Constraint Satisfaction Problem

Aim The aim of this lab is to investigate specific CSP algorithms, and to reflect on the nature of the constraint satisfaction problems and the algorithms used to solve them.

About the lab This lab consists of four parts, and it can be done by <u>at most three persons</u>.

Note that you do not have to do any coding in these two tasks.

Preparation Read Chapter 6 Constraint Satisfaction Problems[1] of the textbook Artificial Intelligence: A Modern Approach.

- Task 1-2 require to run specific CSP algorithms with and without heuristics, and to compute their performance with respect to the nature of the CSP problem. These tasks require two Python libraries search.py and csp.py that are compatible with Python2.7 (not python 3).

  - In the lab rooms, Python2.7 is installed on every computer in the folder
    `C:\Python27\ArcGIS10.3\python.exe`

  - If you work with your own computer. Download Python2.7[2], and install it in `c:\python27`.

  - Download and unzip the library `libraryLab2.zip`[3] in one of your folders, say `myfolder`. This library contains:

    (i) the files sudoku.py, queensS.py and queensCSP.py

    (ii) the folder `aima` with the Python implementation of the aima library.

  - To run for example sudoku.py, open a CMD window in the folder `myfolder` and execute the command:
    `C:\Python27\ArcGIS10.3\python.exe sudoku.py`   (if you work in any lab room)
    `C:\python27\python.exe sudoku.py`   (if you work with your computer)

- Tasks 3-4 are about to implement a CSP algorithm.

- -

Lab examination demonstrate the tasks you solved during any scheduled lab session. The course grade is as follows:

Tasks solved	Grade
2 tasks	3
3 tasks	4
4 tasks	5

---

[1]Available at `http://dellacqua.se/education/courses/tnm096/material/chapters/ch6.pdf`

[2]Downloadable from `https://www.python.org/downloads/release/python-2712/`

[3]Available at `http://dellacqua.se/education/courses/tnm096/material/labs/libraryLab2.zip`

Task 1 - Sudoku

The implementation for sudoku provided in this lab includes calls to the algorithms: Depth-first graph search, AC-3, Backtracking search, and Min-conflicts. The code provides the predefined problem specifications:

```
    . . 3 | . 2 . | 6 . .        4 1 7 | 3 6 9 | 8 . 5        1 . . | . . 7 | . 9 .
    9 . . | 3 . 5 | . . 1        . 3 . | . . . | . . .        . 3 . | . 2 . | . . 8
    . . 1 | 8 . 6 | 4 . .        . . . | 7 . . | . . .        . . 9 | 6 . . | 5 . .
    - - - + - - - + - - -        - - - + - - - + - - -        - - - + - - - + - - -
    . . 8 | 1 . 2 | 9 . .        . 2 . | . . . | . 6 .        . . 5 | 3 . . | 9 . .
    7 . . | . . . | . . 8        . . . | . 8 . | 4 . .        . 1 . | . 8 . | . . 2
    . . 6 | 7 . 8 | 2 . .        . . . | . 1 . | . . .        6 . . | . . 4 | . . .
    - - - + - - - + - - -        - - - + - - - + - - -        - - - + - - - + - - -
    . . 2 | 6 . 9 | 5 . .        . . . | 6 . 3 | . . 7        3 . . | . . . | . 1 .
    8 . . | 2 . 3 | . . 9        5 . . | 2 . . | . . .        . 4 . | . . . | . . 7
    . . 5 | . 1 . | 3 . .        1 . 4 | . . . | . . .        . . 7 | . . . | 3 . .

          easy                          harder                        hardest
```

To run Sudoku, open a CMD window in the folder lab2 and type `c:\python27\python.exe sudoku.py`. Run the four algorithms above on each of the predefined sudoku puzzles. To do so, see the comment on line 29 of `sudoku.py`.

Answer the following questions.

(a) Which of the algorithms Depth-first graph search, AC-3, Backtracking search and Min-conflicts works in a timely manner (say a couple of minutes max) and which doesn't? Explain your results in terms of the capabilities of the algorithms and the nature of the problems.

