CSP

Assignment 4: Constraint Satisfaction Problems

Problem 1 Brute force

Table

The first row indicates the run and the second row indicates the amount of calls.

run_0	run_1	run_2	run_3	run_4	run_5	run_6	run_7	run_8	run_9
20	38	167	170	174	52	139	73	28	168

Mean: 102.9, Standard deviation: 62.758983420702414

When running 10 iterations of the n-queens problem, we get 102.9 mean and 62.8 standard deviation from the mean. This tells us that we have an average run with 103 calls and this could deviate 63 calls from the mean. In other words, solving this problem with backtracking is not really efficient.

Doing this multiple times will eventually result in different answers, but in general we can conclude that the answer differs depending how efficient our choices are made. Because our choices are random, our efficiency is also random.

Problem 2 Forward checking

No LCV and no MRV setting on with 5 iterations

Mean: 3534.2, Standard deviation: 5913.984152836393

No LCV and MRV enabled with 5 iterations

Mean: 67.0, Standard deviation: 0.0

LCV and no MRV enabled with 5 iterations

Mean: 1374.4, Standard deviation: 2015.6246277519037

LCV and MRV enabled with 5 iterations

Mean: 73.0, Standard deviation: 0.0

Conclusion

Looking at this data we can conclude the following. Performing forward checking with no ordering will result us in to a mean of 3500 and a deviation of 6000. This means it still is not efficient.

When enabling MRV, we get a huge improvement. Our mean is now 67 and we have no deviation, in other words this result is guaranteed.

When we only enable LCV and no MRV, we also see an improvement but the improvement is not as big as MRV and we still have a large deviation.

Now looking at both combined, we see a weird phenomenon. Instead of improving our result we get a negative effect on the amount of calls. We do have a consistent amount of calls, meaning we do have an efficient way of finding the result making use of these ordering.

So in conclusion, in our case only enabling MRV will give us better results than if we enable both.

Problem 3 AC3

MRV and LCV enabled

Mean: 55.0, Standard deviation: 0.0

No MRV and no LCV enabled

Mean: 144.0, Standard deviation: 72.29107828771127

Conclusion

From this run we can conclude that running AC3 with ordering is way more efficient. If we compare our answer to Problem 2, we can see that we have less calls. This is because arc consistency will be checked for the whole set instead of one by one.

Even if we compare the result with no ordering enabled, the results are drastically different. The runs with ac3 are way better than if we use forward checking.

Problem 4 Sudoku

Forward checking: 6027 calls [00:56, 100.38 calls/s]
AC3: 2860 calls [01:10, 88.40 calls/s]

Conclusion

Solving the hard problem concludes us with the following. Forward checking can solve it in less than a minute but takes more calls in total. If we compare it with AC3, it is more computing intensive but it takes way less calls than forward checking.

Problem 5 Feedback

Checker for the different problems because there is not a real way to check if you are doing it right. Even though some of the problems are random, and the calls may differ, there should be a way to validate the output. For instance a checker for every type of problem, if it is solved. If we look at the NQueens problem, a check if the solution that is outputted is right or wrong instead of a normal print.

This is a faster check when we have n = 50 for instance.