UNIVERSIDAD DE GRANADA

E.T.S. DE INGENIERÍAS INFORMÁTICA y DE TELECOMUNICACIÓN

Departamento de Ciencias de la Computación e Inteligencia Artificial

Algorítmica

Guión de Prácticas

Práctica 4: Algoritmos de Vuelta Atrás (Backtracking) y de Ramificación y Poda (Branch and Bound)

Curso 2017-2018

Grado en Informática

1. Objetivo

El objetivo de esta práctica es que el estudiante aprecie la potencia de los métodos vuelta atrás para resolver problemas, pero también comprenda sus limitaciones. Para ello cada equipo de estudiantes deberá resolver uno de los problemas (asignado al azar) que se describen en la sección 2, así como exponer y defender su propuesta en clase. Adicionalmente, todos los equipos deberán implementar (y exponer y defender) algoritmos de ramificación y poda para el problema del viajante de comercio siguiendo las indicaciones descritas en la sección 3.

2. Problemas

2.1. División en dos equipos

Se desea dividir un conjunto de n personas para formar dos equipos que competirán entre sí. Cada persona tiene un cierto nivel de competición, que viene representado por una puntuación (un valor numérico entero). Con el objeto de que los dos equipos tengan una capacidad de competición similar, se pretende construir los equipos de forma que la suma de las puntuaciones de sus miembros sea la misma. Diseñad e implementad un algoritmo vuelta atrás para resolver, si es posible, este problema. Realizad un estudio empírico de la eficiencia del algoritmo y comparadlo con la de un algoritmo de fuerza bruta.

2.2. El Continental

En el juego (solitario) del Continental se colocan 32 piezas iguales en un tablero de 33 casillas, tal y como se indica en la figura siguiente (las "x" corresponden a posiciones no válidas):

x	x	o	o	o	x	x
x	x	o	o	o	x	x
o	o	o	o	o	o	o
o	o	o		o	o	o
o	o	o	o	o	o	o
x	x	o	o	o	x	x
x	x	o	o	o	x	x

Solo se permiten movimientos de las piezas en vertical y horizontal. Una pieza solo puede moverse saltando sobre otra y situándose en la siguiente casilla, que debe estar vacía. La pieza sobre la que se salta se retira del tablero. Se consigue terminar con éxito el juego cuando queda una sola pieza en la posición central del tablero (la que estaba inicialmente vacía).

Diseñad e implementad un algoritmo vuelta atrás que encuentre una serie de movimientos para llegar con éxito al final del juego.

2.3. Transporte de mercancías

Una empresa dispone de n centros de fabricación/distribución (cada uno situado en una localización diferente) y necesita abastecer n puntos de venta, situados en diferentes ciudades.

La distancia entre el centro de distribución i y el punto de venta j es $d(i,j),\ i,j = 1,\ldots,n$. La empresa desea abastecer cada punto de venta desde un único centro de distribución, y desea que la suma de las distancias entre cada centro de distribución y su punto de venta asignado sea lo más pequeña posible. Diseñar e implementar un algoritmo vuelta atrás que resuelva este problema. Mejoradlo usando alguna cota que mejore la poda. Realizar un estudio empírico de la eficiencia de los algoritmos (incluyendo también fuerza bruta).

2.4. Formas de sumar n

Diseñad e implementad un algoritmo de vuelta atrás que, dado un número natural n, devuelva todas las formas posibles en que un conjunto *ascendente* de números positivos sume exactamente n. Por ejemplo, si $n = 10$, la salida debería ser:

1+2+3+4
1+2+7
1+3+6
1+4+5
1+9
2+3+5
2+8
3+7
4+6
10

Probad si es posible con dos funciones de factibilidad diferentes, y haced un estudio empírico de su eficiencia.

2.5. Elección de árbitros

En una liga de fútbol participan n equipos (suponemos que n es par). En cada jornada se juegan $n/2$ partidos, que enfrentan a dos equipos, dirigidos por un árbitro. Existen m árbitros disponibles, siendo $m > n/2$. Cada equipo i valora a cada árbitro j con una puntuación $p(i,j)$ entre 1 y 10, indicando su preferencia por ese árbitro (un valor alto indica que le gusta el árbitro y un valor bajo que no le gusta). El objetivo es, dada una jornada concreta con unos emparejamientos entre equipos, asignar un árbitro distinto a cada partido, de manera que se maximice la puntuación total de los árbitros asignados, teniendo en cuenta las preferencias de todos los equipos. Esto se medirá mediante la suma de los productos de las puntuaciones asignadas por cada equipo al árbitro asignado a cada partido.

