Artificial Intelligence 1
Lab 3: Constraint Satisfaction Problems

Edward Boere (s3005690)

LC[7]

May 30, 2018

1: Heuristic and propagation technique analysis

1.1 Solving a small set of equations

option	visited states
none	63254
-iconst 1	63254
-iconst 2	63254
-mrv	63254
-deg	63254
-arc	4
-fc	254

The solver finds one solution for the given CSP description: $A = 3$, $B = 7$ and $C = 2$.

The flag -iconst 1 has no effect on the number of visited states. Because there are no unary constraints on any of the variables, making them node consistent will change nothing. One could argue this flag will take even longer than when it would not be used.

Making the first set of variables arc consistent with eachother using the -iconst 2 flag also has no effect on the number of visited states. This could be because after the first variable is chosen from the reduced domain, the next chosen variable is not chosen from the arc consistent domain, but from the originally defined domains again. Because we have no way to be sure that we have found all solutions, we must try all other permutations. Thus we will still have to visit all possible states.

The variables A, B and C have the same domain, so choosing the variable with the fewest possible options will not make a difference. This is why the -mrv heuristic has no effect.

The -deg heuristic also does not change the number of visited states. Every variable is involved in the same number of constraints, so the solver executes in the same exact way it would without the flag.

As can be seen from the table, running the solver with the -arc flag visits the least states. The -arc flag enables an algorithm that applies arc consistency, like the -iconst 2 flag. However, -arc applies arc consistency every time a value is assigned to a variable, instead of only as a preprocessing step (Maintaining Arc Consistency). Using MAC with the final constraint $A * B * C = 42$ will greatly reduce the domain of the three variables, as their multiple has to equal an integer. When we only take the final constraint in consideration, the domain of A, B and C equals:

$$D_{\mathrm{A,B,C}} = \{1, 2, 3, 6, 7, 14, 21, 42\}$$

This reduced domain consists of 5% of the original domain $A, B, C \leftarrow [0..250]$. The reduced domain becomes even smaller when the other two constraints are processed by the MAC algorithm.

The second flag is the -fc flag. It is not nearly as effective as MAC, but it reduces the number of visited states considerably. Forward checking tracks the remaining legal values for unassigned variables, and terminates search when any variable runs out of legal values. A reason why FC visits more states than MAC is that it checks only the constraints involving the current variable, while MAC takes all variables into account. This will result in more incorrect states.

As the -iconst 1, -iconst 2, -mrv and -deg flags have no impact on the number of visited states, combining them has no effect. The two remaining flags, -arc and -fc, cannot be combined.

1.2 Market

option	visited states
none	10311
-iconst 1	10311
-iconst 2	10311
-mrv	25761
-deg	10311
-arc	19
-fc	115

The solver finds five solutions for the given CSP description. This problem is very similar to the previous, except three unary constraints are introduced for the price of the fruit.

We can see the -iconst 1 and -iconst 2 flags have no effect on the number of visited states. Even though in this case there are unary constraints. We would expect the -iconst 1 flag to reduce the number of visited states, because it would mean we would not have to try the values in the declared domain for the fruit price values. If this would not have been the case, it could lead to 1000^3 different states for only the first three constraints. It could be that the solver has some built in functionality to treat a unary constraint as an assignment. The arc consistency preprocessing still has no effect for the same reason mentioned in problem 1.1.

The Most Restricted Value heuristic increases the number of visited states considerably. This is because the values with the smallest domains are the number of fruits. There are a large number of solutions for the fourth constraint $nr\_orange + nr\_grape + nr\_melon = 100$. By using the -mrv flag, we prioritize computing all these different solutions. When these variables have been assigned a value, we compute all different combinations of the fruit prices to find a solution, even though their constraints are unary.

Using the -deg flag has no effect on the number of visited states because the fruit prices are computed first, so each variable is involved in the same number of constraints with unassigned variables.

The MAC algorithm is again the most efficient propagation technique for the same reasons as mentioned in problem 1.1.

Forward checking also has some success, but it can assign values to the first variables without taking the domain of the other variables in consideration, which will result in the last variable running out of options. This is why it visits more states.

Combining the -iconst 1, -iconst 2 and -deg flags has no effect, as they do not impact the number of visited states. Combining -mrv with -fc results in 15857 visited states, but when we include node or arc consistency as a preprocessing step, we come back to 115 states, the same result we get with only the -fc step. This is because when we use either preprocessing step, we assign the fruit prices first. The -iconst 1 or -iconst 2 flag negates the negative effect of the -mrv flag.

When we combine the -arc flag with the -mrv flag, the solver does not find a solution within 10 minutes of computing time. If a preprocessing step is performed, the negative effect of the -mrv flag is negated and only 19 states are visited again.

1.3 Chain of trivial equations

The equations model that all variables must have the same value. Their domain is:

$$D_{\mathrm{A..Z}} = [0..99]$$

It is trivial that there exists 100 solutions for this problem. To find this result 2601 states were visited. $26 * 100$ states to assign each variable a hundred different values, and one state in which no variable was assigned a value. No heuristic or propagation technique can reduce the number of visited states.

Running the solver with chainA.csp gives us one solution, the only solution possible is where all variables equal 42. The solver visits $2502 = 2601 - 99$ states. It seems like the solver computes the constraints from bottom to top, it chooses a value, assigns every variable from Z to A this value, and then comes to the constraint $A = 42$, which is not true for 99/100 cases.

We can use arc consistency to solve this inefficiency, if the system of chainA.csp is made arc consistent, the domain of all variables will be reduced to $D_{\mathrm{A..Z}} = 42$. The first variable that will be assigned will be the only solution of the problem. By using the -iconst 2 flag, solving chainA.csp only visits 27 states. If only the -arc flag is used, the first variable is chosen, afterwards the system is made arc consistent. This still results in 2502 visited states.

