Introducción a la Optimización Combinatoria

Jose A. Lozano, Roberto Santana

28 de febrero de 2015

1. Objetivos

Los objetivos de este tema son los siguientes:

- Ser capaces de identificar problemas de optimización combinatoria y distinguirlos de otro tipo de problemas de optimización
- Adquirir una noción intuitiva acerca de cuando un problema es NP-hard
- Ser capaces de formalizar un problema de optimización combinatoria. Esto incluye su escritura matemática.

2. Introduccion

Durante este curso nos vamos a enfrentar a problemas de optimización combinatoria. Este tipo de problema en su forma más general, puede escribirse matemáticamente como:

$$\begin{aligned} &min && f(\boldsymbol{x}) \\ &sujeto\ a && \\ & && g_i(\boldsymbol{x}) \geq b_i \quad i = 1, 2, \ldots, m \\ & && h_j(\boldsymbol{x}) = c_j \quad j = 1, 2, \ldots, s \ . \end{aligned}$$

En el problema anterior $\boldsymbol{x}$ representa una posible solución al problema y en el caso general estará expresado por medio de un vector.

A pesar de que el problema de optimización se ha escrito como un problema de minimización, esto no limita la generalidad del enunciado ya que unas pequeñas modificaciones nos llevan a pasar de un problema de minimización a otro de maximización:

$$min\ f(\boldsymbol{x}) = max - f(\boldsymbol{x}) \ .$$

Del mismo modo las restricciones de igualdad pueden convertirse fácilmente en restricciones de desigualdad, $h_j(\boldsymbol{x}) = c_j$ es equivalente a:

$$h_j(\boldsymbol{x}) \geq c_j \ \text{ y } \ h_j(\boldsymbol{x}) \leq c_j$$

En los problemas que se tratarán a lo largo del curso, se supondrá que la función $f(\boldsymbol{x})$ esta definida en un conjunto finito o numerable, es decir, el conjunto de soluciones posibles al problema es un conjunto contable (finito o infinito numerable).

El caso más sencillo, conocido y estudiado de este tipo de problemas de optimización es aquel en el que la función $f(\boldsymbol{x})$ es una función lineal, es decir, suponiendo que $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ se expresa como:

$$f(\boldsymbol{x}) = a_1 \cdot x_1 + a_2 \cdot x_2 + \ldots + a_n \cdot x_n$$

y las restricciones también son lineales. Dicho problema es lo que se denomina un problema de programación lineal y se suele estudiar en los cursos básicos de cualquier grado de ciencia o ingeniería. En este caso, existe un algoritmo que puede resolver este problema de manera bastante eficiente, el algoritmo Simplex. A pesar de que en la formulación general del problema de programación lineal las variables $x_i, \quad i = 1, 2, \ldots, n$ pueden pertenercer a $\mathbb{R}$, el conjunto de soluciones a un problema concreto es finito y viene dado por las soluciones que puede alcanzar el algoritmo Simplex.

En la mayoría de los problemas que se estudiarán más adelante ni la función será lienal ni las restricciones lo serán. La mayoría de estos problemas se encuentran en la clase de los problemas NP-completos, es decir, son problemas para los cuales no se conoce ningún algoritmo de resolución que se ejecute en un tiempo polinomial en relación con el tamaño del problema.

El ejemplo mas clásico de problema NP-completo es el problema del agente viajero. El problema del agente viajero consiste en visitar n ciudades, pasando una sola vez por cada una de ellas y volver al punto del que se partio de manera que se recorra la menor cantidad de Kms. posible.

Los problemas NP-completos tienen la característica de que no pueden resolverse, en general, con algoritmos de optimización exactos. Los algoritmos de optimización exactos son aquellos que garantizan encontrar una solución óptima. Lo que hace inviable esta aplicación de algoritmos de optimización exactos en problemas NP-completos es el tiempo de cómputo. Por poner un ejemplo, si en el problema del agente viajero con 20 ciudades, tuvieramos un ordenador que tardase para enumerar todas las soluciones y elegir la mejor un tiempo de una hora, si el problema tuviese 21 ciudades el ordenador necesitaría 20 horas, para 22 ciudades el tiempo subiría a 17.5 dias, y se necesitarían 600 años para resolver un problema con 25 ciudades.

