6.034 Recitation 3 - Constraint Satisfaction Solution
April 6, 2007

1. Sudoku-BOT

You are involved as a consultant for a startup that plans to build robots that can automatically solve the Sudoku puzzles in the daily paper. An example of such a puzzle is visible in Figure 1.

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

Figure 1: A Sudoku puzzle.

The rules of the game are that each unfilled box must be filled in with a digit within a particular range (in this case 0..9), such that no two digits appear in the same *row*, *column*, or *region*. Thus, in a solved sudoku puzzle, each row, column and region will have a complete set of digits {0..9}. You plan to apply your newly acquired knowledge about Constraint Propagation algorithms to make your robot able to solve Sudoku puzzles with exceeding speed and grace.

a. What do variables represent in your CSP algorithm?

Two possibilities here. The most obvious one will likely be to have each variable in our CSP represent a square to be filled-in. However, another possibility is to represent entire rows, columns, and regions as the variables of our CSP.

b. What are the domains of the variables represented by your CSP algorithm?

If a variable is used for each square, the domains of the CSP variables are the set of legal values to be filled in for that square, namely the digits {0..9} for a full Sudoku puzzle. If a variable is used for each column, row, and region, the set of all possible columns, rows, or regions, respectively (of which there will be 10! values in the entire

set, each.)

c. What are the constraints among the variables of your algorithm? What order are they?

For variables representing each square, there will be 3 N-ary AllDiff constraints: one connecting each variable to the rest of the variables in the same row, one to the rest in the same column, and one to the rest of the squares in each region. To transform these higher-order constraints into binary ones, we can make pairwise-diff constraints from each square variable to the other square variables in each row, column, and region.

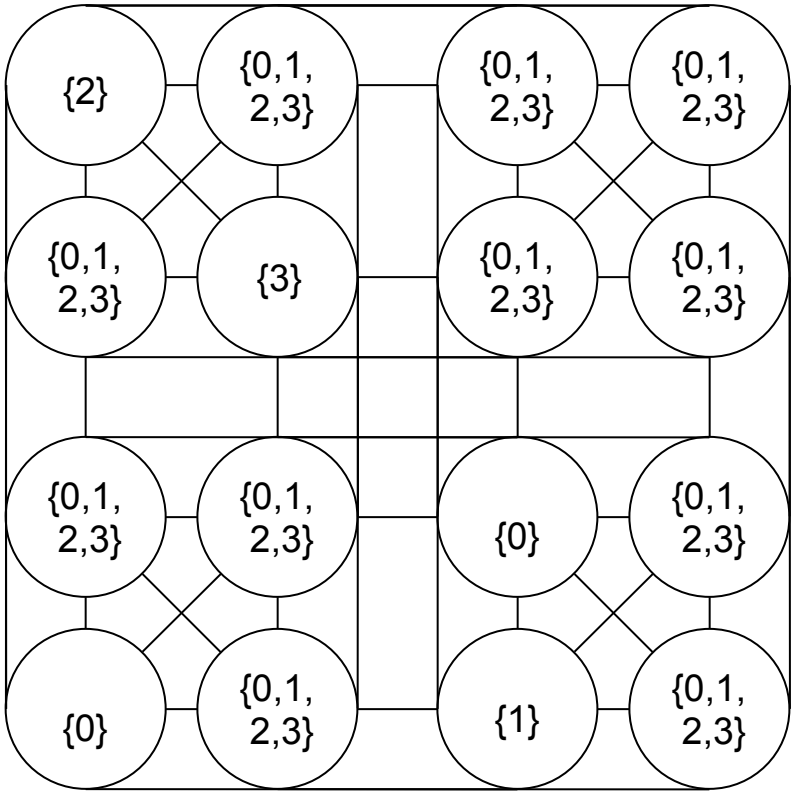
If variables are chosen to represent each row, column and region, any row variable will be constrained by N binary constraints with column variables, and √N binary constraints with region variables. Each column variable is also constrained similarly. Each region variable is constrained by √N constraints with row variables and √N constraints with column variables. These variables will need to agree upon the values being shared among them.

