**Práctica 7**

*Eduardo Blanco Bielsa*

*UO285176*

**Implementación**

Se ha diseñado un heurístico que asegura que ningún nodo es podado, de forma que proporciona exactamente el mismo resultado que Backtracking sin balanceo.

En primer lugar, se ha creado un método al que se llamará posteriormente tanto para la ejecución como para la medición de los tiempos. En este método de la clase PromediadoImagen se llamará al método branchAndBound de la clase BranchAndBound, pasándole como parámetro un AvgNode con el dataset, la profundidad 0 y un arraylist vacío que se corresponderá con las imágenes de cada nodo.

Una vez hecho esto se guardará en los atributos half1 y half2 la mejor solución y se actualizará el contador al número total de nodos recorridos.

Texto

Descripción generada automáticamente

Se ha creado la clase **AvgNode** donde:

* El método **calculateHeuristicValue()** calcula en función de si el nodo es solución o no el valor del heurístico.
* El método **expand()** devuelve un arraylist de Node con las correspondientes soluciones.
* El método **isSolution()** devuelve si un nodo es solución (nodo hoja), comparando su profundidad con la longitud del array de imágenes (dataset).

Texto

Descripción generada automáticamente

Pantalla de computadora con letras

Descripción generada automáticamente con confianza media

Texto

Descripción generada automáticamente

Para conocer el valor del contador del branchAndBound se ha modificado ligeramente el método branchAndBound añadiendo un atributo contador que se modifica en cada llamada y en el bucle while:

Captura de pantalla de computadora

Descripción generada automáticamente

**Medidas**

Para 6 imágenes se obtienen los siguientes resultados:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **n** | **Tiempo\_BT\_sin\_balanceo** | | **Contador BT** | | **Tiempo\_BnB** | **Contador BnB** |
| 2 | 8 | | 13 | | 67 | 13 |
| 3 | 17 | | 40 | | 17 | 40 |
| 4 | 47 | | 121 | | 50 | 121 |
| 5 | 125 | | 364 | | 192 | 364 |
| 6 | 321 | | 1093 | | 675 | 1093 |
| 7 | 996 | | 3280 | | 2247 | 3280 |
| 8 | 3052 | | 9841 | | 5611 | 9841 |
| 9 | 9324 | | 29524 | | **heap space** | **heap space** |
| 10 | 29626 | | 88573 | | **heap space** | **heap space** |
| **ZNCC\_Backtracking** | | **ZNCC\_ BnB** | |
| 0,001746 | | 0 | |
| 0,029555 | | 0,023124 | |
| 0,021497 | | 0,033691 | |
| 0,038218 | | 0,035221 | |
| 0,051588 | | 0,047109 | |
| 0,059882 | | 0,053151 | |
| 0,054618 | | 0,053285 | |
| 0,070974 | | **heap space** | |
| 0,076252 | | **heap space** | |

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Para 8 imágenes se obtienen los siguientes resultados:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **n** | **Tiempo\_BT\_sin\_balanceo** | | **Contador BT** | | **Tiempo\_BnB** | **Contador BnB** |
| 2 | 9 | | 13 | | 73 | 13 |
| 3 | 16 | | 40 | | 16 | 40 |
| 4 | 44 | | 121 | | 52 | 121 |
| 5 | 130 | | 364 | | 184 | 364 |
| 6 | 328 | | 1093 | | 559 | 1093 |
| 7 | 1017 | | 3280 | | 1919 | 3280 |
| 8 | 3123 | | 9841 | | 5515 | 9841 |
| 9 | 9493 | | 29524 | | **heap space** | **heap space** |
| 10 | 29320 | | 88573 | | **heap space** | **heap space** |
| **ZNCC\_Backtracking** | | **ZNCC\_ BnB** | |
| 0,004199 | | 0,001799 | |
| 0,028478 | | 0,03692 | |
| 0,027522 | | 0,027824 | |
| 0,034373 | | 0,040192 | |
| 0,047012 | | 0,054616 | |
| 0,054909 | | 0,058058 | |
| 0,060142 | | 0,060269 | |
| 0,074447 | | **heap space** | |
| 0,079416 | | **heap space** | |

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

**Respuestas a preguntas y conclusiones:**

* En ambos casos se comprueba que el **contador** de Backtracking y BranchAndBound es el mismo, por tanto, nuestro heurístico es correcto.
* Al llegar a un determinado tamaño, se produce un “**heap space**” en el BranchAndBound, es decir, que Java no tiene suficiente memoria para seguir calculando datos del heurístico.
* Los **tiempos** son “similares”, aunque el BranchAndBound tarda más tiempo, por tanto, posee un peor rendimiento que el backtracking normal.
* Ambos tienen un **Zncc** bastante similar, por tanto, deducimos que obtienen una solución muy similar o casi idéntica.