Lab3 - CSP and Satisfiablilty

Naman Goyal (2015CSB1021)

March 10, 2017

Abstract

The Lab implements Sudoku solver first as Constraint satisfaction problem add then as Boolean satisfiability problem.

1 Constraint Satisfaction Problem

Objective To compare the performance of various heuristics over simple backtracking search, by integrating the

- BS-I = Minimum Remaining Value + BS
- BS-II = Least Constraining Value + BS-I
- BS-MAC = Maintaining Arc Consistency + BS-II

1.1 Statistics

The table populates the performance statistics.

Performance Metric	**Simple BS**	**BS-I**	**BS-II**	**BS-MAC**
Average Search Time (in sec)	0.94086	0.129811	0.156545	0.730458
Minimum Search Time (in sec)	0.000299	0.001038	0.001666	0.008039
Maximum Search Time (in sec)	74.0701	4.09858	3.83638	16.9296
Total Search Time (in sec)	141.129	19.4717	23.4818	109.569
Overall Improvement in Time (in sec)	0%	86.20 %	83.36 %	22.36 %
Average Backtracks	1516685	6072	5973	5893
Minimum Backtracks	298	0	11	7
Maximum Backtracks	116038053	172610	147768	146162
Total Backtracks	227502770	910885	896061	884036
Overall Improvement in Backtracks	0 %	99.59 %	99.60 %	99.61 %
Peak Memory Utilization (in KB)	15332	15332	15336	16020
Average Memory Utilization (in KB)	7722	7722	7776	8026

1.2 Analysis

1. **Observation**

   The most significant improvement is because of **Minimum Remaining Value Heuristic** (MRV). It reduces search time from over **2 minutes to less than 20 seconds**.

   Explanation

   This is intuitive since MRV chooses the variable with the fewest legal values i.e. the most constrained variable. It picks a variable that is most likely to cause a failure thereby reducing the number of backtracks by 99.59 % and overall search time.

2. **Observation**

   Least Constraining Value (LCV) though **reduces the average number of backtracks** by 100 over MRV but **increase the search time** by 20 % over MRV alone.

   Explanation

   The reduction in backtracks is expected as LCV chooses the variable value ordering such that that rules out the fewest choices for the neighboring variables in the constraint graph. But the overhead for computing least constraining value heuristic is linear in number of constraints and domain size i.e. $O(c * d)$ hence increase the search time complexity overall.

3. **Observation**

   Maintaining Arc Consistency (MAC) further **reduces the average number of backtracks** by 80 over BS-II but **increase the search time** by a whooping 366 % over BS-II.

   Explanation

   MAC algorithm propagates the constraints and prunes the domain whenever a variable is chosen for assignment and then propagates the same like AC-3. Hence number of backtracks should be reduced as the domains of unassigned variables and thus total possibilities are reduced. But again performing arc consistency is a costly operation i.e $O(d^2)$ for each arch and it can be done at max $O(c * d)$ times i.e. overall $O(c * d^3)$.

4. **Observation**

   The **memory utilization of BS and BS-I is same** while that of **BS-II is higher** and **BS-MAC is highest**.

   Explanation

   This is expected since for MRV (BS-I) only extra computations and no extra memory is required to find the variable with minimum legal values. But for LCV an ordering of values in domain (according to conflicts with neighbouring variables), has to be calculated and stored at each node in search tree. In MAC the arc queue has to be maintained along with revised domains of variables (which may be need to be replaced with old domain values if backtrack returns failure) which leads to increase in memory utilisation.

5. **Observation**

   While doing Arc Consistency for (X_i, X_j) their is no need to call revise if there the domain of X_j contains 2 or more elements for Sudoku problem.

   Explanation

   If the domain of X_j contains atleast two elements 'a', 'b' then for 'a' in X_j, domain of X_i contains everything expect 'a'. Similarly for 'b' in X_j, domain of X_i contains everything expect 'b'; union of which gives the complete domain of X_i.

2 Boolean satisfiability problem

The problem as modeled by converting constraints to cnf in boolean variables.

Propositional variables The solution contains 81 assignments which contains any value from 1 to 9. So we have 81 * 9 propositional variables in total. We can model then as by following equation

$$RowNo * 100 + ColumnNo * 10 + AssignmentValue \tag{1}$$

e.g variable XYZ means row X (0 indexed) and column Y (0 indexed) contains assignment Z.
If a variable is positive then it is true else it is false.

Constraints

1. **Cell Constraint**

   Atleast one value from 1 to 9 in each entry.

   This is represented for each pair (i,j) from (0,0) to (8,8) that is contains atleast one value from 1 to 9. e.g for cell 1,1 it is

   111 112 113 114 115 116 117 118 119 0

   Total 81

2. **Row constraint**

   Each row contains atmost one number from {1,...,9}

   It cannot be that any 2 entries in a row contain same number i.e. atleast one of them is false.

   e.g. for row 1 number 2

   -102 -112 0

   -102 -122 0

   -102 -132 0

   -102 -142 0

   ... (36 in total)

   For each row ${}^9C_2 * 9 = 324$

   Total $324 * 9 = 2916$

3. **Column constraint**

   Each column contains atmost one number from 1,...,9

   It cannot be that any 2 entries in a column contain same number i.e. atleast one of them is false.

   e.g. for column 8 number 9

   -089 -189 0

   -089 -289 0

   -089 -389 0

   -089 -489 0

   ... (36 in total)

   For each column ${}^9C_2 * 9 = 324$

   Total $324 * 9 = 2916$

4. **Box constraint**

   Each box contains atmost one number from {1,...,9}

   It cannot be that any 2 entries in a box contain same number i.e. atleast one of them is false.

   For each box ${}^9C_2 * 9 = 324$

   Total $324 * 9 = 2916$

Total generic constraints are $2916 * 3 + 81 = 8829$

5. **Problem Specific Constraints**

   In given problem the assignment is a unary constraint.

   e.g. if it is given that cell (5,8) contains 7 then a constraint is

   587 0

After making clauses from literals input it to MiniSAT and then read the output for positive literals to get the solution.

e.g. if 645 is output by MiniSAT then (6,4) cell contains assignment 5.