NÚMEROS
Revista de Didáctica de las Matemáticas
http://www.sinewton.org/numeros
ISSN: 1887-1984
Volumen 92, julio de 2016, páginas 35-48

Cómo la naturaleza nos muestra una solución a algunos problemas difíciles: el recocido simulado

Francisco Moreno Soto (IES Zurbarán, Badajoz. España)

Fecha de recepción: 01 de junio de 2012
Fecha de aceptación: 02 de marzo de 2016

Resumen En la naturaleza ocurren procesos que pueden ser simulados por los hombres. Algunos de estos procesos pueden tener usos muy interesantes desde un punto de vista matemático. Éste es el caso del algoritmo de recocido simulado que es el objeto de este artículo. Se trata de un algoritmo moderno de optimización global que surge a partir de la analogía con el proceso físico de recocido al que se someten los sólidos para obtener estados de mínima entropía. Las aplicaciones de este algoritmo son muchas y variadas, describiéndose en este trabajo la relativa al problema del viajante.

Palabras clave Naturaleza, heurísticas, algoritmo, recocido, problema del viajante

Title **How Nature shows a way to solve some difficult problems: Simulated Annealing**

Abstract There are optimization problems that are unmanageable using combinatorial methods. Simulated annealing is a heuristic optimization algorithm (thus named because it works by emulating the physical process whereby a solid is heated and then slowly cooled to get a minimum energy configuration). In this article it describes this algorithm on the traveling salesman problem.

Keywords Nature, simulation, algorithm, annealing, Traveling Salesman Problem

1. Introducción

Durante la segunda mitad del siglo XX, la optimización ha jugado un papel creciente en áreas tan diversas como la Ingeniería Eléctrica, la Investigación Operativa o las Ciencias de la Computación y de la Comunicación. Durante los años 50 y 60 la optimización lineal y no lineal, es decir, la búsqueda de un óptimo de una función de variables continuas, vieron un gran avance dando como resultado el algoritmo del Simplex para resolver problemas de Programación Lineal (Dantzing, 1963). A partir de los años 70, se obtuvieron potentes resultados en optimización combinatoria (Lin, 1973, pp. 498-516).

Por un *problema de optimización combinatoria* entendemos un problema de maximización o de minimización que está especificado por un conjunto de instancias. Una *instancia* de un problema de optimización combinatoria puede ser descrita como un par *(S, f)*, donde *S* denota el conjunto finito (o infinito numerable) de todas las posibles soluciones y *f* es una aplicación definida como:

$f : S \to R$. *S* se denomina espacio de soluciones o *conjunto factible* y *f función de costo*.

Resolver un problema de optimización combinatoria equivale a encontrar la "mejor" solución o solución óptima entre un conjunto finito o infinito numerable de soluciones posibles (Papadimitriou and Steiglizt, 1982). Así, en el caso de minimizar, el problema consiste en encontrar una solución $i_{\text{opt}} \in S$ tal que $f(i_{opt}) \leq f(i)$, $\forall i \in S$, mientras que para el caso de maximizar dicho problema consiste en encontrar una solución $i_{\text{opt}} \in S$ tal que $f(i_{opt}) \geq f(i)$, $\forall i \in S$. La solución $i_{\text{opt}} \in S$ se denomina solución global óptima o simplemente óptimo y puede no ser única; $f_{opt} = f(i_{opt})$ denota el costo óptimo y S_{opt} el conjunto de soluciones óptimas[1].

Muchos de estos problemas de optimización combinatoria, en especial los que tienen aplicación real, corresponden a la denominada clase de problemas NP-duros (los considerados como más difíciles en la clase NP). De esta clasificación se encarga la Teoría de la Complejidad Computacional. Esta es una rama de la teoría de la computación que se centra en la clasificación de los problemas computacionales de acuerdo a su dificultad inherente, y en la relación entre dichas clases de complejidad. Normalmente, esto se hace a un nivel muy teórico, aunque se trata de una cuestión clave para aplicar y solucionar problemas prácticos.

Un análisis superficial de algunas de estas clases podemos verlo aquí:

La clase P

La clase P contiene a aquellos problemas que son solubles en tiempo polinómico. Esta clase juega un papel importante en la teoría de la complejidad computacional debido a que, a grandes rasgos, P corresponde a la clase de problemas que, de manera realista, son solubles en una computadora

La clase NP

Muchas veces podemos evitar utilizar la fuerza bruta en los problemas para obtener soluciones en tiempo polinómico. Sin embargo, para algunos problemas esto no ha podido lograrse, es decir, no se conocen algoritmos que los resuelvan en tiempo polinómico. Quizás estos problemas tengan algoritmos en tiempo polinomial que se basan en principios por ahora desconocidos, o quizás estos problemas no pueden ser resueltos en tiempo polinómico, debido a que son "inherentemente difíciles".