El hecho de que con este tipo de algoritmos exactos sea inviable acometer la solución de problemas de tamaño mediano hace que en la comunidad científica nazcan otro tipo de técnicas que, a pesar de no asegurar que se encuentre la solución óptima, si que se asegura que se encuentra una solución bastante buena pero con la ventaja, de que esta búsqueda se lleva a cabo en un tiempo de ejecución razonable.

3. Problemas de Optimización Combinatoria Clásicos

En este apartado veremos algunos problemas de optimización a los que nos enfrentaremos a lo largo del curso. Estos problemas deben sevir de botón de muestra para la aplicación de los algoritmos que se explicarán más adelante. Por supuesto todos los problemas son de tipo combinatorio y al mismo tiempo pertenecen al grupo de los NP-completos.

3.0.1. El problema del agente viajero

El problema del agente viajero (TSP) consiste en, dadas n ciudades y una matriz $D = [d_{ij}]_{i,j=1,2\ldots,n}$ con las distancias entre cada par de ciudades, encontrar un ciclo que pase todas las ciudades una sola vez, terminando en la ciudad de la que se partio y que recorra el mínimo número de kilómetros. En términos de grafos lo que estamos buscando es el ciclo dirigido Hamiltoniano con menor valor dentro del grafo completo K_n.

En una primera aproximación al problema, podemos considerar como espacio de búsqueda o conjunto de soluciones a:

$$F = \{(\pi_1 \pi_2 \ldots \pi_n) \mid \pi_i \in \{1, 2, \ldots, n\} \text{ y } \pi_i \neq \pi_j \ \forall i \neq j\}$$

es decir, el conjunto de todas la permutaciones cíclicas de los números $\{1, 2, \ldots, n\}$.

Una permutación de este tipo se puede ver como un ciclo si interpretamos π_i como la ciudad que sigue a la ciudad π_{i-1} para $i = 2, \ldots, n$. De esta manera la longitud de una permutación $\pi = (\pi_1 \pi_2 \ldots \pi_n)$ sería:

$$\sum_{i=2}^{n} d_{\pi_{i-1}\pi_i} + d_{\pi_n \pi_1} \quad .$$

En la Figura 1 se puede ver un ciclo para el problema del agente viajero con $n = 6$ cuya permutación es (3 2 5 6 1 4).

Una característica importante de cualquier problema de optimización combinatorio es el espacio de búsqueda o conjunto de soluciones. Su dimensión nos da una medida de la complejidad del problema.

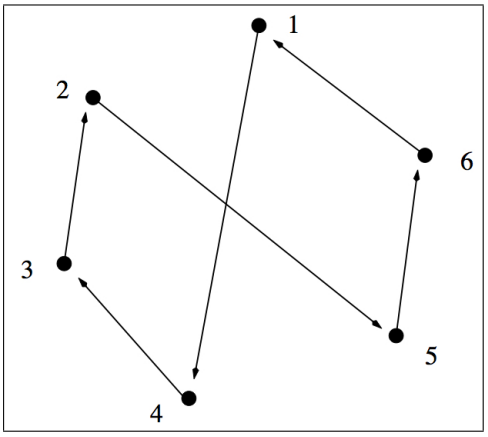

Figura 1: Ejemplo de una solución para el problema del agente viajero

El tamaño del espacio de búsqueda en el caso del TSP es $(n-1)!/2$ si suponemos que la matriz D es simétrica. Para hallar dicho valor basta con considerar el número de permutaciones de n números, claramente $n!$, y ahora tener en cuenta que dada una permutación π, existen otras n permutaciones que son similares a la anterior pero que comienzan en otra cidudad diferente, y cada una de estas puede hacer el recorrido en cualquiera de los dos órdenes posibles. Por lo tanto el número de soluciones posibles es $n!/2n = (n-1)!/2$. Si se supone que la matriz D no es simértica en este caso el espacio de búsqueda es $(n-1)!$.

