

Algoritmo multithreading para el problema de Steiner en grafos

Gerardo Ares

CeCal, Facultad de Ingeniería.
Montevideo, Uruguay
gares@fing.edu.uy

Sergio Nesmachnow

CeCal, Facultad de Ingeniería.
Montevideo, Uruguay
sergion@fing.edu.uy

Resumen

Este artículo presenta una implementación de la metaheurística SN para problema del árbol de Steiner utilizando una técnica de programación multithreading. Describe las decisiones de diseño utilizadas y los resultados obtenidos sobre un conjunto de problemas de prueba.

Palabras clave

Problema de Steiner, STP, multithreading, heurística SN.

INTRODUCCIÓN

El continuo crecimiento en el tamaño de los problemas de diseño de redes de comunicaciones ha conducido a los investigadores a proponer alternativas a los enfoques exactos tradicionales para resolver problemas complejos. En este contexto, varias heurísticas se han aplicado al problema de diseño de redes de comunicaciones confiables, para la resolución de los complejos problemas de optimización subyacentes.

Considerando una red de comunicaciones con nodos distinguidos denominados terminales, el Problema del Árbol de Steiner (Steiner Tree Problem – STP) refiere al diseño de una subred de mínimo costo, que garantice la conectividad entre pares de nodos terminales. El STP forma parte de los problemas clásicos de optimización combinatoria y diversos algoritmos han sido propuestos para su resolución.

El método SN [2] propone utilizar una técnica de subdivisión del problema de optimización original en varios problemas de decisión que se resuelven heurísticamente para las dos alternativas posibles, correspondientes a los resultados "SI" y "NO" del problema de decisión. De acuerdo a su formulación, el método es altamente paralelizable, siendo posible atacar concurrentemente los diversos subproblemas generados.

Este trabajo describe los primeros resultados en la resolución del Problema del árbol de Steiner utilizando una implementación paralela sobre memoria compartida, utilizando la técnica de multithreading. El resto del artículo se organiza del modo que se describe a continuación: la sección 2 brinda una presentación del Problema del árbol de Steiner, la sección 3 introduce la heurística SN y su aplicación al problema del árbol de Steiner. Los detalles de diseño e implementación del algoritmo paralelo se ofrecen en la sección 4. La sección 5 resume los resultados obtenidos en la resolución del STP y la sección 6 ofrece las conclusiones y líneas de trabajo actual y futuro.

EL PROBLEMA DEL ÁRBOL DE STEINER

Dado un grafo conexo con aristas de costos no negativos y un subconjunto distinguido del conjunto de nodos, llamados nodos terminales, el problema del árbol de Steiner consiste en hallar el árbol de costo mínimo que una a todos los nodos terminales. Aquellos nodos no pertenecientes al conjunto de terminales que forman parte del árbol solución son llamados nodos de Steiner. La formulación matemática del problema puede encontrarse en el compendio de Kahn y Crescenzi [2].

En 1973 Karp [3] demostró que el problema del árbol de Steiner es NP-Completo. Desde entonces, diferentes enfoques de métodos heurísticos han sido aplicados para obtener buenas soluciones del problema en tiempos razonables.

METAHEURÍSTICA SN

La metaheurística SN, propuesta por Urrutia y Loiseau [5], plantea dividir un problema de optimización en varios subproblemas de decisión de resolución más simple que el problema original. Utilizando una heurística específica, se obtiene una solución y un grado de confianza para cada uno de los subproblemas. El problema original se transforma en un problema equivalente tomando en cuenta la decisión del subproblema cuya solución tuvo el mayor grado de confianza. La metaheurística se aplica recursivamente a este nuevo problema, mientras sea posible aplicar la división en subproblemas.

El esquema general de la metaheurística se presenta en la Figura 1.

