Sudoku Problem

names

1 Introduction

Sudoku is a popular logic-based combinatorial puzzle game. We are considering A=a Sudoku puzzle consisting of 625 cells, contained in a 25x25 grid. Each cell can contain a single integer ranging between 0 and 24. The grid is further split up into 25 5x5 sub-grids. The purpose of Sudoku is to fill up the entire 25x25 grid such that the following constraints are met:

1. Each row of cells is only allowed to contain the integers zero through to twenty-four exactly once.
2. Each column of cells is only allowed to contain the integers zero through to twenty-four exactly once.
3. Each 5x5 sub-grid is also only allowed to contain the integers zero through to twenty-four exactly once.

A number of cells in the grid are pre-defined by the puzzle setter, resulting in the Sudoku puzzle having a single, unique solution. These puzzles are set at different difficulties.

// Can have a picture of an easy sudoku grid

Various algorithms have been implemented to solve the Sudoku problem. For some Sudoku puzzles, a logic-based algorithm, mimicking the way a human would solve the puzzle, is adequate to attain a solution. Harder puzzles, where guessing is required, can be solved using backtracking algorithms. The problem with backtracking is that the efficiency of the algorithm is dependent on the number of guesses required to solve the puzzle. Hence, harder puzzles will take longer to solve. A solution to this problem could lie in using stochastic optimization techniques.

Some work has been done on solving Sudoku using stochastic optimization techniques. The primary motivation behind using these techniques is that difficult puzzles can be solved as efficiently as simple puzzles. This is due to the fact that the solution space is searched stochastically until a suitable solution is found. Hence, the puzzle does not have to be logically solvable or easy for a solution to be reached efficiently.

Furthermore, stochastic optimization techniques are used to find the global optimum of a problem which contains many local optima. Due to the constraint nature of Sudoku, it is very likely to find a solution which satisfies some of the constraints but not all of them, hence finding a local optimum. Due to the stochastic nature of these techniques, the solution space is still searched even though a local optimum has been found, allowing for the global optimum to be detected.

This project explores the implementation of three different algorithms – namely Backtracking Search and two stochastic optimization techniques as applied to the Sudoku problem: Genetic Algorithm, Simulated Annealing. Each Technique is discussed, implemented and tested on the puzzle. Results are compared and critically evaluated.

Both the stochastic techniques share some common features. Each technique requires an initialization process, where the solution space is defined, as well as a fitness function, which ensures that the objectives and the constraints are being adhered to.

2 Simulated Annealing Algorithm

Simulated Annealing (SA) is an optimization technique, inspired by the annealing process used to strengthen glass or crystals: a crystal or glass is heated until it liquefies and then is cooled slowly, allowing for the molecules to settle into lower energy states.

Unlike GA, SA is not population based but rather alters and tracks the state of an individual, continuously evaluating its energy by using an energy function. SA finds the optimal point by running a series of Markov Chains under different thermodynamic conditions: a neighbouring state is determined by randomly changing the current state of the individual by implementing a neighbourhood function. If a state with a lower energy is found then the individual moves to that state. Otherwise, if the neighbouring state has a higher energy then the individual will move to that state only if an acceptance probability condition is met. If it is not met, then the individual remains at the current state.

The acceptance probability is a function of the difference in energies between the current and neighbouring states as well as the temperature. The temperature is initially made high, making the individual more susceptible to moving to the higher energy state. This allows the individual to explore a greater portion of the search space, preventing it from being trapped in a local optimum. As the algorithm progresses the temperature is reduced, in accordance with a cooling schedule, causing the individual to converge towards the state with the lowest energy and hence the optimal point. A typical SA algorithm works as follows:

1. Initialize an individual's state and energy.
2. Initialize temperature.
3. Loop until temperature is at minimum.
   (a) Loop until maximum number of iterations has been reached.
      i. Determine neighbouring state by implementing the neighbourhood function.
      ii. Determine the energy of the current and neighbouring state.
      iii. If the neighbouring state has a lower energy than the current, then change the current state to the neighbouring state.
      iv. Else, if the acceptance probability is fulfilled then move to the neighbouring state.
      v. Else stick to the current state.
      vi. Keep track of state with lowest energy.
      vii. End inner loop.
      viii. Alter temperature in accordance with the cooling schedule.
   - End Outer Loop

2.1 Implementation Details

Initialization of the Sudoku puzzle is done in a way where each grid contains the numbers 0 to 24 exactly once. Furthermore, any operation carried out on an individual must ensure that this constraint is not violated. This approach ensures that it is computationally less

demanding. The neighbourhood function randomly picks two in a grid and interchanges their values.

The fitness function implemented here involves determining whether an integer is repeated or is not present in a particular row or column (since the constraint of in a grid is always satisfied, only repetitions in rows and columns are considered). A fitness value is assigned to a possible solution based on the number of repeated or non-present integers. The more repeated or non-present integers there are in a solution's rows and columns, the higher the fitness value assigned to that solution. A geometric cooling schedule is chosen which multiplies the previous temperature by a fraction.

3 Conclusion

Whatever we analysed