Each cell value shows the average run time for k = 3, and then k = 4 separated by a comma. Average run times were gathered from 10 trials at each value of n. All times are in milliseconds.

| **Avg. run time** | **backtracking** | **w/ forward check** | **w/ MAC** | **min-conflicts** |
| --- | --- | --- | --- | --- |
| **n = 5** | 10.6342, 0.1873 | 2.6738, 0.2495 | 1.6734, 0.4109 | 4.9321, 0.1012 |
| **n = 10** | 15.3714, 0.434 | 4.2625, 0.6717 | 2.4299, 0.5055 | 7.3927, 0.0583 |
| **n = 25** | 43.128, 0.1498 | 13.264, 0.2674 | 3.3032, 1.491 | 6.7624, 1.6728 |
| **n = 50** | 3046.8897, 2602.2645 | 92.97, 63.3797 | 12.8612, 2.2866 | 10.1966, 3.2802 |

From the data, we can see that backtracking w/ no inference is extremely time consuming at higher values for n, while min-conflicts tends to have consistent run times for increasing values of n. At lower values for n, the difference in run time is still there, but in practice it is not very noticeable. The two fastest algorithms from the experiment were backtracking w/ MAC and the min-conflicts algo. These results are pretty consistent with what I expected, and any errors would probably have come from my implementation. I know that the timing of each algorithm in my implementation may not be exact, as well as some other errors that may have arisen. One thing that is noticeable is that the run times tend to be shorter with k = 4, rather than k = 3 which is probably due to the fact that many times there is no solution for k = 3, so the algos run through longer before coming to that conclusion.