Introduction

Crossword constructors increasingly rely on computational tools to fill grids and finish puzzles. A solution to a crossword puzzle can be viewed as a constraint satisfaction problem, where each fill is a variable and the crossing points and lengths are constraints to be satisfied. With this framework, it is possible to build crosswords more complex than could be generated by human constructors. We apply methods used to solve weighted constraint satisfaction problems to generating three-dimensional crossword puzzles.

Why is this hard?

Figure 1 shows two graphs where the nodes in the graphs are words in a puzzle and an edge connects two nodes if the words cross. The graph on top is for a New York Times Thursday puzzle and the graph on the bottom is for a 4x4x4 3D puzzle. The NYT puzzle has densely connected pockets that are connected to each other by only one or two crossing constraints. Thus, these areas can be filled almost independently. This is not the case for our 4x4x4 puzzle.

Weighted CSPs

Filling a crossword can be considered as a constraint satisfaction problem, where we want to select some number of fills as variables such that they satisfy all the length and crossing constraints. Every variable has a large search space as every combination of letters could feasibly be an acceptable fill. Crossword constructors make use of large corpuses of previously used fills with associated “goodness” scores to assist in puzzle construction. Using these scores we can evaluate partially completed CSPs, turning our problem into a weighted CSP. We can then prune our search tree once we find any acceptable solution. This method is called branch & bound. The initial solution need not be very good in order to vastly improve our computation time.

We want to sort our potential fills so that some solution can be found. This is done by giving fills an adjusted score based on the frequency of their letters and their goodness score and then bounding solutions based on the total goodness score.

Limited Discrepancy Search

One of the issues with branch & bound is that the search can become stuck in some large branch of the tree. This is usually bad because we believe that our heuristics will allow us to find the best solutions in any branch relatively quickly. One way out of this is to stop searching a branch after some number of “discrepancies”. In our application, a discrepancy occurs when we try to place a fill that has already been placed in the puzzle. When the number of discrepancies caused by a fill exceeds some threshold, we leave the branch of the tree that first involved that fill.

Results

The methods we apply allows us to construct 4x4x4 puzzles in fractions of a second. We also constructed a few 5x5x5 puzzles, but this required much more computation time and had significantly lower scored fills. One of the best scoring 4x4x4 puzzles is shown in Figure 3.

Acknowledgement

Thank you to Matt Ginsberg for providing the code for Dr.Fill which inspired many of the techniques used in this project. Also thank you to Dragomir Radev for advising the research.

Generating 3D Crosswords as a Weighted Constraint Satisfaction Problem

Michael Menz

Yale College, New Haven, CT.