La clase de complejidad NP consta de los problemas "verificables" en tiempo polinómico. Por verificable se entiende un problema tal que dado un certificado de solución (candidato a solución), se puede verificar que dicho certificado es correcto en un tiempo polinómico en el tamaño de la entrada. A los problemas en la clase NP usualmente se les llama *problemas NP*.

Un problema famoso dentro de los problemas NP es el problema del viajante: *un viajante tiene que pasar por unas determinadas ciudades siguiendo el camino más corto*. Aunque no se ha podido probar, se cree que no hay solución más eficaz que el enumerar todos los caminos posibles para ver cuál es el mejor (el número de caminos va creciendo como el factorial del número de ciudades).

Afortunadamente en la práctica, en muchas ocasiones, es suficiente encontrar una solución "no

[1] En este artículo sólo se considerarán problemas de minimización. Esto no supone ninguna pérdida de generalidad puesto que la maximización es equivalente a la minimización después de cambiar el signo de la función de costo

perfecta", pero no alejada de la óptima. Por ejemplo, si no se puede determinar el camino más corto, el viajante no se molestará mucho teniendo que seguir uno que le cueste sólo el 5% o 10% más. En muchos casos, se ha podido desarrollar algoritmos especiales para tratar un problema específico como el del viajante y llegar rápidamente a tales soluciones "buenas". Sin embargo, para tratarlos de una manera más general, los investigadores han recurrido a algoritmos aleatorios que en algunos casos fueron inspirados por fenómenos naturales. Antes de hablar de ellos vamos a introducir un nuevo campo: las heurísticas, y más allá de estas, las metaheurísticas. Se califica de *heurístico* a un procedimiento para el que se tiene un alto grado de confianza en que encuentra soluciones de alta calidad con un coste computacional razonable, aunque no se garantice su optimalidad o su factibilidad, o en algunos casos no se llegue a establecer lo cerca que se está de dicha situación. Se utiliza el calificativo heurístico en contraposición a *exacto*. También es usual utilizar el término heurística cuando, utilizando el conocimiento que se tiene del problema, se realizan modificaciones en el procedimiento de solución del problema que, aunque no afectan a la complejidad del mismo, mejoran el rendimiento en su comportamiento práctico (Belén, Moreno et al., 2003, pp. 7-28). Ahora bien, el término metaheurítica se obtiene de anteponer a heurística el sufijo meta que significa "más allá" o "a un nivel superior". Las metaheurísticas son estrategias inteligentes para diseñar o mejorar procedimientos heurísticos muy generales con un alto rendimiento. El termino metaheurística apareció por primera vez en el articulo sobre búsqueda tabú (Fred Glover, 1986). A partir de entonces han surgido multitud de pautas para diseñar buenos procedimientos para resolver ciertos problemas que, al ampliar su campo de aplicación han adoptado la denominación de metaheurísticas. Podemos encontrarlas de varios tipos: metaheurísticas de relajación, constructivas, de búsqueda (como el algoritmo de recocido simulado), evolutivas (como los algoritmos genéticos) y combinaciones de estas.

Como se ha señalado en el párrafo anterior algunas de estas metaheurísticas fueron inspiradas por fenómenos naturales. Por ejemplo, tenemos una basada en el fenómeno físico denominado recocido de cristales. A la metaheurística que trata de simular este proceso se la denomina "**recocido simulado**". Lo más interesante de ésta es que al aplicar el análisis de la teoría de Markov se ha podido probar una convergencia en probabilidad a la solución óptima de problemas con ciertas características comunes. Pero sin perder respeto a lo teórico, lo que realmente impresiona es lo que logra este algoritmo en la práctica.

2. Optimización combinatoria

El problema que deseamos resolver podemos formularlo del siguiente modo:

$$\min_{x \in S} \; f(x) \qquad (2.1)$$

2.1. Algunos métodos de optimización

2.1.1. Optimización local

Los algoritmos de búsqueda local constituyen una interesante clase de métodos de aproximación basados en la mejora paulatina de la función objetivo mediante la exploración de entornos. Para la utilización de un algoritmo de búsqueda local necesitamos la representación previa de las soluciones, la definición de la función de costo y de, al menos, una estructura de entornos.