*Mientras pueda dividirse el problema en n subproblemas
equivalentes distintos hacer*
Para cada i desde 1 hasta n hacer
Resolver heurísticamente el problema
para i obteniendo s_i y gc_i
Fin Para
Modificar el problema utilizando la información
del subproblema i según s_i
Fin Mientras

Figura 1. Algoritmo genérico de la metaheurística SN

Dado un elemento del problema, existirán dos subproblemas de decisión complementarios que lo involucren en cada paso. Estos corresponderán a casos "SI" y "NO" para los cuales se propone un esquema constructivo para la metaheurística resumido en:

- Obtener una solución constructiva al subproblema asumiendo que la decisión es SI.
- Obtener una solución constructiva al subproblema asumiendo que la decisión es NO.

Evaluando la magnitud a optimizar, se toma como solución (s_i) para el subproblema i , la mejor de las dos soluciones constructivas de acuerdo al criterio de optimización utilizado. El grado de confianza se computa como la diferencia entre los valores de las soluciones.

Sobre las transformaciones que se realizan al final de cada iteración se exigen 3 condiciones :

- La transformación tiene que generar un problema que pueda ser dividido en menos subproblemas que en la instancia anterior, de modo de reducir la complejidad a medida que se aplica la metaheurística.
- La solución óptima para el problema transformado es igual o peor que para el problema original.
- Si la solución para el subproblema i (s_i) es correcta, entonces la solución óptima para el problema transformado es igual al problema original.

Aplicación de SN al Problema STP

Los propios creadores de la metaheurística SN han propuesto aplicarla a la resolución del problema del árbol de Steiner [5].

Para el problema del árbol de Steiner, la decisión a tomar en cada iteración de la metaheurística será si un nodo no terminal pertenecerá o no a la solución del problema. Se divide el problema original en subproblemas; cada uno de ellos tomará un nodo no terminal y buscará dos soluciones: una que contenga el nodo (“caso SI”) y otra en que no lo contenga (“caso NO”). El grado de confianza para cada subproblema estará dado por la diferencia en valor absoluto de los costos de los árboles solución para cada caso (gc_i).

Una vez evaluadas todos los subproblemas se escoge el nodo que generó el mayor grado de confianza. El problema se transforma según la decisión que produjo la mejor solución. Si la decisión corresponde a un “caso SI”, se agrega el nodo al conjunto de terminales del grafo. Si por el contrario la mejor solución fue para el “caso NO”, entonces se elimina del grafo el nodo en cuestión.

La metaheurística llegará a su fin cuando haya procesado la totalidad de los nodos no terminales.

ALGORITMO MULTITHREADING PARA EL PROBLEMA STP

La metaheurística SN es paralelizable en un alto grado, dado que los subproblemas a ser resueltos en cada iteración son independientes entre sí. Una implementación paralela reducirá los tiempos de ejecución total del algoritmo utilizando los recursos computacionales existentes y permitirá atacar instancias de mayor dimensión y complejidad del problema de Steiner.

Nuestra propuesta consiste en aplicar el criterio de descomposición de dominio, asignando diferentes secciones

del espacio de búsqueda a diferentes procesadores, tal como se describe en el diagrama de la Figura 2. Un nodo maestro se encarga de la división de tareas entre procesos esclavos. Una vez que todos los subproblemas fueron resueltos, el nodo maestro centraliza la búsqueda del mejor grado de confianza.

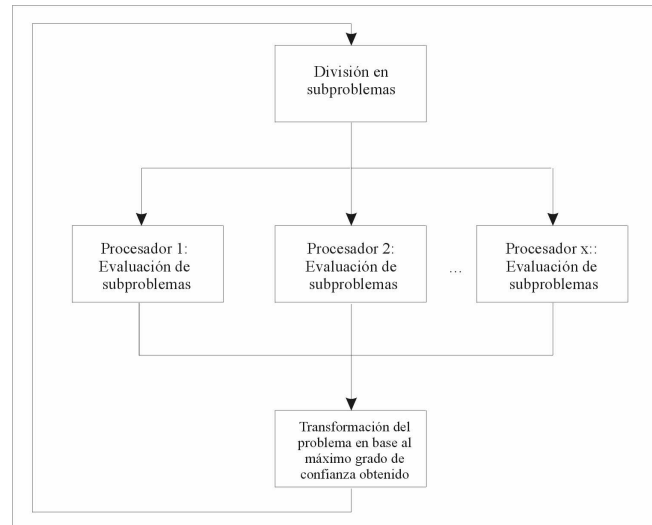


Figura 2. Esquema de paralelización de la metaheurística SN.

