Wrocław University
of Science and Technology

Introduction to Artificial Intelligence

Laboratory 2: Constraint Satisfaction Problem

By:

Sindy Kola 245903

Introduction

My formulation of the solution of both problems other than satisfying the requirements of a CSP, by its implementation resembles what is known as D-CSP(Dynamic CSP). In such an implementation, as the name suggest from what we are used to seeing in dynamic programming, we have constraints, but not in an exactly "explicit way." In this way, we check constraints with domain cutting, leaving in the domain only elements that pass those constraints.

Problem 1: Sudoku

Definitions of Sudoku as a CSP Problem

1. Variable: each blank cell on a 9x9 board
2. Domain: digits from 1 - 9, D = {1, 2, 3, 4, 5, 6, 7, 8, 9}
3. Constraints:
    1. The same digit can not repeat on a row
    2. The same digit can not repeat on a column
    3. The same digit can not repeat on a block(3x3 subboard)

After implementing the program in Python, I obtain the following results, where I shall compare in the form of a table, the execution times for every solution, in each difficulty level and then measure an average time/ level.

Level	Problem ID	Average Execution time/level
1	6, 7, 8, 9, 10	0.567
2	11, 12, 13, 14, 15	1.742
3	16, 17, 18, 19, 20	0.184
4	21, 22, 23, 24, 25	0.552
5	26, 27, 28, 29, 30	0.269
6	31, 32, 33, 34, 45	1.436
7	36, 37, 38, 38, 40	0.339
8	41, 42, 43	2.884
9	44, 45, 46	0.931

Remark:

This solution can be further expanded in attempt to improve it, by adding another heuristic. This way, we might start the solution of Sudoku from a place with the least blanks or a place in the board which, in other words, is more populated. This would lead to faster domain pruning and therefore faster solution.

Problem 2: Jolka - Fill in Puzzle
Definitions of Jolka as a CSP Problem
1. Variable: sequence of "_"
2. Domain: words obtained from the file
3. Constraints:
    1. Each word can be used only once
    2. We only use words with matching length(words should fit between #)
    3. Directions where intersection of two words occurs must contain one common letter

*Solution for Puzzle 1, where **0.000247** is the execution time*
##DAG##
##ARID#
EDIT#OR
VESICLE
ON#CLEF
#SILO##
##OED##
0.00024700164794921875

*Solution for Puzzle 2, where **0.000678** is the execution time*
ZKOQS#ZGN
QEDT#ZOST
OG###WGKT
##ZGLLTR#
QZZTFZOCT
#DOFOFU##
LTTF###GA
RGUL#HOTL
ZGH#LVOFU
0.0006780624389648438

*Solution for Puzzle 4, where **0.0035** is the execution time*
BHNNCN#PPAG#ABDOHNJOAMD#ABCI#LJI
HMLKLMLDIKE#POMHMKJCOFEGCAAA#FJL
FBJGDCHLOFLGOHPEND#FBDADHODPLHIB
EL#MILPNBOLONFFMDIOGCPK#OBDCJAIE
HOPHPIF#APLOFDPDIEEJELKB#EIHCLNO
LIAHLJMKNNNNCJ#PFGMIHNGAAADLBEFN
JG#HKBGDDBMOCIALOMA#GBKIDPMNCJKD
KNMJBIJAM#INGBNODNJPICNMIL#GGMMI
MKBCCN#FLLMDHM#DPCJENMDBMHMBNF##
FLAGONPNHLKLBPB#KLEIF#MEEDNOHELO
NPHAP#PCNDJDLLNEOPPLCEDLLC#ELEPI
DPEGBPKEFE#MLJFLLONLLCON#GEPENOK
##EPDBPAPKBFFIHFP#BGMDFDAFELMMKB
ELMID#JFOJMLILDMMMLHFO#OHNGOGLOE
MAAMGPODBKKL#CCKDJENMEALEGGEN#EG
NNHBKKMHFAGKIHJHP#DPMHAMGIPKAFFL
ANADMOFNCGCMH#FHBGIPCFLB#BBACKAF
FCOMEBCI#BNKA#CIMOPLBAMMHFCNK#BL
HOL#PNOLOCHKMD#DOAHDIFDFINCCIFOA
CCH#NLGKEHCEKC#MMLHKCPMDJHMNKAMM
0.003492116928100586

Solution for Puzzle 3
Execution time was too long to measure.

Remark:
In order to make this solution faster I added an additional Heuristic to my algorithm. First I fill in a word horizontally and then check and try to fill in all the vertical crossings/intersections of that word.

Conclusions:

Sudoku

As we can see from the results obtained, we yield the smallest average execution times, for difficulty levels of 3, 5 and 7. Compared to results from other experiments performed where the solution did not involve backtracking and forward checking, execution times are shorter. This implementation however, also proves to be quite fast.

Jolka

Similar conclusions can be drawn in this case aswell with the exception that the added remark makes in the final execution times. Not taking into account Puzzle 3, the addition of that Heuristic improved dramatically the times taken to find the solution of such puzzles.