El concepto más importante en el análisis de los algoritmos de búsqueda local es el de optimalidad local, que puede ser descrito así:

Definición: Dado el problema *(S, f)* y una estructura de entornos $\aleph$, $\hat{i}$ es una solución mínima localmente o simplemente un mínimo local con respecto a $\aleph$ si $f(\hat{i}) \leq f(j), \quad \forall j \in S_{\hat{i}}$ (2.2)

Frente al concepto de optimalidad local, encontramos el de optimalidad global que puede ser definido así:

Definición: Dado el problema *(S, f)*, i_{opt} es una solución mínima globalmente o simplemente un mínimo global de (2.1) si $f(i_{opt}) \leq f(i), \quad \forall i \in S$ (2.3)

La desventaja principal de los algoritmos de búsqueda local es que éstos generalmente quedan atrapados en un óptimo local. Sin embargo, presentan la ventaja de ser fácilmente aplicables y bastante flexibles puesto que sólo es necesario especificar: el espacio de soluciones, la función de costo y una estructura de entornos.

2.1.2. Optimización global

Para evitar algunas de las desventajas de los algoritmos de búsqueda local, mientras se mantienen los principios básicos de éstos, es decir, iteración entre las soluciones del entorno, se introducen distintas variaciones:

(i) Ejecución de un algoritmo de búsqueda local para un gran número de soluciones iniciales. Asintóticamente (bajo la garantía de que todas las soluciones han sido usadas como solución inicial) tal algoritmo encuentra un óptimo global con probabilidad 1.

(ii) Introducción de estructuras de entornos más complejas, que puedan buscar en una parte más grande del espacio de soluciones.

(iii) Aceptación en una forma limitada de transiciones correspondientes a un incremento en el valor de la función de costo (en los algoritmos de búsqueda local sólo son aceptadas aquellas transiciones que corresponden a una disminución en dicho valor). Estas estrategias de búsqueda alternativas conducen a otros algoritmos que podemos clasificar como:

a) Métodos clásicos de optimización global entre los que podemos citar:

Búsqueda aleatoria pura (Ríos *et al*., 1997)

El método de búsqueda aleatoria pura consiste en generar aleatoriamente un número grande de soluciones y escoger la mejor de ellas. Este método no es muy eficiente, pues no utiliza información sobre f.

Métodos de multicomienzo (Ríos *et al*., 1997)

Este método consiste en generar N puntos desde los que se comienza una optimización con un método local proponiendo como solución la mejor así obtenida.

b) Métodos modernos de optimización global:

Estos métodos surgieron principalmente con los problemas de optimización combinatoria. Uno de estos métodos es el denominado recocido simulado, citado anteriormente, que estudiaremos en mayor profundidad en el siguiente apartado.

3. Recocido simulado

Kirkpatrick *et al*. (1982) y Cerny (1985) introdujeron los conceptos de recocido aplicándolos a problemas de optimización combinatoria. El nombre de recocido surge de la fuerte analogía entre el proceso físico del recocido de sólidos y el problema de resolver grandes problemas de optimización combinatoria.

3.1. Analogía física del recocido simulado

Kirkpatrick *et al*. (1982) describen un algoritmo intensivo de computación para encontrar soluciones a problemas de optimización arbitrarios. La esencia de este método es reconocer que la naturaleza realiza una optimización de la energía de un sólido cuando este es recocido para quitar defectos en la ordenación atómica.

En Física, se denomina recocido al proceso térmico por el que se obtienen estados de baja energía de un sólido en un "baño caliente". El proceso sigue dos pasos (Barker and Henderson, 1976, pp. 587-671):

(1) Incremento de la temperatura del baño caliente hasta un valor máximo en la cual el sólido se funde.

(2) Bajada cuidadosa de la temperatura del baño caliente.

Para cada sólido, hay al menos una estructura atómica que minimiza su energía (entropía mínima) a una temperatura de cero absoluto. Esta configuración de los átomos de mínima energía global es normalmente una estructura cristalina absolutamente regular con todos los átomos organizados. Sin embargo, hay muchas estructuras atómicas que son estables, pero no tienen globalmente energía mínima (estructuras meta-estables): estas estructuras son estables a temperatura cero porque moviendo átomos muy ligeramente aumentarán la energía total. Por consiguiente, representan mínimos locales de energía y corresponden a estructuras estables con ordenamientos no regulares de sus átomos (El mayor desorden atómico lo presentaría un vidrio). El menor mínimo de energía se obtiene cuando el material cristaliza con una ordenación regular de sus átomos y es entonces cuando decimos que se ha alcanzado un mínimo global de energía.

Ahora, utilicemos un símil geográfico para continuar la explicación: sea N el número de átomos. Si representamos gráficamente en un espacio de dimensión 3N+1 las coordenadas de los átomos y su energía, tendríamos un mapa topográfico. La energía total del material depende de la ordenación atómica porque las localizaciones de los átomos en el espacio real determinan la energía de interacción de todos los átomos. Entonces la altitud de cada posición en nuestra carta topográfica es la energía de la ordenación de los átomos representada para esa posición. Por tanto, nosotros hablamos, en una configuración dada, de la carta topográfica como representación de un paisaje de energía.