3.0.2. El problema de la partición de un grafo

Consideremos un grafo $G = (\mathcal{X}, U)$ con conjunto de vértices $\mathcal{X}$ y conjunto de aristas U. Supongamos que el número de nodos es par, es decir $|\mathcal{X}| = 2n$, además cada arista $a_i = \{u, v\}$ tiene asignado un peso p_i. Se trata de hallar una partición $\mathcal{X}_1, \mathcal{X}_2 \in \mathcal{X}$ del conjunto de vértices tal que $|\mathcal{X}_1| = |\mathcal{X}_2|$ y de forma que se minimice la suma de los pesos de los arcos (u, v) con $u \in \mathcal{X}_1$ y $v \in \mathcal{X}_2$.

El espacio de búsqueda en este caso viene dado por el conjunto de particiones del conjunto $\mathcal{X}$ en dos conjuntos del mismo cardinal. El tamaño del espacio de búsqueda es por lo tanto:

$$\begin{pmatrix} 2n \\ n \end{pmatrix}$$

3.0.3. El problema de la mochila

Supongamos que disponemos de un conjunto de objetos $\{o_1, o_2, \ldots, o_n\}$. Cada objeto o_i del conjunto tiene asociado un peso p_i y un valor v_i. El problema consiste en, dada una mochila con capacidad C, encontrar el subconjunto de objetos que sin superar la capacidad de la mochila, maximiza el valor de los objetos introducidos.

Este problema puede escribirse de manera sencilla como un problema de programación entera. Consideremos la variable x_i que toma el valor 1 si el objeto i se introduce en la mochila, y 0 en el caso contrario. Matemáticamente el problema puede expresarse como:

$$\begin{aligned} max \quad & \sum_{i=1}^{n} v_i \cdot x_i \\ sujeto\ a \quad & \\ & \sum_{i=1}^{n} p_i \cdot x_i \leq C \end{aligned}$$

En este caso el espacio de búsqueda está compuesto por todos los subconjuntos de objetos cuya suma de pesos no supera C. El cálculo del tamaño del espacio de búsqueda se hace imposible a menos que se tenga algún conocimiento acerca de los pesos de los objetos.

3.0.4. El problema de la planificación (Scheduling)

El problema de planificación consiste en la minimización del tiempo de realización de un conjunto de trabajos en un conjunto de máquinas.

Supongamos que debemos realizar un conjunto de trabajos $\{T_1, T_2, \ldots, T_n\}$ y que cada trabajo T_i requiere de la realización de un conjunto ordenado de operaciones $\{O_{i1}, O_{i2}, \ldots, O_{im}\}$. Cada una de estas operaciones O_{ij} se debe realizar en una de las m máquinas $\{M_i, M_2, \ldots, M_m\}$ de las que se dispone y posee un tiempo de proceso p_{ij}. Por supuesto debemos hacer varias suposiciones y al mismo tiempo considerar varias restricciones: una operación debe realizarse de forma ininterrumpida en una máquina, en cada instante de tiempo una máquina únicamente puede procesar una operación, en cada trabajo una operación no puede comenzar a procesarse hasta que las operaciones previas hayan terminado de hacerlo.

La función a optimizar más común consiste en procesar todos los trabajos lo más rápido posible, es decir, minimizar el tiempo de terminación del último trabajo. Existen versiones más realistas de este problema, por ejemplo, podríamos considerar que cada trabajo posee un tiempo de entrega d_i. En este caso la función a minimizar sería la discrepancia entre el tiempo de finalización de cada trabajo y la fecha de entrega.

Definir en este caso el espacio de búsqueda de forma matemática no es una tarea trivial. Para ello veamos como podría definirse una solución al problema. Una solución al problema puede venir dada como un tiempo de comienzo para cada actividad, ni que decir tiene que estos tiempos de comienzo de cada actividad deben cumplir las restricciones para obtener una solución válida. Es evidente que existen otras maneras de definir o codificar una solución al problema (más adelante estudiaremos algunas de ellas), pero en cualquier caso calcular el tamaño del espacio de búsqueda parece una tarea complicada.

3.0.5. El problema de la ruta de vehículos

Una compañia dispone de una flota de m vehículos con la cual debe dar servicio a n clientes. El i-ésimo cliente realiza una peticion de c_i unidades, mientras que el vehículo j-ésimo dispone de una capacidad de C_j unidades. La distancia entre cada par de clientes viene dada por una matríz $D = [d_{ij}]_{i,j=0}^{n}$ (se supone que el cliente 0 es el punto de partida de los vehículos) y además cada vehículo no puede recorrer una distancia mayor de D unidades.