(b) What effect does configuring the settings for Backtracking search have on the results? Try the following.

   - Open the file `aima\csp.py` on line 227, and remove the comment symbol `#` for the option you want to use; whether BT, BT+FC, BT+MRV or BT+FC+MRV[4].

   - Set the variable `select_unassigned_variable` to:

      * `first-unassigned-variable` or
      * `=mrv`[5]

   - For each of the two options above, set the parameter `inference` to:

      * `no_inference`
      * `forward-checking`

   Which of these settings work best for sudoku?

---

[4]Select exactly one alternative and comment all the other ones.
[5]mrv=minimum-remaining-values

Task 2 - NQueens

The implementation for the NQueens problem provided in this lab includes calls to the algorithms Depth-first graph search, AC-3, Backtracking search, and Min-conflicts, and allows you to specify n.

The file queensS.py contains the implementation of depth-first tree search algorithm, and queensCSP.py contains the implementation of AC3, Backtracking search and Min-conflicts algorithms.

Run each of the algorithms with various values for n. Answer the following questions.

(a) How large can n be for each of the algorithms? Why?

(b) What Backtracking search settings work the best? Why?

(c) How many steps does Min-Conflicts require to do its work?

(d) Compare the nature of the heuristics deployed in traditional state-based search and constraint-based problem solving.

Task 3 - Classroom Scheduling as a CSP

One of the succesfull application of Min-Conflicts algorithm is to solve scheduling problems[6].
This task is about to write an implementation of Min-Conflicts to solve the problem below.
You can use any programming language.

Classroom Scheduling
You are in charge of assigning classes to classrooms and times. You have to schedule three
classrooms: TP51, SP34 and K3. Classes start on the hour. You can only assign classes to the
hours of:

9 am	10 am	11 am
12 pm	1 pm	2 pm	3 pm	4 pm

You must schedule 22 classes:

MT101	MT102	MT103	MT104	MT105	MT106	MT107
MT201	MT202	MT203	MT204	MT205	MT206	
MT301	MT302	MT303	MT304			
MT401	MT402	MT403				
MT501	MT502					

Your schedule must not violate any of the following requirements:

1. Two classes cannot meet in the same classroom at the same time.

2. Classes whose first digit are the same (like MT102 and MT107) cannot be scheduled at
   the same time because students might take them both in one semester. There is one
   exception to this rule; MT501 and MT502 can be scheduled at the same time since
   students can only take one of them.

Write a display function that prints the schedule you have computed.

	TP51	SP34	K3
	- - - -	- - - -	- - - -
9			MT101
10	MT205	MT303	MT402
11	MT201	MT304	MT107
12	MT202	MT102	MT302
1	MT502	MT206	MT105
2	MT204	MT104	MT501
3	MT106	MT301	MT403
4	MT401	MT203	MT103

---

[6]It was introduced to schedule observations for the Hubble Space Telescope. This algorithm reduced the
time taken to schedule an entire week of observations from three weeks to 10 minutes.

Task 4 - Classroom Scheduling with Preferences

Solve the scheduling problem of Task 3 again but with the addition of these preferences:

- Prefer schedules that do not schedule classes at 9 am, 12 pm and 4 pm.

- Prefer schedules where MT501 and MT502 are scheduled at 1 pm or 2 pm.

Discussion
In my solution I tried to integrate scheduling and preferences directly in one algorithm. It seems to be a difficult problem and my solution does not work so well either (inefficient and mediocre schedules).

Tip To accomplish this task, do not modify your implementation of Min-Conflicts implemented in Task 3. Do the following.

1. Write another function that calls Min-Conflicts repeatedly.

2. For each solution found count the number of preferences that are not satisfied. Save the solution with the smallest count of unsatisfied preferences.

3. Search for the best solution by repeatedly calling Min-Conflicts for a max number of times.

4. At the end return the solution found with the smallest count of unsatisfied preferences.