La mayoría de las posiciones en el paisaje de energía está en una cuesta. Unas cuantas posiciones están en los fondos de valles. El valle más profundo se corresponde con el mínimo global.

Valles superiores a este se denominan mínimos locales. Por supuesto, está la posibilidad de mínimos globales múltiples.

Para ir a lo largo de un camino desde un mínimo local hasta un mínimo global, primero debe ocurrir que la configuración debe aumentar de algún modo su energía. En otras palabras, la única manera de llegar a un valle más bajo en el paisaje de energía es ir ascendiendo primero. Recociendo un material, la energía para los movimientos ascendentes viene del baño caliente circundante.

Para hacer un cristal libre de defectos (partículas perfectamente ordenadas) mediante recocido, se eleva la temperatura del baño caliente hasta un poco por debajo del punto de fundición. A esta temperatura elevada, las fluctuaciones termales crearán y destruirán los defectos. Después de esperar lo suficiente para que el equilibrio termal sea alcanzado, el número de defectos que se crean por segundo es igual al del número destruido. Entonces, no se puede mejorar más la estructura cristalina a esta temperatura, por lo que se procede a bajarla un poco. De nuevo, el ritmo de eliminación de defectos excederá ligeramente al de creación hasta que el equilibrio termal se logre de nuevo. Entonces se vuelve a bajar la temperatura. La sucesión de cambios de temperatura y la cantidad de tiempo gastada en cada temperatura es un *esquema de recocido*. Si recociendo se alcanza el equilibrio termal en cada temperatura, puede demostrarse que el recocido produce energía mínima global a una temperatura de cero absoluto.

El recocido simulado es nada más que la simulación de la historia termal de un material que está sujeto a un esquema de recocido. El recocido simulado usa la función objetivo de un problema de optimización en lugar de la energía de un material real. Las fluctuaciones termales simuladas son los cambios en los parámetros ajustables del problema en lugar de las posiciones atómicas. Si para ese esquema de recocido se logra el equilibrio termal en cada temperatura, entonces la función objetivo alcanza su mínimo global cuando la temperatura simulada es próxima a cero.

3.2. El Algoritmo Metrópolis

El proceso físico de recocido puede ser modelado usando métodos de simulación. Mitra *et al* (1986) introdujeron un algoritmo para simular la evolución de un sólido en un baño caliente hasta alcanzar el equilibrio térmico. Este algoritmo está basado en técnicas Montecarlo y genera una sucesión de estados del sólido de la siguiente manera: dado el estado actual *i* del sólido con energía E_i, entonces el siguiente estado *j* está generado aplicando un mecanismo de perturbación que transforma el estado actual en el siguiente estado por una pequeña distorsión, por ejemplo, el desplazamiento de una partícula. La energía en el estado siguiente es E_j. Si la diferencia de energía, $E_j - E_i$, es ≤ 0, el estado *j* se acepta como el estado actual. Si $E_j - E_i > 0$, el estado *j* es aceptado con una cierta probabilidad dada por

$$\exp\left(\frac{E_i - E_j}{k_B T}\right)$$

donde *T* es la temperatura del baño caliente, que se actualiza cada cierto número de estados y k_B es una constante física denominada *constante de Boltzman*. La aceptación de la regla anterior es conocida como Criterio Metrópolis y el algoritmo como Algoritmo Metrópolis.

Si la bajada de la temperatura está hecha suficientemente despacio, el sólido puede alcanzar el equilibrio termal en cada temperatura. En el algoritmo Metrópolis éste es alcanzado generando un gran número de transiciones para un valor de la temperatura dado. El equilibrio termal está caracterizado por la distribución de Boltzmann (Toda *et al*., 1986).

Definición: [Distribución de Boltzmann] La probabilidad de que un sólido esté en estado *i* con energía E_i a temperatura *T*, está dada por:

$$P_T\{X = i\} = \frac{1}{Z(T)} \exp(-\frac{E_i}{k_B T})$$

donde X es una variable aleatoria que denota el estado actual del sólido y Z(T) es una constante de normalización

$$\text{Z(T)} = \sum_j \exp(\frac{-E_j}{k_B T})$$

donde el sumatorio está definido sobre todos los estados posibles.

3.3. El algoritmo de recocido simulado

Podemos aplicar el algoritmo Metrópolis para generar una sucesión de soluciones de un problema de optimización combinatoria. Para ello, basta observar la analogía existente entre un sistema físico con muchas partículas y un problema de optimización combinatoria.

Dicha analogía está basada en las siguientes equivalencias:

(i) Las soluciones de un problema de optimización combinatoria son equivalentes a los estados de un sistema físico.