Se propuso trabajar sobre una arquitectura de memoria compartida, implementando los procesos esclavos mediante threads. Los detalles de la implementación paralela se ofrecen en la subsección siguiente.

El algoritmo utiliza como heurística para la generación de las soluciones en los casos “SI” y “NO” la construcción de un árbol de cubrimiento para los terminales partiendo de un nodo terminal elegido al azar. En cada paso se procesa el nodo más cercano al árbol de cubrimiento. Si este nodo resulta ser terminal o resulta ser el nodo no terminal de un subproblema “caso SI”, se agregan al árbol todos los nodos (terminales y no terminales) pertenecientes al camino que une el nodo encontrado con el árbol.

En el “caso SI”, el nodo del subproblema es considerado como si fuera un nodo terminal, incluyéndolo al árbol que genera la búsqueda. En el caso NO, se utiliza la misma heurística constructiva pero sobre el grafo resultado de eliminar el nodo del subproblema, generando una solución en donde no participará el nodo en cuestión.

La fase iterativa del algoritmo llegará a su fin cuando haya procesado la totalidad de los nodos no terminales. Se obtiene entonces un subgrafo compuesto por los nodos terminales y aquellos no terminales que se decidió incluir en las etapas iterativas del algoritmo. Sobre este subgrafo se halla el árbol de cubrimiento de costo mínimo utilizando el algoritmo de Prim [4].

El algoritmo utilizado para el problema STP se presenta en la Figura 3.

Mientras la cantidad de nodos terminales es menor que la cantidad de nodos del grafo hacer
Para cada nodo no terminal hacer
Resolver heurísticamente el problema
para “caso SI” y “caso NO”
obteniendo s_i y gc_i
Fin Para
Obtener el mayor gc_i y transformar el problema
según la mejor solución entre el “caso SI” y
“caso NO”.
Fin Mientras

Figura 3. SN aplicado a STP

Ciertos casos particulares han sido contemplados en la etapa de diseño para mejorar la eficiencia del algoritmo:

- El caso en que no se encuentre solución al evaluar algún subproblema. Como ejemplo, en un subproblema “caso NO” el grafo podría quedar desconexo si el nodo en estudio no se incluye en la solución. Ante tal situación se considera al nodo como parte de los nodos terminales, ya que necesariamente deberá formar parte de la solución.
- El caso en que el problema de decisión resultante sea tal que divida al grafo en dos componentes conexas, una compuesta por terminales y otra por no terminales. El algoritmo toma ventaja de esta situación descartando la componente conexa sobre la cual no hay requerimientos de conexión.

Detalles de implementación del algoritmo

Se construyó una implementación genérica de la metaheurística y luego se creó una instancia para el STP. La metaheurística se implementó en el lenguaje de programación C++. Para la paralelización de la resolución de los subproblemas se usó la biblioteca *pthread*.

El esquema de clases que forman la estructura de la implementación es la siguiente:

- **SN** Contiene la iteración general de la metaheurística.
- **SNTTable** Tabla en la cual se almacena la información generada por la resolución de los subproblemas. Cada elemento de esta tabla es un objeto que implementa la clase abstracta **SNData**.
- **SNData** Clase abstracta en donde se encuentra la información básica generada en la resolución de los subproblemas (gc_i y s_i).
- **ThreadI** Pool de threads especializados en resolver la heurística constructiva.
- **SPH** Clase abstracta. Su utilidad es definir una interfaz que debe cumplir una instancia de la heurística constructiva.
- **Solution** Clase abstracta. Define una interfaz que debe cumplir la solución al problema SN.

