Optimización de Algoritmos Genéticos en el Procesamiento de Árboles de Steiner

Mário Jesus†, Sérgio Jesus‡, Alberto Márquez§
†Universidade do Algarve (EST). Dep. Eng. Civil. 8000 Faro - PORTUGAL
‡Universidade do Algarve (UCEH). Dep. Eng. de Sistemas e Computação. 8000 Faro - PORTUGAL
§Universidad de Sevilla (FIE). Dep. Matemática Aplicada I. 41012 Sevilla - ESPAÑA

5/mayo/99

Resumen

Las características de los Algoritmos Genéticos parecen ser muy apropiadas para poder abordar problemas de resolución compleja y problemática como son las Árboles de Steiner. Al introducir algunas heurísticas se pueden mejorar aún más esas propiedades.

Palabras Clave: árbol de Steiner, puntos de Steiner, algoritmos genéticos, epístasis.

1 Introducción

El problema de los *árboles de Steiner* (ASt) en el plano Euclídeo ha llamado la atención de la comunidad científica de diferentes áreas, originando una cantidad apreciable de información sobre el tema [1] [2].

Su reputación tiene origen en la facilidad con que se puede plantear y comprender, aunque, desde luego, su resolución no tenga las mismas características, ya que en [3] se demuestra que el problema de los ASt es NP-duro, luego no tiene un tratamiento en tiempo polinomial y además posee un consumo muy elevado de procesamiento. Esencialmente es un problema de optimización combinatoria que suele producir muchas respuestas óptimas para un determinado número de puntos iniciales.

De una forma simple uno se puede plantear el problema de la siguiente forma: considérese la existencia de unos cuantos puntos que es necesario conectar de tal forma que se obtenga la menor distancia total aunque para tal fin sea necesario añadir más puntos a los iniciales. Éstos puntos auxiliares reciben el nombre de *puntos de Steiner* (Stp) y el resultado final obtenido será un *árbol de Steiner* para ese problema, [4] [5] (ver Figura 1).

De una forma muy general se pueden dividir las investigaciones hechas sobre el tema, en dos grandes grupos: por una parte están las que usan métodos o heurísticas determinísticas, como [6] [7] [8] [9] [10] [11]. Estos métodos, suelen presentar limitaciones y sobre todo se dedican a configuraciones muy especiales de puntos o a casos muy particulares del problema

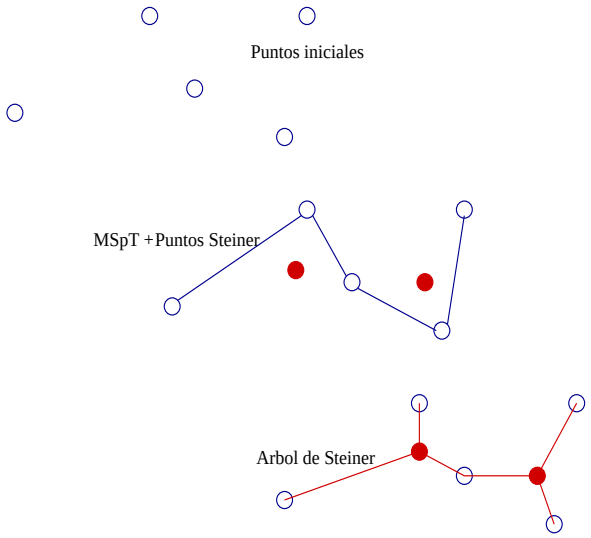

Figura 1: Obtención de un *árbol de Steiner*.

general de los ASt. Si se trata de resolver situaciones más generales, se han diseñado algoritmos de cálculo aproximado como aquellos que incluyen procesos aleatorios evolutivos, [12] [13] [14]. En [12] es señalado que los *algoritmos genéticos* (AGs) "puros"también presentan una convergencia extremadamente lenta, con lo cual es necesario la inclusión de ciertas heurísticas que hagan que mejoren su convergencia.