(ii) El costo de una solución es equivalente a la energía del estado.

(iii) La temperatura del proceso de recocido se modeliza mediante un parámetro denominado de control.

Así, el algoritmo de recocido simulado puede verse como una iteración del algoritmo Metrópolis evaluado para valores decrecientes del parámetro control.

Como en un algoritmo de búsqueda local asumimos la existencia de una estructura de entornos y de un mecanismo de generación.

Definición [Criterio de aceptación]. Sean *(S,f)* una instancia de un problema de optimización combinatoria e *i, j* dos soluciones con costo *f(i), f(j)*respectivamente. Entonces el criterio de aceptación determina si *j* es aceptado desde *i* aplicando la siguiente probabilidad de aceptación:

$$P_c\{aceptar\ j\} = \begin{cases} 1 & si\ f(j) \le f(i) \\ \exp(\frac{f(i) - f(j)}{c}) & si\ f(j) > f(i) \end{cases} \qquad (4.1)$$

donde $c \in R^+$ denota el parámetro control.

Ahora, el mecanismo de generación corresponde al mecanismo de perturbación en el algoritmo Metrópolis, mientras que el criterio de aceptación corresponde al criterio Metrópolis.

Definición [Transición]. Una transición es una acción combinada que da como resultado la transformación de la solución actual en la siguiente. La acción consta de dos pasos:

(i) Aplicación del mecanismo de generación.

(ii) Aplicación del criterio de aceptación.

Una característica del algoritmo de recocido simulado es que, además de aceptar soluciones que supongan mejoras en la función de costo, son también aceptadas algunas otras que suponen un empeoramiento en esta (esta es la forma en que el método evita quedar atrapado en un óptimo local). Inicialmente, para un valor grande de *c*, prácticamente cualquier transición es aceptada; al ir disminuyendo *c*, sólo un número pequeño de transiciones a peores soluciones son aceptadas y finalmente, cuando el valor de c se aproxima a cero, no es aceptada ninguna transición a una solución peor. Esta característica significa que el algoritmo de recocido simulado, al contrario de lo que ocurre con los algoritmos de búsqueda local, puede escapar de un mínimo local mientras sigue manteniendo las características más favorables de estos, es decir, simplicidad y aplicabilidad general. Basta ver que la probabilidad de aceptar peores soluciones se hace mediante comparación del valor de $\exp(\frac{f(i)-f(j)}{c})$ con un número aleatorio generado de una distribución uniforme en el intervalo [0,1), es decir, $\exp(\frac{f(i)-f(j)}{c})$ representa la probabilidad de pasar a soluciones peores. Nótese que esta probabilidad depende de c y de la distancia entre *f(i)* y *f(j)*.

Como conclusión, podemos considerar el recocido simulado como una generalización de la búsqueda local.

Veamos ahora cómo aplicar este algoritmo al problema del viajante

4. El problema del viajante

El origen del problema del viajante, o en su terminología inglesa Traveling Salesman Problem (TSP), no está claro. Una guía para viajantes de 1832 menciona el problema e incluye ejemplos de viajes a través de Alemania y Suiza, pero no contiene un tratamiento matemático del mismo. El problema fue definido en los años 1800s por el matemático irlandés W. R. Hamilton y por el matemático británico Thomas Kirkman. Todo parece indicar que la forma general del TSP fue estudiada, por primera vez por matemáticos en Viena y Harvard, durante los años 1930s. Entre estos destacan Karl Menger, quien definió los problemas, considerando el obvio algoritmo de fuerza bruta, y observando la no optimalidad de la heurística de vecinos más cercanos Hassler Whitney de la Universidad de Princeton introdujo el nombre “travelling salesman problema” poco después (Applegate, D. L. et al., 2006)

Durante los años 1950 a 1960, el problema fue incrementando su popularidad entre el círculo de científicos de Europa y Estados Unidos. Una notable contribución fue la de George Dantzig, Delbert Ray Fulkerson y Selmer M. Johnson de la Corporación RAND en Santa Mónica, quienes expresaron el problema como Programación Lineal en Enteros y desarrollaron para solucionarlo el método de Planos Cortantes. Con este nuevo método, resolvieron una instancia con 49 ciudades, óptimamente,

mediante la construcción de un recorrido y probando que no había un recorrido que pudiera ser más corto. En las siguientes décadas, el problema fue estudiado por muchos investigadores, matemáticos, científicos de la computación, químicos, físicos, etc.

Richard M. Karp mostró en 1972 que el TSP era un problema NP-duro. Esto tiene su explicación matemática por la evidente dificultad computacional para encontrar recorridos óptimos.

Gran progreso tuvo a finales de los 70s y principios de los 80s, donde Grötschel, Padberg, Rinaldi y otros, manejaron soluciones exactas para instancias con 2392 ciudades, usando Planos Cortantes y Ramificación y Acotación.

