Indian Institute of Information Technology

Allahabad

IAIN 532 C - Artificial Intelligence

Sudoku Solving Algorithms

Instructor:
Dr. Krishna Pratap Singh
Department of Information Technology

Submitted By :
Ankur Dengla
Gaganjeet Singh Reen
Manasi Mohandas
Prankur Agarwal
Sayantan Chatterjee

Contents

1 Candidates' declaration

We hereby declare that the work presented in this assignment report entitled **"Sudoku solving algorithms"**, submitted assignment report of course **IAIN532C (Artificial Intelligence)**[1] of Bachelor of Technology (IT) at Indian Institute of Information Technology, Allahabad, is an authenticated record of our original work carried out from Sep. 14 to Sep. 21, 2017 under the instructions of **Prof. Krishna Pratap Singh**.

Due acknowledgements has been made in the text to all other material used. The assignment was done in full compliance with the requirements and constraints of the prescribed curriculum.

Place: Allahabad
Date: 21 September, 2017

Ankur Dengla (IIT 2015 510)
Gaganjeet Singh Reen (ICM 2015 003)
Manasi Mohandas (ICM 2015 501)
Prankur Agarwal (LIT 2015 040)
Sayantan Chatterjee (IIT 2015 511)

2 Introduction to Sudoku

2.1 Problem being addressed

Sudoku is a logic-based,combinatorial number-placement puzzle. The objective is to fill a 9×9 grid with digits so that each column, each row, and each of the nine 3×3 subgrids that compose the grid (also called "boxes", "blocks", or "regions") contains all of the digits from 1 to 9. The puzzle setter provides a partially completed grid, which for a well-posed puzzle has a single solution.

Completed games are always a type of Latin square with an additional constraint on the contents of individual regions. For example, the same single integer may not appear twice in the same row, column, or any of the nine 3×3 subregions of the 9x9 playing board.

2.2 Mathematics of Sudoku

The general problem of solving Sudoku puzzles on $n^2 \times n^2$ grids of $n \times n$ blocks is known to be NP-complete. Many computer algorithms, such as backtracking and dancing links can solve most 9×9 puzzles efficiently, but combinatorial explosion occurs as n increases, creating limits to the properties of Sudokus that can be constructed, analyzed, and solved as n increases.

A Sudoku puzzle can be expressed as a graph coloring problem. The aim is to construct a 9-coloring of a particular graph, given a partial 9-coloring.

3 Theory of algorithms used

3.1 Brute-force (Exhaustive) Search

Brute-force search or exhaustive search, also known as generate and test, is a very general problem-solving technique that consists of systematically enumerating all possible candidates for the solution and checking whether each candidate satisfies the problem's statement.

While a brute-force search is simple to implement, and will always find a solution if it exists, its cost is proportional to the number of candidate solutions – which in many practical problems tends to grow very quickly as the size of the problem increases. Therefore, brute-force search is typically used when the problem size is limited, or when there are problem-specific heuristics that can be used to reduce the set of candidate solutions to a manageable size. The method is also used when the simplicity of implementation is more important than speed.

3.2 Backtracking : Constraint Satisfaction Problem (CSP)

Backtracking is a depth-first search (in contrast to a breadth-first search), because it will completely explore one branch to a possible solution before moving to another branch. Although it has been established that approximately $6.67x10^21$ final grids exist, a brute force algorithm can be a practical method to solve Sudoku puzzles.

Briefly, a program would solve a puzzle by placing the digit "1" in the first cell and checking if it is allowed to be there. If there are no violations (checking row, column, and box constraints) then the algorithm advances to the next cell, and places a "1" in that cell. When checking for violations, if it is discovered that the "1" is not allowed, the value is advanced to "2". If a cell is discovered where none of the 9 digits is allowed, then the algorithm leaves that cell blank and moves back to the previous cell. The value in that cell is then incremented by one. This is repeated until the allowed value in the last (81st) cell is discovered.

Advantages of this method are:

- A solution is guaranteed (as long as the puzzle is valid).
- Solving time is mostly unrelated to degree of difficulty.
- The algorithm (and therefore the program code) is simpler than other algorithms, especially compared to strong algorithms that ensure a solution to the most difficult puzzles.

3.3 Genetic Algorithm

Sudoku can be solved using stochastic (random-based) algorithms. An example of this method is to:

- Randomly assign numbers to the blank cells in the grid.
- Calculate the number of errors.
- "Shuffle" the inserted numbers until the number of mistakes is reduced to 0.

A solution to the puzzle is then found. Approaches for shuffling the numbers include simulated annealing, genetic algorithm and tabu search. Stochastic-based algorithms are known to be fast, though perhaps not as fast as deductive techniques.

A genetic algorithm (GA) is a metaheuristic inspired by the process of natural selection that belongs to the larger class of evolutionary algorithms (EA). Genetic algorithms are commonly used to generate high-quality solutions to optimization and search problems by relying on bio-inspired operators such as mutation, crossover and selection.

In a genetic algorithm, a population of candidate solutions (called individuals, creatures, or phenotypes) to an optimization problem is evolved toward better solutions. Each candidate solution has a set of properties (its chromosomes or genotype) which can be mutated and altered; traditionally, solutions are represented in binary as strings of 0s and 1s, but other encodings are also possible.

The evolution usually starts from a population of randomly generated individuals, and is an iterative process, with the population in each iteration called a generation. In each generation, the fitness of every individual in the population is evaluated; the fitness is usually the value of the objective function in the optimization problem being solved. The more fit individuals are stochastically selected from the current population, and each individual's genome is modified (recombined and possibly randomly mutated) to form a new generation. The new generation of candidate solutions is then used in the next iteration of the algorithm. Commonly, the algorithm terminates when either a maximum number of generations has been produced, or a satisfactory fitness level has been reached for the population.

A typical genetic algorithm requires:

- a genetic representation of the solution domain,
- a fitness function to evaluate the solution domain.

Once the genetic representation and the fitness function are defined, a GA proceeds to initialize a population of solutions and then to improve it through repetitive application of the mutation, crossover, inversion and selection operators.

4 Analysis and statistics

Test Case 1:

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

Solving time:

- Brute-force: 0.014s
- Backtracking: 0.124s
- Stochastic (GA): 145 generations (5min)

Stochastic parameters:

- Population limit: 10000
- Probability of crossover: 0.7
- Probability of mutation: 0.1:

Test Case 2:

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

Solving time:

- Brute-force: 0.209s
- Backtracking: 1.595s
- Stochastic (GA):3989 generations(7.5hr)

Stochastic parameters:

- Population limit: 10000
- Probability of crossover: 0.7
- Probability of mutation: 0.1:

Test Case 3:

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

Solving time:

- Brute-force: 0.210s
- Backtracking: 1.548s
- Stochastic (GA):17373 generations(2.5 days)

Stochastic parameters:

- Population limit: 10000
- Probability of crossover: 0.7
- Probability of mutation: 0.1:

References

[1] Stuart J. Russell Peter Norvig. Artificial intelligence: A modern approach. Prentice Hall Publishers.