2			
	3		
		0	
0		1	

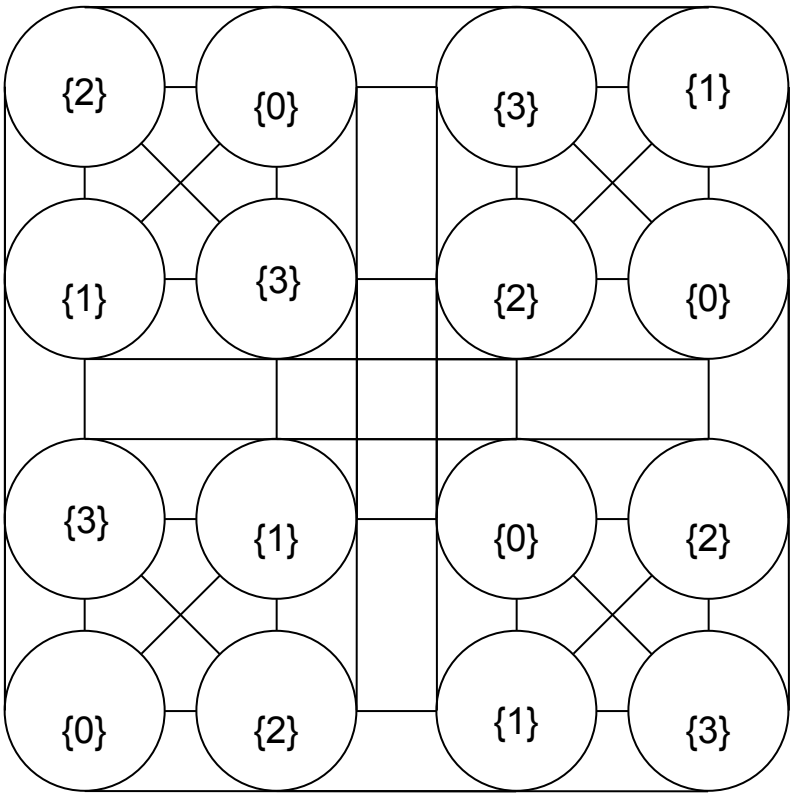
Figure 2: Mini-sudoku. Values of squares constrained to range {0,1,2,3}

d. Draw a constraint hypergraph for the mini-sudoku puzzle in Figure 2, then solve the puzzle using pure arc-consistency (if possible)

Assuming that we are using the first representation (a variable per square) discussed above with only binary constraints, the hypergraph would appear as in the following figure (note that lines tangent to nodes connect every pair of such nodes):



Using repeated pure arc-consistency, we achieve:



e. For a general Sudoku puzzle of N by N squares (where regions are X by XN) , how many constraint arcs are there?

For the representation of having a variable per square, each row, column, and region form fully connected graphs of N nodes. Therefore there are N(N-1)/2 connections in each row, column, and region. Since there are N rows, N columns, and N regions, there will be a total of 3N (N(N-1)/2), or (3N^2) * (N-1)/2 consistency arcs. Unfortunately, though, by adding the region constraints, we have double-counted row and column arcs that appear in the region constraints and the row/column constraints. We can subtract this value out by observing that each region contributes √N * (√N choose 2) row arcs and √N * (√N choose 2) column arcs; therefore we have a surplus of N regions * N(√N - 1) arcs. Subtracting this, we get a total of: (3/2)N^3 - N^2√N - (1/2)N^2 arcs.*

For the representation of having a variable per row, column, and region, there will be √N binary arcs tying each region variable to the column variables overlapping each region, and √N binary arcs tying each region variable to the row variables overlapping each region, and N^2 binary arcs between the row variables and column variables that intersect at a single square. Since there are N regions, there will be a total of N^2 + 2N√N arcs.

f. For a general Sudoku puzzle of N by N squares using forward checking and most constrained variable reordering, what determines the depth of the search tree? What determines the branching factor at each level?

Depth - number of variables (= number of spaces or rows/columns/regions, according to representation chosen)
Branching factor - with forward checking, the number of valid values remaining for that variable -- so the size of the variable's domain after elimination. For "proper" sudoku puzzles which have a single solution, if variables are re-ordered with each consistency check, the branching factor often typically 1, but is not guaranteed to be 1; if b was always guaranteed to be 1, then search would not be necessary -- and arc consistency would always find the answer. However, since in general, arc consistency alone does not always find a solution even if there is a single, unique solution (see slide 3.2.16), b is not always guaranteed to be 1 even for puzzles which have a unique solution.