En los 90s, Applegate, Bixby, Chvátal, y Cook desarrollaraon el programa Concorde, el cual es usado en muchos de los registros de soluciones recientes. Gerhard Reinelt publicó la TSPLIB en 1991, una colección de instancias de pruebas de dificultad variable, la cual es usada por muchos grupos investigativos para comparar resultados. En 2006, Cook y otros, obtuvieron un recorrido óptimo para 85,900 ciudades.

Descripción del problema

En el problema se presentan N! rutas posibles, aunque se puede simplificar ya que dada una ruta nos da igual el punto de partida y esto reduce el número de rutas a examinar en un factor N quedando (N-1)! Como no importa la dirección en que se desplace el viajante, el número de rutas a examinar se reduce nuevamente en un factor 2. Por lo tanto, hay que considerar (N-1)!/2 rutas posibles.

En la práctica, para un problema del viajante con 5 ciudades hay 12 rutas diferentes y no necesitamos un ordenador para encontrar la mejor ruta, pero apenas aumentamos el número de ciudades las posibilidades crecen exponencialmente (en realidad, factorialmente):

Para 10 ciudades hay 181.440 rutas diferentes

Para 30 ciudades hay más de 4·10^31 rutas posibles. Un ordenador que calcule un millón de rutas por segundo necesitaría 10^18 años para resolverlo. Dicho de otra forma, si se hubiera comenzado a calcular al comienzo de la creación del universo (hace unos 13.400 millones de años) todavía no se habría terminado.

Puede comprobarse que por cada ciudad nueva que incorporemos, el número de rutas se multiplica por el factor N y crece exponencialmente (factorialmente). Por ello el problema pertenece a la clase de problemas NP-duros.

a) Modelación como un problema de grafos

El TSP puede ser modelado como un grafo ponderado no dirigido, de manera que las ciudades sean los vértices del grafo, los caminos son las aristas y las distancias de los caminos son los pesos de las aristas. Esto es un problema de minimización que comienza y termina en un vértice especifico y se visita el resto de los vértices exactamente una vez. Con frecuencia, el modelo es un grafo completo (cada par de vértices es conectado por una arista). Si no existe camino entre un par de ciudades, se añade arbitrariamente una arista larga para completar el grafo sin afectar el recorrido óptimo.

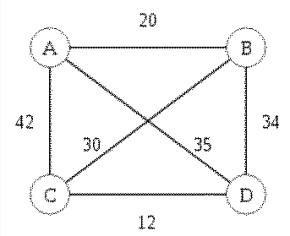

Figura 1: TSP simétrico con 4 ciudades

Problemas relacionados:

-Problema del agente viajero con cuello de botella (bottleneck TSP)

- La generalización del TSP trata con "estados" que tienen (una o más) "ciudades" y el viajante tiene que visitar exactamente una ciudad de cada "estado". También se conoce como el Problema del Político Viajero.

- El problema de ordenamiento secuencial trata con el problema de visitar un conjunto de ciudades donde se tienen en cuenta las relaciones de precedencias entre las ciudades.

- El problema del viajante comprador trata con un comprador que está cargado con un conjunto de productos. Él puede comprar estos productos en varias ciudades, pero tienen diferentes precios y no todas las ciudades ofrecen los mismos productos. El objetivo es encontrar una ruta entre un subconjunto de ciudades, los cuales minimicen el costo total (costo de viaje + costo de la compra) y habilite la compra de todos los productos requeridos.

b) Formulación como un problema de programación lineal entera

En el problema del viajante existe un conjunto de n ciudades (nodos), V = {1, 2, 3,…, n}, y un conjunto de caminos (arcos) uniendo cada una de las ciudades. Así el camino (i,j) Є A, c_{ij} es la "distancia" (función objetivo) para ir de la ciudad i a la ciudad j. Un viajante debe realizar un recorrido (tour) comenzando en una cierta ciudad de origen y luego visitar todas las otras ciudades una única vez y retornar a la ciudad de origen. El problema consiste en hallar el recorrido (tour) de distancia mínima. evitando subtours.

El problema del TSP tiene el siguiente modelo matemático:

Sea x_{ij} la variable de decisión del problema:

$$x_{ij} = \begin{cases} 1, & si \quad el \quad arco \quad (i,j) \quad es \quad utilizado \quad para \quad hacer \quad el \quad tour \\ 0, & en \quad caso \quad contrario \end{cases}$$

El modelo matemático asume la siguiente forma:

$$\min \quad z(x) = \sum_{(i,j)\in A} c_{ij} x_{ij} \quad (1)$$

s.a.:

$$\sum_{\{i:(i,j)\in A\}} x_{ij} = 1, \quad \forall j \in V \quad (2)$$

$$\sum_{\{j:(i,j)\in A\}} x_{ij} = 1, \quad \forall i \in V \quad (3)$$

$$\sum_{\{(i,j)\in A: i\in U,\, j\in (V-U)\}} x_{ij} \geq 1, \quad 2 \leq |U| \leq |V| - 2 \quad (4)$$

La ecuación (1) corresponde al cálculo de la función objetivo.