En éste trabajo se utilizan técnicas del segundo grupo, más precisamente el procesamiento evolutivo garantizado por la aplicación de AGs; en dichos algoritmos, se ha incluido cierta heurística, que mejora la convergencia del algoritmo y que resuelve algunos de los problemas que implementaciones anteriores presentaban. En cualquier caso, quisiéramos señalar que nuestro trabajo aún se encuentra en una fase inicial, aunque ya podemos presentar algunos resultados concretos.

En la siguiente sección señalaremos los principales problemas observados hasta el momento en la implementación de *algoritmos genéticos* para la resolución de ASt y las soluciones que proponemos para dichos problemas. Tras una sección dedicada a mostrar algunos de los resultados obtenidos hasta la fecha, terminaremos señalando las posibles líneas de continuación.

2 Característica del problema

El procesamiento evolutivo es caracterizado por la utilización de princípios y métodos naturales en sistemas que representan potenciales soluciones de un problema. Un programa evolutivo es un algoritmo probabilístico que mantiene un conjunto de respuestas de forma cíclica, [15].

En particular los *algoritmos genéticos*, [16], usan una analogía directa con el comportamiento de las especies en la naturaleza, donde prevalece el más fuerte. Utilizando una representación adecuada para el problema, y mediante la actuación de unos cuantos operadores "genéticos" (selección, cruce y mutación) se logra obtener una buena aproximación para la respuesta final.

Una primera aproximación para el problema de los ASt lo proporciona el uso de los "minimum spanning trees" (MSpT), que son procesados con la entrada de los puntos iniciales, en una primera fase. El punto crucial es el cálculo de los *puntos de Steiner*. Dicho cálculo trata de minimizar una cierta función coste que no es más que la longitud del MSpT de una nube de puntos (la de los puntos originales más los Stp obtenidos) [1] [2]. Es en este punto donde juegan un papel fundamental los AGs ya que han probado su eficacia para la obtención de extremos relativos de funciones cuando los métodos tradicionales del análisis han fallado.

Sin embargo, como hemos señalado anteriormente, surgen algunas dificultades al aplicar estos métodos para la resolución del problema de los ASt. Las soluciones que hemos propuesto para resolver estos problemas son los que caracterizan nuestra aportación sobre otras, así que señalamos aquí estos problemas y pasaremos más adelante a exponer nuestras propuestas:

Codificación Cada elemento en el sistema que va a evolucionar bajo los AGs, es con frecuencia designado por cromosoma y posee un conjunto de parámetros que identifican una potencial respuesta. Un cromosoma deberá poseer la información identificadora de los *puntos de Steiner* necesarios para obtener la respuesta final, una vez añadidos a los terminales o puntos iniciales, que son fijos para cada caso. Esta codificación debe poder recoger cualquier posible solución al problema y uno de los inconvenientes que anteriores codificaciones utilizadas (basadas en las coordenadas cartesianas de los Stp) es lo que se ha llamado como *epístasis* en la literatura [15] [17] [18]. A grosso modo, la *epístasis* de una codificación implica que diversos conjuntos de bites asociados a un mínimo local (pero no global) son arrastrados por el algoritmo impidiendo obtener el mínimo global deseado. Que sepamos, este es el primer trabajo en el que se propone una solución para este problema, que veremos en la próxima subsección, en el caso de los ASt.

Aceleración de la convergencia Como se ha señalado, si al proceso general se le deja totalmente libre dentro de un campo aleatorio, la convergencia del algoritmo es extremadamente lenta. Es por tanto necesario incluir ciertas heurísticas relacionadas con la geometría concreta del problema y que permitan obtener soluciones válidas en periodos razonables de tiempo.

Fiabilidad de las soluciones En los mecanismos evolutivos existe la posibilidad que el algoritmo de por buenas soluciones subóptimas distintas de las buscadas. De nuevo diversas estrategias permiten asegurar que las soluciones obtenidas son las adecuadas.

