Using Evolutionary Algorithms to Solve Sudoku Problems

640034510

October 27, 2017

1 Introduction

The Sudoku puzzle is a combinatorial number placement where the player must fill a partially completed nine by nice grid with the numbers one to nine. These numbers must be placed in a way such that each row, column, and three by three sub grid contain a unique set. Due to this constraint on the placement of numbers, Sudoku proves to be a NP-complete problem which makes it impracticable to solve using a direct method. Instead a heuristic search method must be applied to the problem, of which the genetic algorithm metaheuristic is highlighted in this report. Inspired by the biological process of Meiosis, genetic algorithms encode potential solutions as population of chromosomes which undergo a series of genetic operators to produce offspring. Over successive generations of offspring, the average fitness of the population should increase until the final solution is found.

2 Design

2.1 Solution Space and Representation

The solution space for an empty nine by nine Sudoku grid is vast, with an approximate $6.67 * 10^{21}$ possible grids. As the number of clues given in a Sudoku puzzle increases then the possible solution space decreases but still remains large. To further reduce the size of the solution space, it was decided that all solutions must have valid rows. That is, all rows of an individual must contain some permutation of the set of numbers one to nine. All subsequent operators on a population must therefore conserve this rule to ensure that the search space remains reduced and errant solutions do not enter the population.

Solutions are represented as a one dimensional list of integers, the length of which is set to the number of spaces in the Sudoku grid. This representation was chosen as it makes representing rows, columns, and grids within the solution trivial. Additionally, this representation allows for genetic operators, described below, to be applied to solution chromosomes in an analogous way to that of their biological counterparts.

2.2 Fitness Function

The success state of a Sudoku puzzle when the contents of every row, column, and box are each a permutation of the numbers of the numbers one to nine. Therefore the fitness function first calculates the sum of the number of unique digits in each column and box. This number is then normalized in the range of zero and one by dividing it by the maximum number of unique digits in all rows and columns. In the case of a nine by nine Sudoku grid, the maximum number of unique digits in the rows and columns is 162. Rows have been excluded from this metric as all solutions already have valid rows which present the maximum score. By only including rows and columns in the metric, any positive change to the fitness of a row or column will have a greater impact on the overall fitness score of the solution.

2.3 Population Initialisation

The population is initialised by copying the two dimensional grid into a one dimensional chromosome, preserving the pre set numbers and setting zero for grids which are blank. For each row in the grid, the set of the existing numbers is calculated and the blanks are filled from the inverse of this set, creating rows which are a permutation of the numbers one to nine.

2.4 Selection and Replacement

To select parents to produce offspring, binary tournament selection is used. This operator picks two solutions from the population and compares their fitness functions. The solution with the greatest fitness function is chosen to be a parent used to produce the subsequent generation. A binary tournament selector was used as it reduces selection pressure which is important because of the amount of local maximum present within a Sudoku problem. By allowing less optimal solutions to affect the subsequent generation, a wider diversity of solutions will be present which may be able to overcome a local maximum.

Talk about generationalism and possible elitism?

2.5 Crossover Function

The crossover function chosen is an n-point crossover function which preserves the ... Talk about pairs of parents

2.6 Mutation Function

The mutation function chosen is a singe swap mutator which ensures that the resultant mutant retains rows with valid permutations. To do this, the mutator first picks a row within the grid and then two genes within. If the two selected indexes are distinct then the genes contained within are swapped. In this way, the diversity of possible candidate solutions is increased and may help to overcome a local minima while preserving permutations. The chance that a gene will be mutated is set to 0.5 such that half of all chromosomes within a population will undergo the mutation. This was chosen to ensure diversity while still maintaining a proportion of strong solutions to increase selection pressure.

2.7 Termination Criteria

There are three termination criteria which halt the operation of the genetic algorithm. The first is when an optimal solution, the fitness function equals one, has been found as simply no more generations need to be produced. The second is when a thousand generations have been reached which allows time for small population sizes to reach a solution while not being overly long. This number is reduced to 200 in the case of a population size of 10000 due to the expectation that an optimal solution should be reached before. Finally, if there has been not change to the highest fitness function in a population for 200 generations then the runtime is terminated. The shows that a local maximum has been reached and the population has subsequently stagnated.

3 Experiments

A number of experiments were designed and run to test the affect that various genetic operators have on the progress of a population towards a solution. These genetic operators were tested at a range of population sizes and on increasingly harder Sudoku problems. The first experiment run was on a genetic algorithm as described in the design section but without any

4 Questions