The chainZ.csp takes the constraint $A = 42$ into consideration first, and then all the following variables only have one possible solution. So chainZ.csp always visits 27 states.

All flags apart from -iconst 2 have no effect on the three provided problems.

1.4 Cryptarithmetic puzzles

The solver indeed finds the solution given in the assignment. For the first time we see the -mrv having a positive effect on the number of visited states. This is because when we choose the most restricted variable and we conclude that it is inconsistent, we can eliminate a larger fraction of states than we could have with a variable with more options.

The -arc flag is the most effective, this is because there are four constraints of a higher order. This means we can shrink the domains of the variables considerably.

Forward checking is also successful in reducing the number of visited states. There are multiple higher order constraints, so the number legal values the variables can be assigned reduces fast.

When -mrv is combined with -arc or -fc, we have a heuristic to choose which variable is processed first, and we have a propagation technique to reduce the number of visited states. In general, combining heuristics can lead to better performance.

cryptarithmeticmoney	
option	visited states
none	1349854
-iconst 1	1349854
-iconst 2	1349854
-mrv	46264
-deg	1349854
-arc	399
-arc -mrv	40
-fc	7440
-fc -mrv	127

cryptarithmeticonze	
option	visited states
none	142472
-iconst 1	142472
-iconst 2	142472
-mrv	2869
-deg	142472
-arc	58
-arc -mrv	72
-fc	1916
-fc -mrv	182

cryptarithmeticeighty	
option	visited states
none	X
-iconst 1	X
-iconst 2	X
-mrv	1108534
-deg	X
-arc	21417
-arc -mrv	1227
-fc	1897027
-fc -mrv	205118

cryptarithmetichurts	
option	visited states
none	9836
-iconst 1	9836
-iconst 2	9836
-mrv	67956
-deg	9836
-arc	92
-arc -mrv	189
-fc	707
-fc -mrv	7666

As can be seen from the tables, the -arc or -mrv -arc options visit the least states. We can also see that in general, using -mrv and -fc together results in less visited states.

1.5 Finding the first 20 primes

option	visited states
none	1048576
-iconst 1	1048576
-iconst 2	21
-mrv	1048576
-deg	1048576
-arc	27570
-fc	524289

The solver returns one solution. The most efficient flag is -iconst 2. This is because for the array $p[20]$, each index can become one of 70 different values. For each of these values, $19 * 70 = 1330$ different tests must be computed. This means there are at least 1862000 unique states for the array p. If we can shrink the domain $p[i]$ a lot of these tests will not have to be computed.

1.6 Solving Sudokus

sudokusmiley		sudokuhard	
option	visited states	option	visited states
none	2020824	none	X
-iconst 2	1427	-iconst 2	356499
-iconst 2 -mrv	1541	-iconst 2 -mrv	5901253
-mrv	2020824	-mrv	X
-arc	1179487	-arc	X
-arc -mrv	207055	-arc -mrv	X
-arc -iconst 2	82	-arc -iconst 2	4099
-arc -mrv -iconst 2	82	-arc -mrv -iconst 2	2383
-fc	1435983	-fc	X
-fc -mrv	209377	-fc -mrv	X
-fc –iconst 2	362	-fc –iconst 2	103978
-fc -mrv -iconst 2	90	-fc -mrv -iconst 2	7855

For solving a sudoku, making the problem arc consistent and maintaining arc consistency is the most important heuristic available. In this way we can use the sudoku rules to shrink the domains of the variables, instead of randomly guessing and checking if it is right.

If there is state in which there exists no square with a domain with only 1 value, we will have to make an educated guess. It is generally smartest to guess the square with the fewest options; when the choice turns out to result in an inconsistent system, we can discard a larger fraction of possible states. For this reason the -mrv flag has a positive effect when combined with a propagation technique.

1.7 *n*-queens problem (again)

n queens	solutions	visited states
4	2	9
5	10	46
6	4	41
7	40	240
8	92	655
9	352	2622
10	724	7401

For this problem the -arc flag is the most effective, adding heuristics does not make a difference. This problem is purely about the domains of the remaining queens.

At the initial state there are no queens on the board, thus using -iconst 1 or -iconst 2 has no effect.

The ”quality” of a queens position is not directly connected to the freedom she has; in the end every queen will have 0 freedom. For this reason the -mrv and -deg flags have no substantial effect.

2: Writing CSP descriptions

2.1 Constraint graph

variable	solution 1	solution 2
A	3	4
B	3	4
C	4	2
D	2	3
E	1	1

The solver found two solutions for the given CSP description. Because it is a problem with a lot of binary constraints, making the problem arc consistent and maintaining arc consistency is the most effective strategy. Depending on how the -iconst 2 and -arc flags are implemented, using the -mrv flag could speed up the process of attaining arc consistency. When the most restricted variable is compared with other variables first, their domains shrink to a size relative to the smallest domain at the start. This will not reduce the number of visited states, but may reduce the number of computations necessary to attain arc consistency.

2.2 Magic squares

The "magic total" for an NxN magic square is given by the following formula:

$$((N * N * N) + N)/2$$

For $N = 4$, the magic total is 34. Using the -mrv, -deg and -arc flags, 7040 solutions are found. This includes vertical, horizontal and diagonal mirrors and rotations.

2.3 Boolean satisfiability (SAT)

9 solutions were found for the boolean expression. No flag was necessary to solve the problem, as the domains of all the variables consisted of only two options. The -arc flag could be used for a bigger problem to reduce the number of visited states.