2.1 Codificación

Una primera característica que se ha utilizado siempre para la codificación es que un ASt para N terminales, posee como máximo $N-2$ *puntos de Steiner* [1].

Una particularidad de los AGs consiste en la iniciación del conjunto de cromosomas, designado por población, hecha de una forma totalmente aleatória. Existe la posibilidad de que algunos cromosomas se sitúen fuera de la envolvente convexa que limita los puntos iniciales, presentando, de esa forma, una perdida de esfuerzo computacional.

Utilizando la propiedad que la envolvente convexa de los terminales es la misma que encierra el *árbol de Steiner* y el siguiente teorema:

Teorema 1 *[19] Sean $P_1, \ldots, P_n$ puntos de $\Re^2$. O conjunto de todas las combinaciones lineales*

$$x_1 P_1 + \ldots + x_n P_n$$

con $0 \leq x_i \leq 1$ y $x_1 + \ldots + x_n = 1$, es un conjunto convexo.

Se puede limitar la generación de puntos al espacio interior de la envolvente convexa, si cada punto fuera expresado como una combinación lineal de los puntos que forman la envolvente de los puntos iniciales. Además, con esta representación conseguimos eliminar el problema de la *epístasis* ya que es probado en [17] que la representación mediante combinaciones lineales como la que aquí proponemos elimina al mínimo dicho inconveniente.

2.2 Heurísticas de optimización

Hemos considerado dos tipos de heurísticas que poseen objetivos distintos: una para acelerar la convergencia del algoritmo y la otra para aumentar la fiabilidad de la respuesta obtenida, solucionando así los dos problemas señalados anteriormente.

2.2.1 Área de supervivencia

La idea principal que se encuentra tras este método consiste en delimitar, para cada punto, un área mínima donde él ejerce su influencia sin oposición. La eliminación directa de los demás puntos que puedan encontrarse en dicha región de supervivencia de un cromosoma se justifica técnicamente porque dos *puntos de Steiner* muy "próximos"no aportan beneficio directo en el contexto global del ASt.

El concepto que queda por fijar es la cuantificación de muy "próximos"que en nuestro caso ha sido calculado de acuerdo con el área ocupada por los puntos iniciales, la población y la distribución que tengan los cromosomas dentro del espacio de búsqueda.

2.2.2 Rectificación de los pre-árboles de Steiner

Una vez obtenido un ASt previo, se repasan los *puntos de Steiner* obtenidos y se analizan sus valencias. El teorema fundamental que se utiliza en este paso es:

Teorema 2 *[1]*

1. *En un árbol de Steiner dos lados no podrán hallarse con un ángulo inferior a* 120°.
2. *Un árbol de Steiner no posee cruce de aristas.*
3. *Cada punto de Steiner en un ASt tiene exactamente grado tres.*

La heurística desarrollada es compuesta por las siguientes fases:

- Los vértices que presentan valencia uno, son eliminados.
- Aquellos con valencia dos, también son eliminados, y sus dos vecinos son unidos mediante una nueva arista (Figura 2).

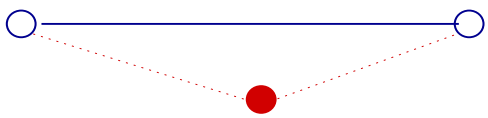

Figura 2: Elininación de un vértice de grado dos.

- En el caso de vértices de valencia tres, el punto es sustituido por uno que presente las características señaladas en anterior teorema por la técnica geométrica directa de Torricelli (Figura 3).

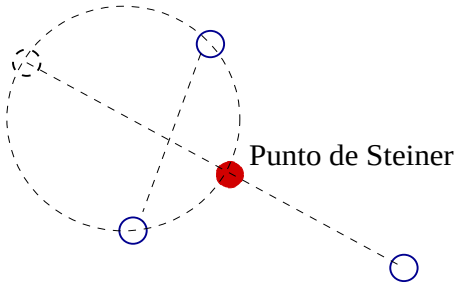

Figura 3: Método de Torricelli.