Diseñad e implementad un algoritmo de vuelta atrás para resolver el problema. Mejoradlo usando alguna cota que mejore la poda. Realizar un estudio empírico de la eficiencia de los algoritmos (incluyendo también fuerza bruta).

3. El problema del viajante de comercio

El problema del viajante de comercio ya se ha comentado y utilizado en la práctica sobre algoritmos voraces, donde se estudiaron métodos de este tipo para encontrar soluciones razonables (no óptimas necesariamente) a este problema. Si se desea encontrar una solución óptima

es necesario utilizar métodos más potentes (y costosos), como la vuelta atrás y la ramificación y poda, que exploren el espacio de posibles soluciones de forma más exhaustiva.

Así, un algoritmo de vuelta atrás comenzaría en la ciudad 1 (podemos suponer sin pérdida de generalidad, al tratarse de encontrar un tour, que la ciudad de inicio y fin es esa ciudad) e intentaría incluir como parte del tour la siguiente ciudad aún no visitada, continuando de este modo hasta completar un tour. Para agilizar la búsqueda de la solución se deben considerar como ciudades válidas para una posición (ciudad actual) sólo aquellas que satisfagan las restricciones del problema (en este caso ciudades que aún no hayan sido visitadas). Cuando para un nivel no queden más ciudades válidas, el algoritmo hace una vuelta atrás proponiendo una nueva ciudad válida para el nivel anterior.

Para emplear un algoritmo de ramificación y poda es necesario utilizar una cota inferior: un valor menor o igual que el verdadero coste de la mejor solución (la de menor coste) que se puede obtener a partir de la solución parcial en la que nos encontremos.

Una posible alternativa sería la siguiente: como sabemos cuáles son las ciudades que faltan por visitar, una estimación optimista del costo que aún nos queda será, para cada ciudad, el coste del mejor (menor) arco saliente de esa ciudad. La suma de los costes de esos arcos, más el coste del camino ya acumulado, es una cota inferior en el sentido antes descrito.

Para realizar la poda, guardamos en todo momento en una variable C el costo de la mejor solución obtenida hasta ahora (que se utiliza como cota superior global: la solución óptima debe tener un coste menor o igual a esa). Esa variable puede inicializarse con el costo de la solución obtenida utilizando un algoritmo voraz (como los utilizados en la práctica 2). Si para una solución parcial, su cota inferior es mayor que C entonces se puede realizar la poda.

Como criterio para seleccionar el siguiente nodo que hay que expandir del árbol de búsqueda (la solución parcial que tratamos de expandir), se empleará el criterio LC o "más prometedor". En este caso consideraremos como nodo más prometedor aquel que presente el menor valor de cota inferior. Para ello se debe de utilizar una cola con prioridad que almacene los nodos ya generados (nodos vivos).

Además de devolver el costo de la solución encontrada (y en su caso el tour correspondiente), se deben de obtener también resultados relativos a complejidad: número de nodos expandidos, tamaño máximo de la cola con prioridad de nodos vivos, número de veces que se realiza la poda y el tiempo empleado en resolver el problema.

3.1. Tareas a realizar

Construir un programa que utilice la técnica de ramificación y acotación para resolver el problema del viajante de comercio en las condiciones descritas anteriormente, empleando la función de acotación comentada, o alguna otra. Las pruebas del algoritmo pueden realizarse con los mismos datos empleados en la práctica 3 (teniendo en cuenta que el tamaño de problemas que se pueden abordar con estas técnicas es mucho más reducido que con los métodos voraces). La visualización de las soluciones también puede hacerse de la misma forma que en la práctica 3 (usando gnuplot).

Opcionalmente, construir también un programa que utilice vuelta atrás, pero utilizando también la función de acotación descrita anteriormente, y realizar un estudio experimental comparativo con el algoritmo de ramificación y poda.