PRACTICA SESIÓN 7 *Raúl Fernández España*

Todas las medidas han sido tomadas en una maquina con 64 Gb de Ram con un procesador i7-10700K

Tal y como vimos en la práctica anterior (Práctica 6) los algoritmos voraces proporcionan una solución subóptima. Backtracking es capaz de encontrar la solución óptima, pero a costa de una complejidad temporal exponencial. A pesar de introducir la condición de balanceo para reducir la densidad del árbol de expansión (o implícito) mediante una poda selectiva de nodos, apenas fuimos capaces de procesar poco más de una docena de imágenes.

Tras implementar el código necesario para R&P los resultados de la toma de mediciones son los siguientes:

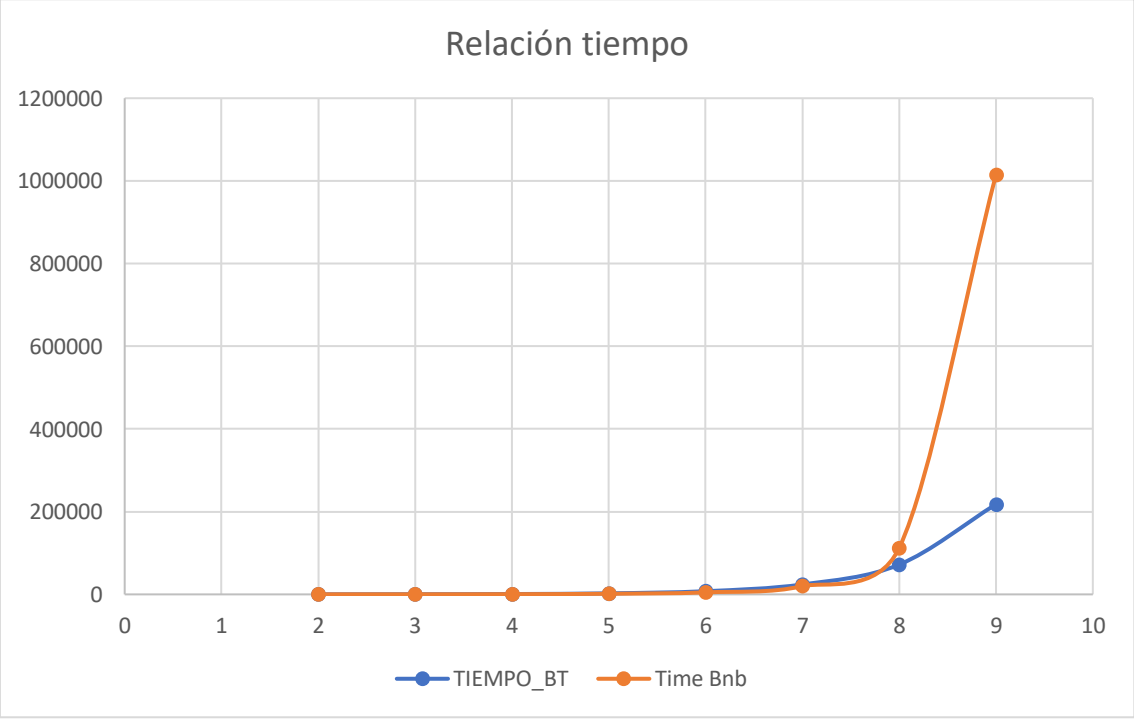
N	TIEMPO_BT_BALANCEO	Time Bnb	Nodes Bnb	ZNCC_BT_BALANCEO	ZNCC BNB
2	125	115	13	0.025883357971906662	0.008772714994847775
3	321	198	40	0.025883357971906662	0.027497705072164536
4	647	473	121	0.02425304800271988	0.02441886067390442
5	1526	1450	364	0.04134903848171234	0.03896237909793854
6	3617	4910	1093	0.042642273008823395	0.043444279581308365
7	8380	19940	3280	0.0540006197988987	0.05588269978761673
8	19605	112422	9841	0.049358393996953964	0.05746633931994438
9	44786	1015428	29518	0.06835588812828064	0.06793230772018433

Aquí tenemos los tiempos anteriores de Backtracking al lado de los nuevos tiempos con R&P

Aquí tenemos los tiempos sin balanceo

N	TIEMPO_BT
2	103
3	320
4	819
5	2493
6	7666
7	23961
8	72035
9	217995

A si queda la comparación de tiempos comparando con los resultados de la practica anterior



Como el árbol que se crea no es implícito si no parcialmente explicito por lo que al no tener que almacenar en memoria los nodos consume menos recursos haciendo que tarde menos tiempo. La ramificación y poda

requiere de más tiempo de ejecución debido a que calculamos todas las posibiidades.