Es de señalar que aunque se han presentado anteriormente heurísticas que conseguían los mismos resultados que la nuestra [12], la aquí descrita es la primera que utiliza métodos directos y por tanto computables en tiempo constante y no iterativos.

3 Modelo computacional y resultados obtenidos

Utilizando librerías especializadas [20] [21] se ha desarrollado un modelo (cGa) para permitir el estudio del comportamiento de los AGs frente a un problema como éste y al mismo tiempo utilizar distintas heurísticas señaladas y otras posibles futuras para acelerar su tarea de optimización y aumentar su precisión en la respuesta final. En el esquema representado

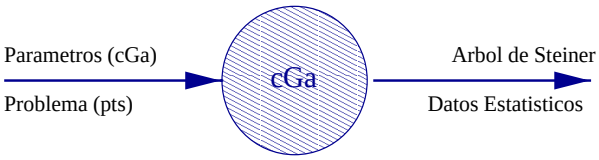

Figura 4: Esquema general de funcionamento del cGa.

en la Figura 4 se puede apreciar la forma de funcionamiento del modelo. El cGa ha sido desarrollado en Linux y el ejecutable ha sido obtenido con librerías C++ de GNU.

Para las pruebas realizadas se han utilizado mallas regulares de puntos, ya que sus *árboles de Steiner* son conocidos [22]. Además se ha de señalar que dichos conjuntos de puntos presentan muchas posibles soluciones óptimas, con lo cual se puede comprobar la robustez de los métodos propuestos.

En la Tabla 1 se resúmen los valores obtenidos de la ejecución del modelo computacional en un ordenador Pentium Pro (200Mhz), teniento como entrada, mallas de puntos de 3×4, 3×5, 3×6 y 4×4. Se hicierón 50 ejecuciones del modelo con cada tipo de malla,

Tipo	$\|MSpT\|$	$\|ASt\|$	$Generacion_{opt}$	$Tiempo_{opt}$	$\|ASt\|_{opt}$
3×4	22	20.3922	593	$61.9seg$	20.4027
3×5	28	25.8563	1169	$173.5seg$	25.8768
4×4	30	27.3204	2145	$242.1seg$	27.6386
3×6	34	31.3204	2200	$453.6seg$	31.3710

Tabla 1: Tabla de resultados.

utilizándose una población con 80 cromosomas con tasas de cruzamiento y mutación de 0.80 y 0.05, respectivamente. El número de generaciones utilizadas fue distinto, 2000 para los dos primeros casos y 3000 para los otros dos.

Las columnas $|MSpT|$ y $|ASt|$ informan los valores referenciales de los "minimum spanning tree" y los *árboles de Steiner* para cada malla permitindo una fácil comparación con el valor médio calculado por el modelo en cada caso, como está referido en la última columna de la tabla.

Se han utilizado un excesivo número de generaciones para garantizar que el algoritmo evolucione hasta obtener un resultado suficientemente estable. Las columnas $Generacion_{opt}$ y $Tiempo_{opt}$ indican el número de generaciones necesarias para obtener el óptimo de la ejecución y el tiempo tardado, considerando un promedio de 50 ejecuciones.

Las Figuras 5 y 6 presentan gráficos de ejecuciones típicas de los AGs, teniendo en cuenta el número de generaciones utilizado en los casos presentados.