El conjunto de restricciones (2) indica que se puede llegar a cada ciudad desde una única ciudad anterior.

El conjunto de restricciones (3) indica que desde la ciudad *i* se puede pasar a una única ciudad (de la ciudad *i* se puede salir por un único camino).

El conjunto de restricciones (4) evita que se generen subtours.

El problema tiene considerables aplicaciones prácticas, aparte de las más evidentes en áreas de logística de transporte, así como en cualquier negocio de reparto, pequeño o grande. Por ejemplo, en robótica, permite resolver problemas de fabricación para minimizar el número de desplazamientos al realizar una serie de perforaciones en una plancha o en un circuito impreso. También puede ser utilizado en planificación de rutas de vehículos, manufactura flexible, etc.

Debido a la importancia práctica del problema, muchos autores se plantearon cómo abordar la solución a este problema. Puesto que se trata de un problema NP-duro, las líneas tradicionales para atacarlo son las siguientes:

- Formular algoritmos para encontrar soluciones exactas (estos trabajan más rápidos en problemas con dimensiones pequeñas).

- Formular algoritmos heurísticos o "subóptimos" (por ejemplo: algoritmos que den aparentemente o probablemente buenas soluciones, pero no devuelven el óptimo necesariamente).

- Encontrar los casos especiales para el problema ("subproblemas") para los cuales son posibles heurísticas mejores o algoritmos exactos.

La solución más directa puede ser, intentar todas las permutaciones (combinaciones ordenadas) y ver cuál de estas es la menor (usando una Búsqueda de fuerza bruta). Otra solución posible viene proporcionada por los algoritmos de ramificación y acotación, los cuáles pueden ser usados para procesar TSP que contienen entre 40 y 60 ciudades. También tenemos algoritmos de mejoras progresivas (iterativas) los cuales utilizan técnicas de Programación lineal que trabajan bien para más de 200 ciudades. Implementaciones de ramificación y acotación y un problema específico de generación de cortes (Ramificación y poda); este es el método elegido para resolver grandes instancias. Esta aproximación retiene el record vigente, resolviendo una instancia con 85,900 ciudades, (Applegate et al., 2006)

Según se ha indicado anteriormente otro método para abordar el problema de encontrar soluciones al TSP viene dado por la formulación de Algoritmos heurísticos y aproximados. De entre estos varios que retornan rápidamente buenas soluciones han sido creados. Métodos modernos pueden encontrar soluciones para problema extremadamente largos (millones de ciudades) en un tiempo razonable. De entre todos estos vamos a centrarnos en los que se basan en mejoras aleatorias EL TSP es la base para muchas heurísticas diseñadas para la optimización combinatoria como: los algoritmos genéticos, el recocido simulado, la Búsqueda tabú, la optimización por colonias de hormigas, entre otras.

Veamos cómo podemos aplicar el algoritmo de recocido simulado al TSP

- El espacio de soluciones

Tenemos n ciudades y una matriz *nxn* (c_{ij}) cuyos elementos denotan la distancia entre cada par *i,j* de las n ciudades. Un tour o recorrido es definido como un camino cerrado en el que se visita cada ciudad exactamente una vez. El problema consiste en encontrar el camino de longitud mínima. Para ese problema cada solución está dada por una permutación cíclica $x=(x(1),x(2),...,x(n))$ donde $x(k)$ denota la siguiente ciudad a visitar después de la ciudad k, con $x^1(k)\neq k$, $l=1, ...n-1$ y $x^n(k)=k$, $\forall k$ [2]. De este modo cada solución se corresponde con un único tour.

