CS531 Programming Assignment 3: SuDoKu

Michael Lam, Xu Hu
EECS, Oregon State University

November 5, 2012

Abstract

In this assignment we design, implement and discuss constraint propagation and backtracking search algorithms in order to solve a specific constraint satisfication problem, SuDoKu.

1 Introduction

SuDoKu is a puzzle and constraint satisfication problem in which every unit (i.e. row, column or box) is an all-diff constraint. Each of the 81 squares can be represented as a variable on a domain of $\{1, 2, 3, ..., 9\}$. SuDoKu may be solved by backtracking search with constraint propagation.

A SuDoKu problem can be classified as easy, medium, hard or evil depending on what rules are required (and also if backtracking search is required) to solve the puzzle.

2 Algorithm

The algorithm consists of two major components: constraint propagation and backtracking search. In fact, the constraint propagation step is incorporated into the backtracking search. Some puzzles can be solved with just an application of the constraint propagation. Some puzzles require "sophisticated" rules (e.g. naked triples) to solve with just constraint propagation. Other puzzles require search and backtracking.

2.1 Constraint Propagation

The constraint propagation step applies the following rules to the SuDoKu board in this order to reduce the domain values for variables:

Rule 1 Assign to any cell a value x if it is the only value left in its domain.

Rule 2 Assign to any cell a value x if x is not in the domain of any other cell in that row, column or box.

Rule 3 In any row, column or box, find k squares that each have a domain that contains the same k numbers or a subset of those numbers. Then remove those k numbers from the domains of all other cells of that unit. (Called the "naked double" and "naked triple" rule for $k = 2$ and $k = 3$ respectively.)

Afterwards, these rules are "propagated" across the board, updating the board to be consistent with the new assignments. The process repeats until the board no longer updates (converges) or a variable is found to have an empty domain (contradiction).

In our experiments section, we will show the performance of the constraint propagation step when toggling a subset of these rules on or off.

The pseudocode for the constraint propagation step can be found below:

(ALGORITHM)

2.2 Backtracking Search

Backtracking search is a depth first search algorithm. For each step of the algorithm, it picks a variable to assign a value and then performs constraint propagation as described above. If there is a contradiction after the constraint propagation step, the search discards that assignment and backtracks, making an alternative assignment. If the constraint propagation step was successful, then the search continues making another assignment. The search continues until it finds a complete assignment that satisfies the puzzle, otherwise returns a failure.

There are several heuristics that can be implemented for the backtracking search. For our project, there are two different heuristics for choosing the variable to assign:

1. Pick the most constrained square (i.e. the variable with the least number of domain values other than those already assigned)

2. Pick any random square

In fact, the first heuristic performs better than the second heuristic in general constraint satisfication problems because choosing an assignment from the most constrained variable will help prune the search tree faster. We will demonstrate the validity of this theory in our experiments.

The pseudocode for the backtracking algorithm can be found below:

(ALGORITHM)

3 Experiments

Experiments here.

4 Discussion

Discussion here.