CPS331 Lecture: Constraint Propagation as an Alternative to Search

last revised September 17, 2018

Objectives:

1. To introduce constraint propagation

Materials:

1. Projectable and handout of Garden Puzzle
2. Prolog generate and test + "human" solution process
3. Projectable of Sudoku Puzzle
4. Sudoku solver program; demo.sudoku puzzle

I. Introduction

A. Recall that, in our introduction to search, we mentioned that a key problem in search is something called "combinatorial explosion". Basically, what we are dealing with is that state space size can grow very rapidly - e.g. we showed that, for a search with 13 steps and 3 alternatives at each node the state space contains over 1 million nodes! If we go to just 20 steps, the size increases to over 3 billion! When we hit 21 steps, the search space size is greater than the present population of the world.

B. Although heuristic methods can help a great deal by helping us to focus on the alternatives that are most likely to lead to a solution, good heuristics are not necessarily easy to find.

C. In certain cases, we can minimize search or even avoid it altogether by using a strategy called constraint propagation. While the problems to which this applies are limited in number, the benefits to be gained are enormous.

D. One weakness of standard AI search techniques, like those we've been looking at, is that they are often quite different from the ways humans solve the problem.

1. Recall what we saw in the case of solving the "Garden Dilemma Puzzle" using a search.

   HANDOUT + PROJECT

a) The Prolog program we looked at used a naive generate and test strategy.

PROJECT

b) Recall that we saw that, while this works for a small problem, it quickly falls prey to combinatorial explosion (e.g. with our program a problem with 15 people would take 25 times the age of the universe to solve!)

c) For this particular sort of problem, I'm not aware of any good heuristics that would allow us to use an informed search (and I suspect this is not a hot research area!)

2. Of course, a human wouldn't solve the problem this way at all.

PROJECT human approach

That is, a human makes use of <u>constraints</u> to fairly quickly home in on a solution. For example, since we are told that each person bought a different tool, we can eliminate that tool as a possibility for the other four people once we learn which person bought it.

3. When a problem lends itself to the application of constraints, it is often possible to arrive at a solution very quickly.

II. **Constraint Propagation in Sudoku**

A. We will use, for further examples, the familiar Sudoku puzzle.

1. Explain the puzzle

2. Solving a Sudoku can be regarded as a search problem in which the states correspond to partially filled in puzzles.The start state is the given initial state

a) The goal state is a state in which all squares are filled in

b) The operators are "write a particular number in a particular square"

c) Example: consider a typical puzzle:

PROJECT Sudoku Solver program with demo puzzle

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

(1)The available operators (consistent with the rules of the game) are

Put a 6 in the upper-left corner
Put an 8 in the upper-left corner
(1-5, 7, and 9 are ruled out by others in same row/column/block)
...
Put a 3 in the lower-right corner
Put a 4 in the lower-right corner
Put a 9 in the lower-right corner
(1, 2, and 5-8 are ruled out by others in same row/column/block)

(2)In fact, for this particular state, there are 175 legal operators!

3. However, if we tried to use a conventional search technique to solve this puzzle, we would quickly run into combinatorial explosion

B. The key to solving a puzzle like this is to recognize that, in a legal puzzle, at any time, there will always be one or more squares whose value is constrained to a single value.

   Example: In this puzzle, the third square in the first row is constrained to be an 8 by the fact that no other value is possible.

   - 3, 7, 2 are eliminated by other values in the same row
   - 5, 6 are eliminated by other values in the same column
   - 4, 1, 9 are eliminated by other values in the same block

   Example: In this puzzle, the second square in the first row is constrained to be a 5 by the fact that this is the only square in the block where a 5 can go. (5 cannot go in the other vacant squares in the block due to a 5 elsewhere in the same column)

   Example: In this puzzle, the first square in the second row is constrained to be a 7 by the fact that this is the only square in the block where a 7 can go. (7 cannot go in the other vacant squares in the block due to a 7 elsewhere in the same row.)

C. This leads to the following strategy for solving this puzzle:

   1. Initial setup:

      a) Associate with each square the set of legal values that can appear in that square, based on constraints propagated from other squares in the same block, row, and column.

Example: The legal values for the vacant squares in the upper-left block are

{ 6, 8 }	{ 5, 8 }	{ 8 }
{ 3, 6. 7. 8 }	-	-
-	-	{ 3, 8 }

b) Associate with each block a list of squares where each value that has not been used in that block can occur [which can be extracted from the above]

Example: In the upper-left block

1 has been used
2 has been used
3 { middle-left, lower-right }
4 has been used
5 { upper-middle }
6 { top-left, middle-left }
7 { middle-left }
8 { top-left, top-middle, top-right, middle-left, lower-right }
9 has been used

c) Do something similar for each row

d) Do something similar for each column

2. Now perform the following process repeatedly until the puzzle is solved

a) Select a set which contains just one element (if there is none, we're stuck)

b) Give the corresponding square the appropriate value

c) Propagate constraint resulting from this choice to other cells

(1) The value just given to the cell can be removed from the sets of possible values for other cells in the same row, column, and block

(2) The square can be removed from the sets of possible locations for unused values in the row, column, and block of which it is a part

3. DEMO: Run Sudoku Solver program with demo.sudoku

   a) Click OK to get to point where solution has been set up

   b) Click Show Details to show possible values for each cell

      Observe:

      - If a cell has only one possible value, it is shown in yellow. (This includes cells which represent the only possible location for the value in a row, column, or block.)

   c) Step through first few steps of solution, showing how constraint is propagated

      Observe:

      - The cell whose value is being fixed is highlighted in red

      - Cells with a single possible value are selected in the order in which this fact was discovered - hence the 7 in the second row being the first to be fixed (only possible location for 7 in its block)

      - A square which is about to have its set of possible values reduced is highlighted in purple and then on the next step is reduced to new value.

D. Actually, this process is not totally sufficient

Demo: Click solve - note how solution is stuck. (This was actually a fairly challenging puzzle).

1. This puzzle can be solved by noting another constraint. In the middle block, the only place where an 8 can occur is in the top row. Hence, one of these two squares will eventually be an 8 - in which case the 8 in the middle-right block top row is not actually possible, forcing this square to be a 4.

2. Manually enter and show how puzzle can now proceed to solution.

3. This sort of constraint could also be incorporated programatically - but I didn't choose to do so in this program.

III. Constraint Satisfaction Problems (CSPs)

A. One could argue that the Sudoku puzzle is still something of a "toy problem". In fact, it is a simple example of a class of problems known as constraint satisfaction problems.

B. A CSP involves a graph having a set of nodes connected by a set of arcs (edge).

For example, in Sudoku the 81 squares are the nodes and there are arcs connecting each node to each other node in the same row, to each other node in the same column, and to each other node in the same 3 x 3 block.

C. A CSP is characterized by a set of constraints of two types - node consistency and arc consistency. Node consistency constraints specify what values are possible for a given node and arc consistency constraints specify what must hold about the values in the nodes at each end of the arc.

For example, in Sudoku those squares which contain values at the start constitute node consistency constraints, and the requirement that nodes in the same row, column, or block have different values give rise to arc consistency constraints.

D. There exist algorithms that solve CSP's expressed this way very quickly. (A CSP-solver could solve a Sudoku puzzle expressed in terms of constraints in less than 0.1 second)

1. For some CSPs (e.g. Sudoku) it is possible to solve the problem without any search by simply propagating constraint along the arcs until the puzzle is solved, because there is a unique solution forced by the constraints.

2. Other CSPs require limited search with backtracking because the constraints themselves do not force a unique solution.