Activity 1. Branching heuristic

At each step you pick the next unvisited node whose edge‑cost plus a cheap “best‑case” estimate of the remaining tour is smallest. (Is taking the “best” unvisited node given a heuristic).

Before descending, you check if the current cost plus that estimate still could meet your target (within the tolerance). If not, you prune that branch.

This “best‑bound” ordering and pruning makes you find a valid full tour much faster than pure backtracking.

Activity 2. Part D: Table

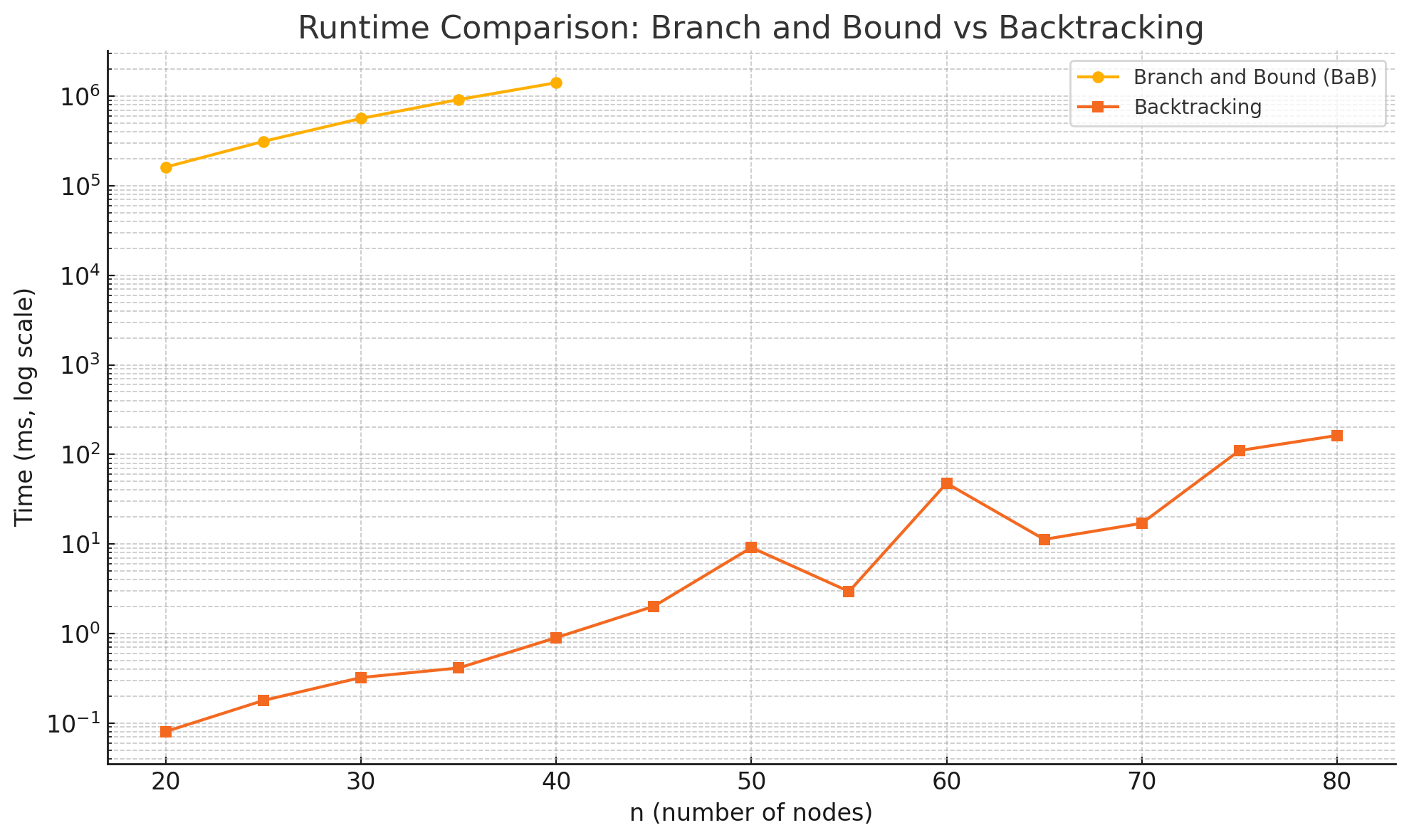
We measured average runtimes (in milliseconds) for two algorithms solving the NullPath problem as the number of nodes n increases:

|  |  |  |
| --- | --- | --- |
| **n** | **t BaB (ms)** | **t Backtracking (ms)** |
| 20 | 161,774 | 0,080467 |
| 25 | 313,011 | 0,178919 |
| 30 | 564,334 | 0,32252 |
| 35 | 917,279 | 0,41224 |
| 40 | 1413,045 | 0,899015 |
| 45 | OoT | 2,014633 |
| 50 | OoT | 9,117259 |
| 55 | OoT | 2,94463 |
| 60 | OoT | 47,619499 |
| 65 | OoT | 11,25175 |
| 70 | OoT | 17,012901 |
| 75 | OoT | 110,366254 |
| 80 | OoT | 162,678025 |

**Key observations:**

1. **Initial advantage of Branch & Bound:**  
   For n <= 20n, BaB completes in the hundreds of milliseconds, while backtracking finishes in under a millisecond.
2. **Rapid blow-up of BaB:**  
   Between n=20 and n=40, BaB’s runtime grows by almost an order of magnitude (from 0.16 S up to over 1.4 S) demonstrating its exponential sensitivity to problem size.
3. **Timeouts beyond n=40:**  
   Starting at n=45, BaB fails to complete within our cutoff (“OoT”), whereas backtracking still manages to solve up to n=80 (albeit more slowly).
4. **Backtracking’s steadier growth:**  
   Although backtracking is far slower than BaB for small n, its runtime increases more gradually, peaking at ~163 mS for n=80, and never timing out in our experiments.

The log-scale comparison below



vividly shows BaB’s steep rise and early collapse versus backtracking’s gentler curve.