Las clases que instancian el problema del árbol de Steiner son:

- **SNDataInstance**. Extiende la clase **SNData**. Contiene, además de los datos de **SNData**, las soluciones para el “caso SI” y “caso NO” del nodo del subproblema.
- **SPHInstance**. Extiende la clase **SPH**. Instancia que contiene la resolución de la heurística constructiva. Conjuntamente contiene el grafo del problema e implementa las operaciones que realizan la transformación del problema.
- **Edge**. Clase que representa una arista del grafo.
- **Vertex**. Clase que representa un nodo del grafo.
- **SetEdges**. Extiende la clase **Solution**. Representa a un conjunto de aristas. La solución al STP constituye un ejemplo de un objeto de esta clase.
- **SetVertex**. Representa a un conjunto de nodos.
- **Graph**. Conjunto de nodos (**SetVertex**), aristas (**SetEdges**) y nodos terminales (**SetVertex**) que forman el problema.

Un esquema general de las clases se ofrece en la figura 4.

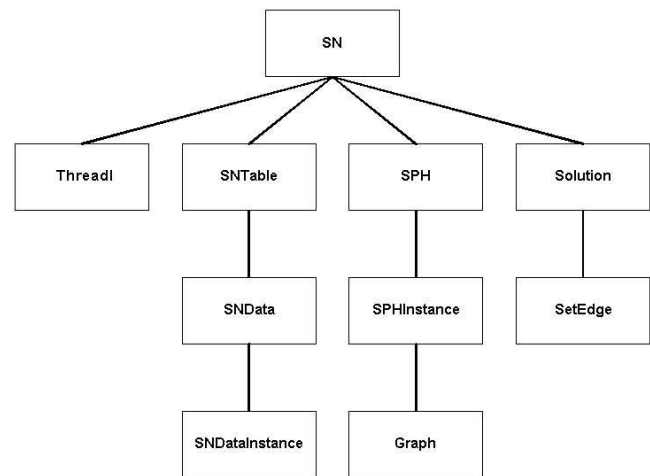


Figura 4. Diagrama de clases de la implementación multithreading para la metaheurística SN

La clase **SN** provee un método de nombre *run*, el cual tiene la estructura de iteración general presentada en la Figura 1. Para llevar a cabo la resolución de la metaheurística la clase **SN** define un pool de threads (clase **ThreadI**), una tabla en donde se almacena la información que genera la resolución de los subproblemas en cada iteración (clase **SNTTable**), un objeto que contiene la implementación de la heurística constructiva a aplicar en cada subproblema “caso SI” y “caso NO” (clase **SPH**), y la mejor solución encontrada hasta el momento (clase **Solution**).

La clase **ThreadI** representa al pool de threads. Estos se especializan en resolver únicamente los subproblemas. Son creados al comienzo de algoritmo y quedan a la espera de una señal que activará al máximo número de threads que deben ejecutar en paralelo. Este valor es determinado como el menor entre un número configurable (como ejemplo, la cantidad de procesadores disponibles) y la cantidad de nodos no terminales que tenga el problema.

Cada thread invoca a los métodos *solutionYes* y *solutionNo* de la clase *SPHInstance* pasando como parámetro la entrada en la tabla *SNTTable* correspondiente al nodo al cual se le aplica la resolución. Los métodos *solutionYes* y *solutionNo* cargan la entrada de la tabla con la información obtenida por la heurística constructiva para el “caso SI” y “caso NO” respectivamente. Posteriormente se determina la mejor resolución y se definen los valores gc_i y s_i para la entrada. Si existen más subproblemas a resolver, el thread selecciona uno y lo resuelve. Si no, queda a la espera para ser invocado en una próxima iteración de la metaheurística.

Finalizada la ejecución de todos los threads activados, el control pasa nuevamente al método *run*, quien busca el mejor grado de confianza obtenido en la iteración. Una vez hallado, se transforma el problema en un subproblema equivalente, aplicando los métodos *winYes* o *winNo* de la clase *SPHInstance* según corresponda al caso “SI” o “NO”.

