En este sentido, las implementaciones pueden optimizarse para ser capaces de abordar instancias más complejas del GSP, eventualmente aplicando técnicas de procesamiento de alta performance.

Referencias bibliográficas

[1] E. Alba, C. Cotta. *Optimización en entornos geográficamente distribuidos. Proyecto MALLBA*. Primer Congreso Español de Algoritmos Evolutivos y Bioinspirados, pp. 38–45, 2002.

[2] L. Davis. *Handbook of Genetic Algorithms*. Van Nostrand Reinhold, New York, 1991.

[3] H. Esbensen. *Computing Near-Optimal Solutions to the Steiner Problem in a Graph Using a Genetic Algorithm*. Networks 26, pp. 173-185, 1995.

[4] L. Eshelman. *The CHC adaptive search algorithm: How to have safe search when engaging in nontraditional genetic recombination*. Foundations of Genetic Algorithms, pp. 265-283, 1991.

[5] L. Ford, D. Fulkerson. *Flows in Networks*. Princeton University Press, Princeton, 1962.

[6] V. Kahn, P. Crescenzi. *A compendium of NP optimization problems*, Disponible en línea en `http://www.nada.kth.se/theory/problemlist.html`. Consultado julio 2005.

[7] R. Karp. *Reducibility among combinatorial problems*. Complexity of Computer Communications, pp. 85-103, Plenum Press, 1972.

[8] S. Kirkpatrik, C. Gelatt, M. Vecchi. *Optimization by simulated annealing*, Science, pp. 671–680, 1983.

[9] S. Nesmachnow, H. Cancela, E. Alba. *Técnicas Evolutivas Aplicadas al Diseño de Redes de Comunicaciones Confiables*. Actas del Tercer Congreso Español de Metaheurísticas, Algoritmos Evolutivos y Bioinspirados (MAEB '04), pp. 388-395, 2004.

[10] S. Nesmachnow. *Algoritmos Genéticos Paralelos y su Aplicación al Diseño de Redes de Comunicaciones Confiables*. Tesis de Maestría en Informática, PEDECIBA, Uruguay, 2004.

[11] N. Mladenovic, P. Hansen. *Variable Neighborhood Search: principals and Applications*. European Journal Operational Research, 130, pp. 449-467, 1999.