Práctica 4: Backtracking y Branch & Bound
Mi pareja de prácticas

Dpto. Ciencias de la Computación e Inteligencia Artificial
E.T.S. de Ingenierías Informática y de Telecomunicación
Universidad de Granada

Algorítmica

Grado en Ingeniería Informática D

Índice de contenido

1. Introducción

Los objetivos de este guión de prácticas son los siguientes:

- Aplicar la técnica Backtracking para la resolución de problemas.
- Aplicar la técnica Branch & Bound estudiando cotas que permitan acelerar la exploración del árbol de soluciones. Con tal fin el alumno deberá tener presente la técnica Voraz que permitirá dar una primera aproximación a la solución del problema y además le ayudará a definir la cotas para la poda.

Los requisitos para poder realizar esta práctica son:

1. Haber estudiado el Tema 1: Introducción a la eficiencia de los algoritmos
2. Haber estudiado el Tema 3: Voraces
3. Haber estudiado el Tema 5: Backtracking y Branch & Bound

2. Objetivo.

El objetivo de esta práctica es establecer para un problema como se aplica las técnicas basadas en la exploración del árbol de soluciones : Backtracking, y Branch & Bound. Y por otro lado también se resolverá con la técnica de diseño Voraz.

3. Mi pareja de prácticas

Los alumnos de algoritmica del grupo D van a crear una empresa dedicada a buscar una pareja de prácticas a un alumno con objeto que la pareja formada tenga el menor numero de diferencias. Es decir sean lo más compatibles posibles.

Para obtener estas parejas, por cada alumno tenemos el nivel de afinidad sobre distintas aficiones o formas de vida. Esta información la tenemos recogida en los fichero alumnos<n>.txt.

Así por ejemplo el fichero alumnos_small.txt es el siguiente:

```
#Valores entre 0-10.1.Practica deportes, 2 #Frecuencia de lectura 3 Escucha musica 4 Gusta las
fiestas
6 4
5 10 9 10
10 5 5 0
0 2 6 1
10 0 0 0
5 5 5 0
1 2 3 3
```

La primera línea es un comentario que indica los aspectos encuestados a cada alumno. En este caso tenemos 6 alumnos y 4 características. Las cuestiones son respondidas dando un valor en el rango [0,10], 0 siendo la menor afinidad y 10 mayor afinidad. En este fichero las cuestiones realizados son:

1. Si práctica deportes
2. Frecuencia de Lectura
3. Escucha música

4. Si le gusta las fiestas.

Por ejemplo el primer alumno ha contestando 5 10 9 10 a cada una de las cuestiones. Por lo tanto le gusta la lectura, y las fiestas, algo menos la música y regular los deportes.

Con esta información de gustos por cada individuo podemos obtener una matriz de discrepancias entre individuos siendo calculada como:

$$D(i,j)=\sum_{k=1}^{ncar}|M(i,k)-M(j,k)| \forall\, i\neq j \qquad D(i,j)=\infty si \quad i=j$$

Esta matriz no es más que la suma de las diferencias en valor absoluto por cada una de las respuestas. Por ejemplo la matriz de discrepancia para el fichero alumnos_small.txt sería el siguiente:

```
Discrepancias:
inf 24  25  34  19  25
24  inf 15  10   5  17
25  15  inf 19  10   6
34  10  19  inf 15  17
19   5  10  15  inf 12
25  17   6  17  12  inf
```

Como se puede observar la discrepancia sobre sí mismo se ha puesto a infinito (inf).

Nuestro objetivo es obtener las mejores parejas de individuos que minimicen la discrepancia total. Así una solución $X=\{x_1, x_{2,}\cdots, x_n\}$ (siendo x_i el alumno que formará pareja con i) cumple que

$$minimiza \sum_{i=1}^{npersonas} D(i, X(i))$$

sujeto a que si la pareja de i es X(i)=j entonces la pareja de j es X(j)=i

3.1. Solución con voraces

En primer lugar este problema vamos a resolverlo con la técnica Voraz. Una función de selección trivial sería ir recorriendo los individuos y buscar entre los libres aquel que muestre menor discrepancia con él.

3.1.1. Programa Voraz

La ejecución del programa que resuelve este problema con voraces podría ejecutarse desde la línea de órdenes como:

```
prompt%  bin/mipareja_paracticas_voraz datos/alumnos_small.txt
```

Siendo los parámetros de entrada el fichero con el número de individuos, número de características y vector de gustos de cada alumno. La primera línea es un comentario (empieza con el carácter #).

