Lecture 2: DPLL, Sudoku and Project 1

* Propositional logic
  + Inconsistent sentence S
  + Valid sentences S
  + Satisfiable sentence S
* Task of SAT solving finding a “model”
* Clause Form
* DIMACS Format
* DPLL
* Heuristics for splitting
  + RAND
  + DLCS
  + DLIS
  + Jeroslow-Wang Heuristics
  + MOM’s Heuristic
* Sudoku
  + Problems: set of constraints on the solution
    - In PL
  + Finding a solution
    - Ask if set of constraints is satisfiable
    - Find a satisfying truth assignment
  + Solving the problem = find satisfying truth assignment
* Solve Sudoku with PL
  + Problem formulates as constraints
  + All squares must have exactly one number from 1-9
  + No number can appear twice in a row, column or square
* Ex.2
  + Problem as constraints
    - All squares much have precisely one number from 1-4
    - No number can appear twice in a row, column or square
* How to write constraints in PL?
  + Describe the world
  + One propositional variable for each value in each position on the board
  + Puzzle: 111^144^243^314^421
  + Total we need 16x4=64 variables
  + Is propositionalizing
  + Describe the constraints
    - At least one number in each square
    - At most one number in each square
  + No number appears twice in a row
  + “” in a column
  + “” in a box
  + Total: 64 variables, 4x64=272 statements
  + Write given numbers
  + All have to be put in clausal form
    - So puzzle is … ] 272
  + Now: can we find a truth-assignment to all variables which makes all these 272 statements true?
    - How do we find such a truth-assignment?
      * 2 methods
        + Truth tables

2^64 rows = 18,446,744,073,709,551,616 = 18 million trillion

Q: what would this be by for 9x9?

* + - * + Davis-Putnam

Experiment With <http://www.cs.qub.ac.uk/~I.Spence/SuDoku/sudoku.jar>

From <http://www.cs.qub.ac.uk/~I.Spence/SuDoku/SuDoku.html>

<http://www.cs.qub.ac.uk/~I.Spence/SuDoku/SuDoku.html>

Project 1: Build SAT Solver

* Read problem (= clauses) as DIMACS file
* •Implement DP + two heuristics
  + Choose 2 heuristics for Assignment
    - RAND (baseline)
    - GRAB FIRST
    - DLIS
    - RDLIS
    - 2-sided Jeroslow- Wang
* •Write output (= variable assignments) as DIMACS file
* •Apply this to Sudoku’s in Clausal Form
* •You get the Clausal encoding of Sudoku rules for free
* •You get a test set of 22.000 Sudoku’s

Your project, part II: design & run an experiment

* 1: Formulate Hypothesis
  + •Strategy X outperforms strategy Y, which outperforms basic DP
  + •Sudoku's with fewer givens are harder, with a linear correspondence between number of givens and runtime
  + •Sudoku's which are hard for people are also hard for our SAT solver
  + •Difficulty increases quadratically with the size of the Sudoku (requires encoding the rules of 2x2, 4x4 and 9x9 sudoku’s in DIMACS format)
  + Brainstorm:   
    - Performance of solver
      * Of different decision heuristics
      * Compared to the complexity of a sudoku
    - Influence of givens in Sudoku SAT-solving
      * To make sudoku unique: need at least 17 clues and not more than 77 clues
        + Looks harder to solve Sudoku with 17 clues than with 18 or more clues
      * Sudoku with fewer given elements needs more spits and backtracks than Sudokus with more given elements
        + Set up:

372 different 9x9 sudokus

With different number of clues, between 18-25

Not evenly distributed

Sudoku sorted by number of clues

For each number of clues this was calculated

Avg. of splits

Random or heuristic truth value assignments

Number of polarity switches

Backtracks

Back to an upper level

Focus

Splits

Backtracks

For basic SAT-Solver

DLIS

2-Sided Wang SAT-Solver

Average of all three

Correlation

Pearson correlation

Linear correlation expected between backtracks or number of uncertain truth assignments with number of clues

The variables were all measured on interval scale

* 2: Execute Experiment
* •Decide on a good metric (runtime, number of splits, backtracks, learned conflicts, ….)
* •Implement the metric in your SAT solver
* •Run a number of experimental conditions
* 1.Introduction (Problem description, related work)
* 2.The **design decisions** for your SAT solver
* 3.Which **approach (e.g. heuristics)**you implemented (and why, and how)
* 4.Your **hypothesis**(plus why it is interesting and plausible)
* 5.Your **experimental design** (experimental conditions, test sets, metrics, and why)
* 6.Your **experimental results** (tables, plots, graphs, statistical significance)
* **7.Findings**(interpretation of the results)(what do your results say about your hypothesis)
* 8.Conclusion