Computational Intelligence

Project 2: Reasoning

Umberto Grandi, Dennis Wilson, Sylvain Cussat-Blanc
{umberto.grandi, dennis.wilson, sylvain.cussat-blanc}@irit.fr

Spring 2018

Overview

Final presentations are scheduled for 5 April from 11:00 to 12:30 in the Arsenal Library (salle 1). Each team will be composed of 3 people, and will have 15 minutes to present the following:

- Justifications of their modelling choices (see details below).
- An explanation of the two algorithms used (details of the first few steps, justification of the heuristics and the constraint solver used).
- (Optional) A detailed explanation of any of the optimisation done.
- Their results comparing the time of execution of the two algorithms, and a discussion of their implementation.

Problem description

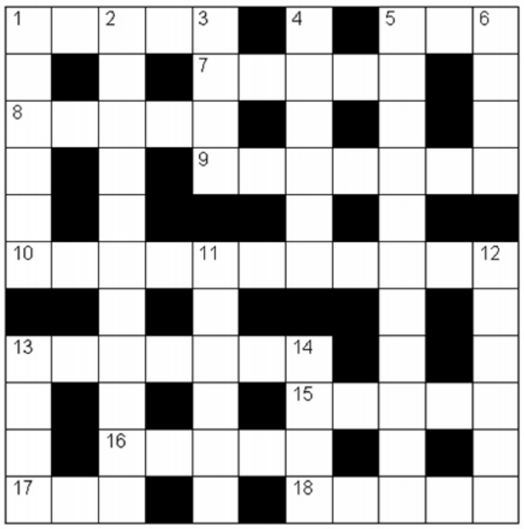
Consider the problem of *creating* crosswords. You are given the grid at the following page, which you cannot modify, and you have access to a list of words in English on the Moodle (`words.txt`). You have to fill the white cells in the grid with letters, such that each column and row are formed by words in the dictionary.

Assignments

Modelling. Formulate the problem of creating crosswords with the given grid:

- As a standard search problem. Analyse the search space (number of states, branching factor of search tree, any other parameter that you might think useful).
- As a constraint satisfaction problem. Variables are letters or words? Write the constraint graph. How many variables? How many constraints? Analyse the form of the constraints.

What is the best formulation in your opinion? Provide one slide in which you compare the two proposed problem formulations.



Solving. Consider the following two approaches to find a solution:

- Devise a suitable heuristic for this problem. Show whether it is an admissible or consistent heuristic. Show on the slides the first few steps of A* with your proposed heuristic.
- Consider forward-checking and min-conflicts. Show on the slides the first few steps of applying the chosen algorithm to your CSP formulation. Which one do you think will be faster? Optimise the algorithms in any way you may find useful (optimisation is optional).

In your opinion, this problem is better solved as a search problem or as a CSP? Provide one slide in which you compare the two approaches theoretically (time and space complexity, and any other parameter you might find useful).

Implementing. Implement the problem of solving the crossword grid in Phyton, both using your A* heuristic and as a CSP problem. Compare the time execution of the two approaches, making sure that you run the problem a sufficient number of times on the same machine, and averaging the results. Which one is faster?

Resources

The file `words.txt` gives you a list of admissible English words. We provide three python files that can help you visualise your results:

`char_grid.py`: draws the filled-in crossword by providing a 2D matrix of character indices.

`word_grid.py`: draws the filled-in crossword by providing two lists of words; the over direction words, then the down direction words.

`utils.py`: an example of how to search the dictionary for a specific word, and how to check if a list of letters is a valid word.

Both `char_grid.py` and `word_grid.py` show an example of drawing a crossword with a random grid. The Internet abounds with code for A* search. To solve CSPs in phyton we advise you to install the following python module on your machines:

- Go to `https://github.com/python-constraint/python-constraint` and download the module, where documentation is also available
- Extract the downloaded zip file. Copy the 'constraint' folder to the working directory of your project
- On Spyder, import the module `constraint`

Good work!