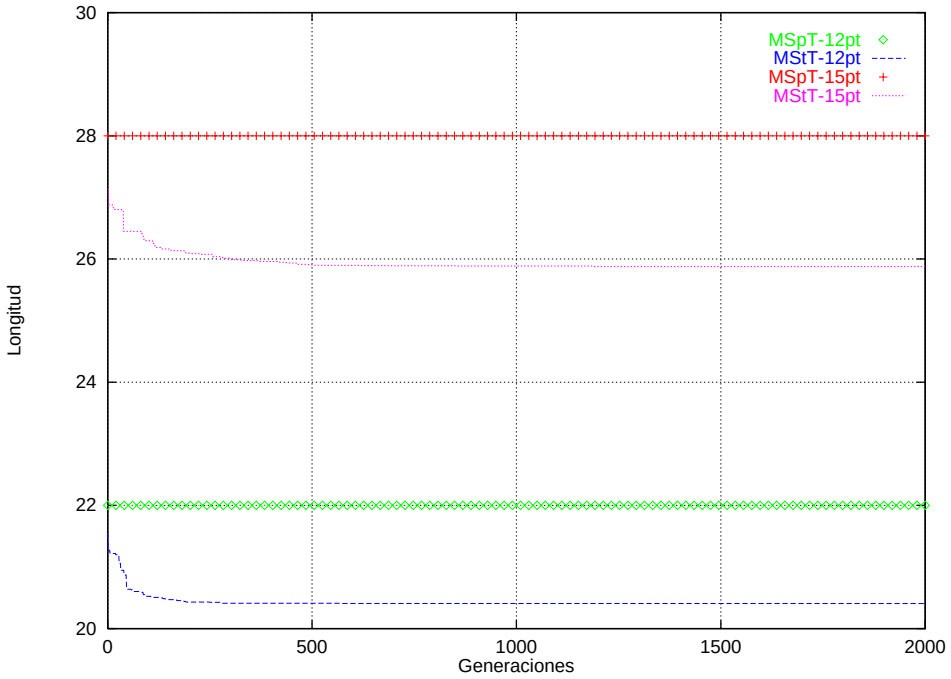

Figura 5: Ejecución del cGa en los casos de 3×4 y 3×5.

4 Conclusiones

El modelo aquí presentado, parece prometedor y nos permite suponer que la continuación del trabajo en la misma línea que la actual puede llevar a conseguir métodos computacionales prácticos para encontrar *árboles de Steiner* en situaciones totalmente generales con gran número de puntos iniciales. Además por la propia naturaleza de los *algoritmos genéticos*, el método aquí presentado puede ser paralelizado muy eficazmente, consiguiéndose así una respuesta en tiempo más breve.

Tendremos que seguir aplicando test para la comprobación de nuestro algoritmo. Ampliar el número inicial de puntos y cambiar su disposición en el plano pueden ser pasos inmediatos para el modelo.

Creemos que sería interesante continuar mediante:

- Regularización de mallas.