Para este ejemplo la solución que da voraces con la función de selección comentada anteriormente es :

```
Algoritmo voraz...
Persona 1 asignamos a 5
Persona 2 asignamos a 4
Persona 3 asignamos a 6
Persona 4 asignamos a 2
Persona 5 asignamos a 1
Persona 6 asignamos a 3
Discrepancia Con voraces : 70
Tiempo 0.000105
_________________
```

El resultado de la ejecución muestra las parejas formadas y la discrepancia total encontrada, además del tiempo de ejecución.

Ejemplo

Suponed para cuatro individuos con 10 preguntas tal que cada fila es la contestación de un alumno a cada una de las 10 preguntas

$$\begin{bmatrix} 4 & 10 & 9 & 10 & 0 & 6 & 0 & 1 & 0 & 7 \\ 8 & 7 & 9 & 8 & 9 & 6 & 1 & 1 & 9 & 7 \\ 9 & 0 & 7 & 3 & 8 & 1 & 9 & 9 & 7 & 8 \\ 0 & 0 & 7 & 7 & 8 & 5 & 1 & 8 & 6 & 1 \end{bmatrix}$$

Con estos datos obtenemos la matriz de discrepancias:

$$\begin{bmatrix} \infty & 28 & 62 & 48 \\ 28 & \infty & 40 & 36 \\ 62 & 40 & \infty & 34 \\ 48 & 36 & 34 & \infty \end{bmatrix}$$

Aplicando voraces el alumno 1 escogería como pareja al alumno 2 ya que tiene la menor distancia 28. Por lo tanto al escoger 1-2 queda que 2 escoge 1. A continuación escoge el individuo 3 y solamente le queda seleccionar el 4 individuo. Obteniendo:

Discrepancia total: 124

Persona 1 con Persona 2--->28

Persona 2 con Persona 1--->28

Persona 3 con Persona 4--->34

Persona 4 con Persona 3--->34

3.2. Solución Backtracking

En este caso aplicaremos la técnica_Backtraking con dos versiones:

1. Se aplica backtracking cuando encuentra una solución (obtiene todos los emparejamientos).
2. Se aplica backtracking cuando la solución parcial tiene una discrepancia estimada mayor que la mejor solución actual.

Para estimar la discrepancia estimada podemos hacerlo como:

$$d_{estimada} = d_{actual} + CotaInferior(individuos\ libres)$$

d_{actual} es la discrepancia de las parejas hechas hasta el momento.

$CotaInferior(individuos\ libres)$ obtiene la discrepancia de emparejar los individuos libres aunque haya repeticiones y con la posibilidad de que si i escoge j, j no tiene la restricción de elegir i si este no es el individuo libre de menor discrepancia con él. Así podríamos tener que el individuo 1 escoge al individuo 3 porque tiene la menor discrepancia con 1, y 3 ahora escoge el individuo 6 porque tiene la menor discrepancia con 3.

Para resolver este problema usaremos un árbol de soluciones de tipo permutacional (ver ficheros Apermutacion.cpp y Apermutacion.h). Recordemos que la representación de este tipo de árbol es:

```
class Apermutacion{
  private:
   vector<int>datos;
   int level;
```

Para un árbol permutacional con n elementos datos[i] contiene valores entre 1 y n. Si datos[i]=0 significa que el elemento i aún no se ha considerado. Si datos[i]>0 significa que el individuo i escoge como pareja datos[i].

Para poder usarlo en este problema hemos añadido el siguiente método:

```
bool Apermutacion::CondicionProblem(){
  for (int i=0;i<=level && datos[i]>0 ;i++){
   if (datos[i]==(i+1)) return false;//asignar i a i no se puede
   if (datos[datos[i]-1]>0) //si se asigna 1-5 tiene que existir 5-1
      if (datos[datos[i]-1]!=(i+1)) return false;
  }
   return true;
}
```

Con este método nos aseguramos que el alumno i no se escoja así mismo como pareja. Y además si i ha escogido como pareja datos[i]=j entonces j ha escogido como pareja datos[j]=i.

