Genetic Algorithm for solving Sudokus

Here is a quick overview how we tried to solve this problem.

* **Representation:**

Our Sudokus are represented by 2-dimensional arrays with the dimension 9 x 9.

* **Initialization:**

After the already known fixed numbers are added to the initial Sudoku, we calculate additional fixed numbers. In order to do so, we check for every cell if there exists a number which can only be placed in this particular cell based on the already known numbers in its row, column and 3x3 subgrid. If so, we add that number to the initial Sudoku.

When this process finds no more additional initial values, all the 3x3 subgrids are filled with values, such that every subgrid if feasible (Contains every value from 1 to 9 once). This process is repeated x times, in order to get an initial population of size x.

* **Evaluation:**

In order to evaluate the fitness, we count the existing contradictions of a Sudoku (Multiple occurrences of numbers in rows or columns are considered a contradiction). Then we also calculate the average deviation value, by adding each number of a row or column and checking the deviation to the estimated value of 45 (Which is the sum of the numbers from 1 to 9). Finally the fitness values is calculated by considering both the number of contradictions and the average deviation value.

* **Recombination:**

The recombination in our solution is done by slices between the 3x3 subgrids. In a k-Tournament the 2 fittest out of k possible parents are selected, which are then recombined randomly via either a single slice (Slice after a random number of 3x3 subgrids) or multiple slices (50% chance of a slice after every 3x3 subgrid). With this kind of recombination we guarantee that our 3x3 subgrids stay feasible.

* **Mutation:**

Our mutation is basically performed via a permutation inside of a 3x3 subgrid. We randomly select between three approaches: Either two random cells are switched, or all the values between two cells are reversed, or we do both. This process happens with a probability or 1/9, since we have 9 3x3 subgrids in a Sudoku. Because we only use permutations we guarantee that our 3x3 subgrids stay feasible.

* **Selection:**

In our selection process we consider all parents and newly added children of our population. We sort them based on their fitness and then select x out of them, with x being our population size. The fitter a Sudoku is, the more likely it is to get selected by the algorithm (Although no Sudoku, except the fittest one, can be sure to get selected).

Additional Info:

* We have implemented two strategies in our algorithm, the so called solveSingle and the solveMultiple. In the solveSingle mode we run the algorithm normally after generating our initial population. But in the solveMultiple mode we generate a few initial populations in order to run the algorithm on all of them and get the fittest solutions of all of them. These fittest solutions are then combined to a new initial population and the algorithm is performed one more time with this population.
* To run the project please import it into Eclipse (Import existing project into workspace) and run the SolverExec.java file.