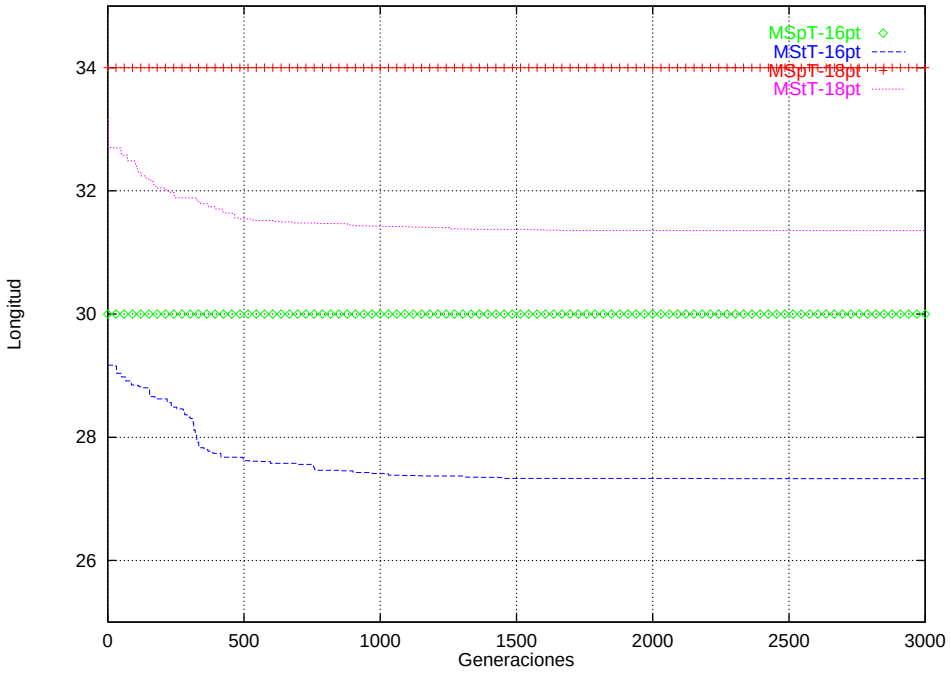

Figura 6: Ejecución del cGa en los casos de 3×6 y 4×4.

- Control de la topología del ASt final.
- Generación de "quasi-ASt".
- Generación de ASt incompletos.
- Análisis estadístico de las salidas obtenidas en función de la población inicial.

Referencias

[1] F. Hwang, D. Richards and P. Winter, *The Steiner problem*, Elsevier Science Publishers B.V., 1992.

[2] Dietmar Cieslik, *Steiner minimal trees*, Kluwer Academic Publishers, 1998.

[3] M. Garey, R. Graham and D. Johnson, "The complexity of computing Steiner minimal trees", *SIAM Journal Applied Mathematics*, Vol. 32, pp. 835-850, 1997.

[4] F. Preparata and M. Shamos, *"Computational geometry - an introduction"*, Springer-Verlag, 1985.

[5] J. Goodman and J. O'Rourke, *"Handbook of discrete and computational geometry"*, CRC Press, 1997.

[6] L. Kou, G. Markowsky and L. Berman, "A fast algorithm for Steiner trees", *Acta Informatica*, Vol. 15, pp. 141-145, 1981.

[7] V. Rayward-Smith and A. Clare, "On finding Steiner vertices", *Networks*, Vol. 16, pp. 283-294, 1986.

[8] J. Baesley, "An SST-based algorithm for the Steiner problem in graphs", *Networks*, Vol. 19, pp. 1-16, 1989.

[9] J. Ganley and J. Cohoon, "Rectilinear Steiner trees on a checkerboard", *ACM Tansactions on Design Automation of Electronic Systems*, Vol. 1, pp. 512-522, 1996.

[10] J. Ganley and J. Cohoon, "Improved computation of optimal rectilinear Steiner minimal trees", *International Journal of Computational Geometry & Application*, Vol. 7, pp.457-472, 1997.

[11] J. Weng, "Expansion of linear Steiner trees", *Algorithmica*, Vol. 19, pp. 318-330, 1997.

[12] J. Hesser, R. Männer and O. Stucky, "Optimization of Steiner trees using genetic algorithms", *Proceedings of ICGA'89*, 1989.

[13] A. Kapsalis, V. Rayward-Smith and G. Smith, "Solving the graphical Steiner tree problem using genetic algorithms", *Journal of Operational Research Society*, Vol. 44, pp. 397-406, 1993.

[14] H. Esbensen "Finding (near)-optimal Steiner trees in large graphs" *Proceedings of ICGA'95*, 1995.

[15] Z. Michalewicz, *Genetic algorithms + data structures =evolution programs*, Springer-Verlag, 1996.

[16] J. Holland, *Adaptation in natural and artificial systems*, MIT press, 1992.

[17] M. Iglesias et al., "Epistasis and unitation", *preprint*.

[18] M. Iglesias et al., "Multary epistasis", *preprint*.

[19] S. Lang, *Linear algebra*, Springer-Verlag, 1987.

[20] M. Wall, *GAlib: a C++ library of genetic algorithm components, (ver. 2.4)*, Massachusetts Institute of Technology, 1996.

[21] K. Mehlhorn et al., *LEDA: library of efficient data types and algorithms, (ver. 3.7)*, Max-Plank-Institut für Informatik, 1998.

[22] M. Brazil et al., "Minimal Steiner trees for rectangular arrays of lattice points", *Journal of Combinatorial Theory*, Series A 79, pp. 181-208, 1997.