RESULTADOS OBTENIDOS

La correctitud de la implementación multithreading fue verificada sobre un conjunto de problemas de prueba extraídos del repositorio OR-Library[1]. Las características de las instancias de prueba se presentan en la tabla 1, conjuntamente con los mejores resultados obtenidos aplicando el algoritmo en su versión serial.

Tabla 1. Resultados obtenidos

Grafo	Nodos	Aristas	Terminales	Óptimo	Resultado
c01	500	625	5	85	85
c02	500	625	10	144	144
c03	500	625	83	754	754
c04	500	625	125	1079	1080
c05	500	625	250	1579	1579
c06	500	1000	5	55	55
c07	500	1000	10	102	102
c08	500	1000	83	509	511
c09	500	1000	125	707	716
c10	500	1000	250	1093	1095
c11	500	2500	5	32	32
c12	500	2500	10	46	46
c16	500	12500	5	11	11
c17	500	12500	10	18	18
c18	500	12500	83	113	120

El proyecto se halla en etapa de desarrollo, por lo cual no se han realizado pruebas más exhaustivas del algoritmo implementado. Sobre el conjunto reducido de ejecuciones realizadas, los valores medios no se apartan significativamente de los mejores valores presentados en la tabla 4. Tomando en cuenta las pruebas de validación, el algoritmo ha presentado soluciones de buena calidad para el conjunto de problemas de prueba considerado.

Respecto a la eficiencia, el algoritmo presenta tiempos de ejecución serial superiores a los reportados por el algoritmo presentado en el trabajo [5]. Las pruebas de ejecución paralela han mostrado un speedup casi lineal cuando se utilizan dos procesadores, pero aún no se han completado pruebas más completas, utilizando un número mayor de elementos de procesamiento.

CONCLUSIONES Y TRABAJO FUTURO

Se ha diseñado e implementado un algoritmo multithreading para la resolución del problema del árbol de Steiner aplicando la metaheurística SN. Las pruebas de verificación del algoritmo han mostrado su correctitud y capacidad de alcanzar resultados de buena calidad para el conjunto de problemas de prueba presentado,

Los tiempos de ejecución son altos en comparación con otras implementaciones. Dado que el estudio de performance es uno de los objetivos del proyecto, en la actualidad se estudia la implementación de mejoras que se han sugerido para la resolución del STP utilizando SN [5] :

- Reducción del grafo del problema.
- Evaluar solamente un subconjunto de problemas en cada iteración, especificando condiciones que determinen los subproblemas a evaluar
- Mantener un repositorio de soluciones para tener respuestas rápidas al evaluar subproblemas.
- Buscar y agregar soluciones al repositorio de acuerdo a una probabilidad a determinar.

En la actualidad se trabaja con el objetivo de incorporar estas mejoras en el algoritmo multithreading.

Un estudio exhaustivo de la calidad de resultados y eficiencia del algoritmo multithreading se plantea como natural conclusión del proyecto. En este sentido, se plantea extender el conjunto de problemas de prueba incluyendo grafos de mayor dimensión, de modo de determinar la escalabilidad de la solución propuesta al aumentar el tamaño de los problemas.

REFERENCIAS

- [1] Beasley, J. *OR-Library:Distributing test problems by electronic mail*. Journal of the Operational Research Society 41 (1990), 1069 – 1072
- [2] Crescenzi P, Kahn V. *A compendium of NP optimization problems*. Disponible online: <http://www.nada.kth.se/~viggo/problemset/>. Consultada junio 2003.
- [3] Karp, R. *Reducibility among combinatorial problems*, Complexity of Computer Computations, R.E. Miller and J.W.Tatcher, eds., Plenum Press, New York (1972)
- [4] Prim, R. *Shortest Connection Networks and Some Generalizations*, Bell System. Technical Journal 36 (1957), 1389-1401.
- [5] Urrutia, S., Loiseau I. *A new metaheuristic and its application to the Steiner Problems in graph*. Tesis de Licenciatura, FCEyN, Univ. de Buenos Aires, 2001.