3.2.1. Programa Backtracking

La ejecución del programa que resuelve este problema con la técnica Backtracking puede invocarse como:

```
prompt%  bin/mipareja_paracticas_backtracking datos/alumnos_small.txt
```

Siendo los parámetros de entrada el fichero con el número de individuos, número de características y vector de gustos de cada individuo. La primera línea es un comentario

Para la ejecución anterior el resultado sería:

```
Algoritmo Backtracking....
Numero de nodos recorridos 78 total nodos 1956 Porcentaje
3.98773
Discrepancia con Backtracking: 70
Persona 1 con Persona 5
Persona 2 con Persona 4
Persona 3 con Persona 6
Persona 4 con Persona 2
Persona 5 con Persona 1
Persona 6 con Persona 3

Tiempo 0.000722

_____________________________________
Algoritmo Backtracking con Poda....
Numero de nodos recorridos 61 total nodos 1956 Porcentaje
3.11861
Discrepancia con Backtracking y Poda: 70
Persona 1 con Persona 5
Persona 2 con Persona 4
Persona 3 con Persona 6
Persona 4 con Persona 2
Persona 5 con Persona 1
Persona 6 con Persona 3

Tiempo 0.000901

_____________________________________
```

En este ejemplo de ejecución se puede observar que la solución obtenida por ambos es la misma, pero el número de nodos recorrido cuando se aplica la Poda usando la discrepancia estimada se reduce de 78 a 61 nodos.

En el fichero “alumnos_small_debugbacktracking.txt” se puede ver una tabla que muestra los pasos en cada iteración para el algoritmo backtracking con poda. Esta tabla muestra

1. Permutación: se corresponden con las parejas hechas. Por ejemplo 2 1 0 significa que la persona 1 ha escogido la persona 2 y la persona 2 ha escogido la persona 1. La persona 3 aún no ha sido estudiada.
2. d_{actual}: es la discrepancia actual de las parejas hechas
3. best_discre: el mejor valor de la discrepancia total
4. $d_{estimada}$: la discrepancia estimada

3.3. Solución Branch & Bound

También vamos a resolver este problema usando la técnica Branch & Bound. Al igual que backtracking usará un árbol permutacional. Lo novedoso con esta técnica es que se ramifica por el nodo que tenga una discrepancia estimada menor hasta el momento. Además se usa una cota superior y una cota inferior de la solución que se puede llegar a obtener a partir de un nodo. La cota inferior, CI, se puede obtener como:

$$CI = d_{actual} + CotaInferior(individuos\,libres)$$

Siendo la CotaInferior la misma que hemos usado en la técnica de backtracking.

Y la cota superior CS se puede obtener como:

$$CS = d_{actual} + CotaSuperior(individuos\,libres)$$

$CotaSuperior(individuos\,libres)$ obtiene la discrepancia de emparejar los alumnos libres sin repeticiones y con la restricción de que si i escoge j, j tiene la restricción de elegir i .En este caso la CotaSuperior se puede implementar como la técnica Voraz sobre los individuos libres.

Finalmente la discrepancia estimada puede definirse como:

$$d_{estimada} = \frac{(CI + CS)}{2}$$

el promedio de la Cota Inferior y la Cota Superior. La discrepancia estimada en cada nodo, establece la prioridad de cada nodo de ser analizado. En este problema al ser un problema de minimización deberíamos explorar primero aquellos nodos que tienen la menor discrepancia estimada.

Al ser un problema de minimización hay que recordar que la cota C se obtiene como:

$$C = min(\{CS(j) \forall\, nodo\ j\ generado\}, \{Valor(s) \forall\, s\ solucion\ final\})$$

La poda ocurre en un nodo i si $CI(i) \geq C$

Así los pasos fundamentales del algoritmo Branch & Bound podrían ser los siguientes:

```
P:arbol permutacional
D:matriz de discrepancias
pq: cola de prioridad de nodos para guiar la exploracion del árbol de soluciones
CI: Cota superior
CS: Cota Inferior
DE: discrepancia actual
C:cota de poda
dactual: discrepancia actual
dmejor : mejor discrepancia

1.-Definir P con n elementos
2.-CI=CotaSuperior(todos los individuos)
3.-CS=Cotasuperior(todos los individuos)
4.-DE = (CI+CS)/2;
5.-C =CS;
6.-dactual=0;
7.-Crear un nodo n con (dactual,CI,DE,CS,P)
8.-Insertar n en pq
9.-repetir mientras pq no sea vacía
       n = nodo mas prioritario de pq
       Si n.CI<C
        por cada hijo h de n
              dactual = SumaDiscrepancias(h)
              CS= dactual+CotaSuperior(h)
              CI= dactual+CotaInferior(h)
              DE = (CS+CI)/2
              if (Level(h)==n-1 AND dactual<dmejor)//es hoja y mejor solucion
                   dmejor=dactual
                   if (C>dactual) C=dactual
                   mejorPermutacion=h.P;
              else
                if (CI<=C) //no se poda
                   crear un nodo nnuevo por el hijo h con (dactual, CI,DE ,CS,h.P)
                   insertar nnuevo en pq
                    if (C>CS) C=CS;
               end
       end
      end
10.- end repetir
```