El espacio de soluciones viene dado por $S=\{$ las permutaciones cíclicas x de n ciudades $\}$

- La estructura de entornos

Para este trabajo se ha considerado la estructura de entornos N_k, llamada *k-cambio*, que considera para cada solución i un entorno S_i consistente en el conjunto de soluciones que pueden obtenerse desde la solución dada i cambiando k aristas del tour correspondiente a la solución i y reemplazándolos con otras k aristas tal que se obtiene un nuevo tour (Lin, S. (1965))

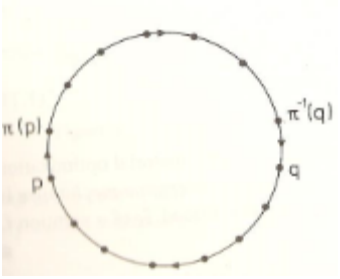

Figura 2. Un tour en una instancia del TSP

La estructura de entornos descrita anteriormente para este problema no es la única que podemos encontrar en la literatura. Otro ejemplo de estructura de entornos puede ser obtenido utilizando los $N_2(p,q)$ *cambios* (Aarts, E. and Korst, J. (1989))

- La función de costo la hemos definido anteriormente

Una vez definido el problema y los elementos necesarios para aplicar el algoritmo veamos algunos resultados prácticos. La figura de abajo compara una solución aleatoria al problema del

[2] $x^1(k)$ representa la ciudad visitada l trayectos después de haber visitado la ciudad k

viajante para 100 ciudades bien organizadas a una después de aplicar un recocido simulado durante un millón de iteraciones. Éste representa un problema muy grande, ya que la cantidad de soluciones como la de abajo comparada con el espacio de todos los caminos posibles es muy pequeña.

Figura 3 Dos ejemplos de caminos para el viajante: El primero seleccionado al azar, el segundo encontrado después de un millón de iteraciones del recocido simulado.

5. Conclusiones

En este pequeño trabajo se han presentado ejemplos de cómo podemos imitar procesos que ocurren en la naturaleza con el objeto de resolver problemas reales que se presentan en distintas ramas del conocimiento. Algo tan inicialmente alejado de un problema de optimización combinatoria como puede ser el proceso físico de recocido, da lugar al desarrollo de un algoritmo, el de recocido simulado, que nos permite abordar alguno de estos problemas. Entre los que permite resolver podemos citar al famoso "problema del viajante", ampliamente abordado en la literatura. La ventaja principal de este algoritmo es la de que en algunos casos va permitir pasar a "una solución peor" evitando de esta manera quedar atrapados en óptimos locales.

Bibliografía

Applegate, D. L.; Bixby, R. M.; Chvátal, V.; Cook, W. J. (2006), *The Traveling Salesman Problem*, Princeton University Press, cop.

Aarts, E. and Korst, J. (1989) *Simulated Annealing and Boltzmann Machines*, JOHN WILEY & SONS

Barker, J.A. And Henderson, D. (1976) "What is liquid? Understanding the states of matter", *Reviews of Modern Physics*, 48, 587-671

Belén Melián, José A., Moreno Pérez, J. y Moreno Vega, M. (2003) "Metaheurísticas, una visión global " *Inteligencia Artificial, Revista iberoamericana de Inteligencia Artificial*, 19, 7-28

Cerny, V. (1985) "Thermodynamical approach to the traveling salesman problem: an efficient dimulation algorithm", *Journal of Optimization Theory and Applications*, 45, 41-51.

Dantzig, G.B. (1963), *Linear Programming and Extensions*, Pricenton, Pricenton University Press.

Glover, F.(1982) Future paths for integer programming and links to artificial intelligence. *Computers and Operations Research*, 5, 533-549

Kirkpatrick, S. Gelatt, C.D. Andvecchi, M.P. (1982) "Optimization by simulated annealing", *Sience*, 220, 671-680.

Lin, S. (1965) "Computer solutions of the traveling salesman problem", *Bell System Technical Journ*, 44, 2245-2269.

Lin, S. (1973) "An effective heuristic algorithm for the travelling salesman problem", *Operation Research*, 21, 498-516.

Metropolis, N., Rosembluth, A., Teller, A. And Teller, E. (1953) "Equation of state calcualtions by fast computing machines", *Journal of Chemicall Physics*, 21, 1087-1092.

Papadimitriou, C.H. And Steiglizt, K. (1982) *Combinatorial Optimization: Algorithms and Complexity*, New York, Prentice Hall.
Reeeves, C.R. (2003) *Genetic Algorithms* Cap.3 en F. Glover y Kochenberger, 0. (eds.) *Handbook on MetaHeuristics.*
Ríos Insua, D. Ríos Insua, S y Martín, J. (1997) *Simulación. Métodos y aplicaciones*, "Ra-Ma".
Toda, M., Kubo, R. And Saitô, N. (1983) *Statistical Physics*, Berlín, Springer-Verlag.

Francisco Moreno Soto, Licenciado en Matemáticas en la Especialidad de Estadística e Investigación Operativa por la Universidad de Sevilla, DEA por la Universidad de Extremadura (UEX), Diplomado en Estadística por la UEX, Técnico Superior en Administración y Finanzas. Ha presentado ponencias y escrito artículos siempre en relación con las Matemáticas centrados fundamentalmente en temas relacionados con la Estadística y la Investigación Operativa.
pacomoreno@unex.es