|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **method** | **heuristic** | **domain\_heuristic** | **nodes** | **time** |
| forward | SEQUENTIAL | SEQUENTIAL | 4071.166667 | 4.524630785 |
| backtracking | RANDOMISED | RANDOMISED | 4449.833333 | 0.275529544 |
| forward | SEQUENTIAL | RANDOMISED | 5922.5 | 5.551896453 |
| backtracking | SEQUENTIAL | RANDOMISED | 16074.83333 | 0.82543985 |
| backtracking | SEQUENTIAL | SEQUENTIAL | 29017.5 | 1.627212564 |
| backtracking | RANDOMISED | SEQUENTIAL | 29017.5 | 1.412077824 |
| forward | RANDOMISED | RANDOMISED | 468790 | 1111.91979 |
| forward | RANDOMISED | SEQUENTIAL | 645729.6667 | 1439.581292 |

Moja implementacja Forward Checkingu jest bardzo nieoptymalna pod kątem czasowym, więc czas wykonania forward będzie zasadniczo dużo dłuższy.

Przy sekwencyjnej heurystyce średnio forward checking odwiedza mniej węzłów, natomiast backtracking odwiedza więcej węzłów

Natomiast backtracking wymaga mniej zasobów, dlatego jest przy mojej implementacji szybszy. Jednak byłaby to rzecz, którą można optymalizować

Natomiast przy losowości sytuacja się odwraca. Losowość wartości powoduje, że forward checking musi bardzo często się wracać, więc generuje bardzo dużo

Przez to czas wykonania był ogromny

Porównanie herystyk to pokazuje: Średnio dużo częściej odwiedza się węzły, a czas jest dłuższy (przez forward checking)

Porównanie Heurystyk wyboru wartości – losowa była delikatnie lepsza

Wniosek: Forward Checking jest lepszą metodą, jednak wymaga lepszej implementacji