Assignment 3

TDT4136 - Introduction to Artificial Intelligence
Fall 2023

Elena Filippini
Francesca Grimaldi

06/10/2023

1 Sudoku boards as CSPs

The purpose of this assignment is to implement a general solver for **Constraint Satisfaction Problems**, using *backtracking search* and the arc-consistency algorithm AC-3. This is used to solve Sudoku boards of different difficulties (`easy`, `medium`, `hard` and `very hard`).

We completed the functions in the `code-handout` [Cod] based on the pseudo-code provided by the book [RN22], and we tested our implementation using the given boards.

The following subsection contains details to explain our choices for what concerns some of the functions to implement. All the results of the code execution are provided in Sections 2-5: for each board, it is possible to see the solution as well as the number of times that the `backtrack` function was called and, eventually, failed. Lastly, Section 6 contains our comments on the outcomes.

1.1 Implementation choices

As it is possible to see from our code, we decided to use the **Minimum Remaining Values** (MRV) heuristic to choose the next unassigned variable to consider in the context of the `backtrack` function (`Select-Unassigned-Variable`). Since the program is designed to solve Sudokus, we opted for the variable with the fewest values in the domain to be assigned next.

As per the `Order-Domain-Values` mentioned in the book pseudo-code, we decided to access the values in the order in which they are stored in the dictionary.

2 Easy board

Task:

		4		3			5	
6		9	4					
		5	1			4	8	9
				6		9	3	
3			8		7			2
	2	6		4				
4	5	3			9	6		
					4	7		5
	9			5		2		

Figure 1: Easy board

Program solution:

```
[Running] python -u "c:\Users\fragr\git\sudoku-csp\Assignment.py"
Easy sudoku
Solution:
7 8 4 | 9 3 2 | 1 5 6
6 1 9 | 4 8 5 | 3 2 7
2 3 5 | 1 7 6 | 4 8 9
------+-------+------
5 7 8 | 2 6 1 | 9 3 4
3 4 1 | 8 9 7 | 5 6 2
9 2 6 | 5 4 3 | 8 7 1
------+-------+------
4 5 3 | 7 2 9 | 6 1 8
8 6 2 | 3 1 4 | 7 9 5
1 9 7 | 6 5 8 | 2 4 3
Calls to the Backtrack function:  1
Fails of the Backtrack function:  0
```

Figure 2: Program's solution for board A

As shown in figure 2, the function backtrack is called 1 time, and it never fails.

3 Medium board

Task:

				3			4	
1		9	7					
			8	5	1		7	
		2	6		7	8	3	
9		6		1		2		7
	3	1	5		2	9		
	1		3	6	9			
					5	7		3
	9			7				

Figure 3: Medium board

Program solution:

```
Medium sudoku
Solution:
8 7 5 | 9 3 6 | 1 4 2
1 6 9 | 7 2 4 | 3 8 5
2 4 3 | 8 5 1 | 6 7 9
------+-------+------
4 5 2 | 6 9 7 | 8 3 1
9 8 6 | 4 1 3 | 2 5 7
7 3 1 | 5 8 2 | 9 6 4
------+-------+------
5 1 7 | 3 6 9 | 4 2 8
6 2 8 | 1 4 5 | 7 9 3
3 9 4 | 2 7 8 | 5 1 6
Calls to the Backtrack function:  2
Fails of the Backtrack function:  0
```

Figure 4: Program's solution for board A

As shown in figure 4, the `backtrack` function is called 2 times, and it never fails.

4 Hard board

Task:

1		2		4				7
				8				
		9	5			3		4
			6		7	9		
5	4						2	6
		6	4		5			
7		8			3	4		
				1				
2				6		5		9

Figure 5: Hard board

Program solution:

```
Hard sudoku
Solution:
1 5 2 | 3 4 6 | 8 9 7
4 3 7 | 1 8 9 | 6 5 2
6 8 9 | 5 7 2 | 3 1 4
------+-------+------
8 2 1 | 6 3 7 | 9 4 5
5 4 3 | 8 9 1 | 7 2 6
9 7 6 | 4 2 5 | 1 8 3
------+-------+------
7 9 8 | 2 5 3 | 4 6 1
3 6 5 | 9 1 4 | 2 7 8
2 1 4 | 7 6 8 | 5 3 9
Calls to the Backtrack function:  7
Fails of the Backtrack function:  2
```

Figure 6: Program's solution for Hard board

As shown in figure 6, the `backtrack` function is called 7 times, and it fails 2 times.

5 Very hard board

Task:

		1			7			
6			4			3		
				3			6	4
3	8			7	6			
							3	6
2	7			1	5			
				2			5	1
7			1			2		
		8			9			

Figure 7: Very Hard board

Program solution:

```
Veryhard sudoku
Solution:
4 3 1 | 8 6 7 | 9 2 5
6 5 2 | 4 9 1 | 3 8 7
8 9 7 | 5 3 2 | 1 6 4
------+-------+------
3 8 4 | 9 7 6 | 5 1 2
5 1 9 | 2 8 4 | 7 3 6
2 7 6 | 3 1 5 | 8 4 9
------+-------+------
9 4 3 | 7 2 8 | 6 5 1
7 6 5 | 1 4 3 | 2 9 8
1 2 8 | 6 5 9 | 4 7 3
Calls to the Backtrack function:  56
Fails of the Backtrack function:  43
```

Figure 8: Program's solution for Very Hard board

As shown in figure 8, the `backtrack` function is called 56 times, and it fails 43 times.

6 Comments about the results

The `backtrack` function repeatedly chooses a variable and tries all the values in its domain, and tries to extend each value through a recursive call [RN22].

The results obtained for Sudoku puzzles of varying difficulty levels align with this theory. In simpler puzzles, only a few attempts are typically needed, while the highly challenging ones demand a more exhaustive search. This can also be verified trying to solve the Sudoku, in the case of the first board, it is sufficient to look at the values stored in the rows and columns to find the correct values, whereas in the very hard board, multiple attempts are often required.

AC-3 algorithm is considered effective to solve easy Sudoku puzzles, and indeed the collected data show that increasing the computational complexity leads to a significant increase of the function calls, as well as the amount of failures of the function. As described in sections 2 and 3, the number of function calls experience a slightly change between easy and medium boards, passing from 1 to 2, while the function fails remain equal to zero. From medium to hard, the changes appear more evident, as the number of calls is increased more than twice in 4. Analogously, the amount of failures goes from 0 to 2. The most significant change is seen from hard 5 to very hard 6: the number of function calls pass from 7 to 56, of which 43 failures.

In light of these findings, we expect that further increasing the board's difficulty would result in a task that is too complex to be completed in a reasonable amount of time and function calls using AC-3, requiring other algorithms with more complex strategies.

References

[Cod] Code-handout, available on the Blackboard page of the course.

[RN22] Stuart Russell and Peter Norvig. *Artificial Intelligence, A Modern Approach*. Pearson, 2022.