3.3.1. Programa Branch & Bound

La ejecución del programa que resuelve este problema con Branch & Bound desde la línea de órdenes:

```
prompt%  bin/mipareja_paracticas _branchbound datos/alumnos_small.txt
```

Siendo los parámetros de entrada el fichero con el número de individuos, número de características y vector de gustos de cada individuo. La primera línea es un comentario

Para la ejecución anterior el resultado sería:

```
Algoritmo Branch & Bound ....
Numero de nodos recorridos 5 total nodos 1956 Porcentaje
0.255624
Discrepancia con Branch & Bound: 70
Persona 1 con Persona 5
Persona 2 con Persona 4
Persona 3 con Persona 6
Persona 4 con Persona 2
Persona 5 con Persona 1
Persona 6 con Persona 3

Tiempo 0.000444


_____________________________________
```

En el fichero “alumnos_small_debugbranch_bound.txt” se puede ver una tabla que muestra los pasos en cada iteración para el algoritmo Branch & Bound. Esta tabla muestra

1. La información de un nodo:
    1. Tupla. Se corresponde con la Permutación, que indica las parejas hechas. Por ejemplo 2 1 0 significa que la persona 1 ha escogido la persona 2 y la persona 2 ha escogido la persona 1. La persona 3 aún no ha sido estudiada.
    2. d_{actual}: es la discrepancia actual de las parejas hechas
    3. CI:cota inferior del nodo
    4. $d_{estimada}$: la discrepancia estimada
    5. CS: cota superior del nodo
2. C: la cota de poda.
3. best_dis: la discrepancia total de la mejor solución hasta el momento.

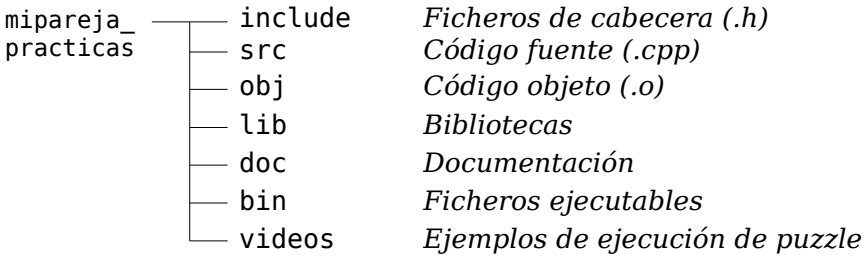
3.4. Eficiencia

La eficiencia vamos a obtener las técnicas Backtracking y Branch & Bound. En este caso vamos a realizar una eficiencia empírica, viendo el número de nodos que se explora y el tiempo que se dedica por cada nodo. Asi cada algoritmo mostrará como resultado el numero de nodos explorados frente al total.

4. Práctica a entregar

El alumno deberá empaquetar todos los archivos relacionados en el proyecto en un archivo con nombre “practica4.*tgz*” y entregarlo antes de la fecha que se publicará en la página web de la asignatura. Tenga en cuenta que no se incluirán ficheros objeto, ni ejecutables, ni la carpeta datos. Es recomendable que haga una “limpieza” para eliminar los archivos temporales o que se puedan generar a partir de los fuentes.

El alumno debe incluir el archivo *Makefile* para realizar la compilación. Tenga en cuenta que los archivos deben estar distribuidos en directorios*:*



Para realizar la entrega, en primer lugar, realice la limpieza de archivos que no se incluirán en ella, y sitúese en la carpeta superior para ejecutar:

```
prompt% tar zcv practica4.tgz practica4
```

tras lo cual, dispondrá de un nuevo archivo practica4.tgz que contiene la carpeta practica4 así como todas las carpetas y archivos que cuelgan de ella.