El problema consiste en establecer el recorrido de los camiones de forma que se satisfaga toda la demanda de los clientes y que la distancia recorrida por los camiones sea mínima.

Este problema puede escribirse como un problema de progamación lineal, veamoslo. La solución se puede representar como una permutación de los números $\{1, 2, \ldots, n\}$ que puede escribirse de la siguiente forma:

$$(\pi_{11}, \pi_{21}, \ldots, \pi_{n_1 1}, \pi_{12}, \pi_{22}, \ldots, \pi_{n_2 2}, \ldots, \pi_{1m}, \pi_{2m}, \ldots, \pi_{n_m m}) \ .$$

en la permutación anterior π_{ij} representa la i-ésima ciudad que recorre el vehículo j-ésimo, por lo tanto el vehículo k recorre n_k ciudades diferentes. Teniendo en cuenta la representación anterior el problema puede ser escrito como:

$$min \sum_{k=1}^{m} \sum_{i=0}^{n_k} d_{\pi_{ik}\pi_{i+1k}}$$

sujeto a

$$\begin{aligned} \sum_{i=1}^{n_k} c_{\pi_i k} &\leq C_k \quad k = 1, 2, \ldots, m \\ \sum_{i=0}^{n_k} d_{\pi_{ik}\pi_{i+1k}} &\leq D \quad k = 1, 2, \ldots, m \ . \end{aligned}$$

Al igual que ocurría con el problema anterior el cálculo del tamaño del espacio de búsqueda no es una tarea nada trivial.

3.0.6. Otros problemas

Los ejemplos vistos en las secciones anteriores deben servir para darnos cuenta de la magnitud del problema al que nos enfrentamos. Todos los ejemplos se corresponden con problemas combinatorios, sin embargo el mundo

de las ciencias y de la técnica está lleno de problemas de optimización. Incluso tareas que se resuelven en principo a diario constituyen un problema de optimización de difícil solución.

Enunciaré a continuación un conjunto de problemas que serán tratados de alguna manera a lo largo del curso: el problema de la partición de un grafo, el problema del conjunto de vértices independientes, coloreado de un grafo, problema del emplazamiento, asignación de tareas, visión artificial, asignacion de recursos a lugares, construcción de horarios, localización de concentradores en redes de comunicación.

4. Ventajas y desventajas de los métodos heurísticos frente a los exactos

Este apartado compara cuales son las ventajas y desventajas de los métodos heurísticos frente a los métodos exactos.

Las principales ventajas de los métodos exactos de optimización frente a los heurísticos son las siguientes:

Alcanzan la solución óptima.

Esta es la principal ventaja de los métodos exactos, si se deja ejecutar al algoritmo el tiempo suficiente estos devuelven una solución óptima al problema. La seguridad de alcanzar una solución óptima viene avalada por demostraciones matemáticas.

Estan desarrollados y estudiados matemáticamente.

La mayoría de estos algoritmos de optimización exactos fueron desarrollados en las décadas de los 60 y los 70. Desde entonces muchos investigadores se han dedicado a estudiarlos matemáticamente obteniendo condiciones de convergencia y velocidades de convergencia.

Están experimentalmente muy probados.

Como se ha apuntado anteriormente estos algoritmos tienen una antigüedad de unos 40 años, esto supone que han sido frecuentemente aplicados durante este tiempo para resolver problemas prácticos. Esta frecuente aplicación supone una fuente inestimable de información y experiencia. De hecho es posible encontrar en la bibliografía cantidad de libros dedicados a dichos métodos.

Las desventajas de estos métodos frente a los heurísticos radican en que:

Es imposible acometer problemas con alta dimensionalidad.

La mayoría de los problemas que son de interés en la práctica son problemas NP-completos. Sin embargo para dichos problemas no existen algoritmos capaces de resolverlos de forma óptima en un tiempo polinomial. Por lo tanto, dado que los algoritmos exactos son capaces de resolverlos, esto significa que su tiempo de ejecución va a ser exponencial en el tamaño del problema. Por tanto los algoritmos exactos sólo pueden ser aplicados en la práctica a problemas de dimensión o tamaño pequeño.

Necesitan de una formulación matemática del problema.

En la mayoría de los casos estos métodos necesitan de una formulación similar a la utilizada por la programación lineal. Es decir, necesitan que tanto la función a optimizar como las restricciones se puedan escribir de manera matemática. Además en muchas ocasiones son métodos desarrollados ad hoc para un tipo de problemas y no pueden ser aplicados a otros.

Poseen una implementación complicada.

Otra de las desventajas de estos métodos radica en la implementación. Los algoritmos de optimización exactos suelen tener unas profundas bases matemáticas lo que significa que la implementación tiene que tener en cuenta estas matemáticas y por lo tanto no es sencilla de realizar en el ordenador.

Por otro lado las ventajas de los métodos heurísticos frente a los exactos son:

Consumen un tiempo de ejecución razonable.

Habitualmente los métodos heurísticos consumen tiempos de ejecución razonables. Del mismo modo tienen la ventaja de que pueden ser detenidos en cualquier momento y obtener la mejor solución conseguida por el método hasta dicho instante.

Son aplicables de manera general.

Esta es una de las características más importantes de este conjunto de métodos. Su aplicación es de caracter general, no necesitan de una formalización matemática, ni siquiera de una expresión analítica para la función a optimzar. Dado un problema las adaptaciones a realizar en el método son mínimas.

Son, en general, fáciles de implementar y de entender.

Otro ventaja de estos métodos es la facilidad con la que se comprenden. Como veremos en capítulos posteriores, todos ellos están basados en ideas que son muy intuitivas pero que sorprendentemente conducen a resultados muy satisfactorios. Esta facilidad para comprender los métodos y la simpleza que ello indica, provoca que la implementación de los mismos se realice de forma muy sencilla.

Las desventajas de los métodos heurísticos son las siguientes:

Son algoritmos de optimización aproximados.

Como ya hemos dicho anteriormente los algoritmos heurísticos no nos aseguran la obtención de un óptimo. Esto tiene varias implicaciones. Por un lado es imposible saber lo alejada que esta la solución que nos devuelve el método, de un óptimo. Por otro no existe ningún criterio de parada del algoritmo en función de la calidad de la solución. En muchos casos los algoritmos llegan a situaciones donde no puede esperarse ninguna mejora, con lo que hay que detener de la ejecución.

Estan poco estudiados matemáticamente.

Los métodos heurísticos, son casi todos, mucho más recientes que los métodos exactos. Además la mayoría de ellos utilizan la probabilidad para llevar a cabo la búsqueda. Estas dos razones han dado lugar a que el desarrollo matemático llevado a cabo para estos métodos haya sido muy escaso. Los resultados matemáticos obtenidos para los algoritmos heurísticos suelen hacer referencia a la convergencia del método y desgraciadamente no aportan gran cosa desde el punto de vista práctico.

Es muy dificil realizar una comparación de los métodos.

Llevar a cabo una comparación de los métodos heurísticos es algo muy complicado. Dada la falta de resultados matemáticos la única manera de llevarla a cabo es de forma experimental. Esto obviamente significa que las comparaciones hay que mirarlas con mucho recelo ya que la mejora de un algoritmo sobre otro puede ser debida a aspectos de implementación como la habilidad del programador, la propiedades de optimización de código del compilador, el uso de los diferentes componentes del ordenador por parte del programa (paginación, cache, etc,..).

5. A modo de conclusión

Tras esta introducción varias ideas nos deben quedar claras.

En general se puede decir, que en la mayoria de los problemas siempre va a funcionar mejor un método creado exclusivamente para dicho problema que uno de los métodos que vamos a estudiar a continuación. Sin embargo, lo que les hace interesantes a los métodos que estudiaremos es su versatilidad, para cualquier problema

para el que no haya ningún método conocido, siempre va a ser posible aplicar estos métodos y además con un éxito considerable.

Por otro lado es importante tener en cuenta que no existe un método de optimzación que sea mejor que otro. Siempre es posible encontrar problemas donde el primer método supera al segundo y problemas donde el segundo supera al primero. Se trata por lo tanto de encontrar para cada problema aquel método que mejores resultados obtiene, que además en muchos casos consistirá en un híbrido de varios métodos.