Genetic Algorithms

Base Version

In this assignment a genetic algorithm was implemented to solve the 8 queens problem based on the information provided in the lecture. Genetic algorithms work by implementing ideas from biology like genetic mutation and recombination as well as natural selection to explore the search space more effectively than random search. This can be beneficial for problems such as the 8 queens problem, where exploring all potential problem states becomes computationally unfeasible, and on the first view, no clear problem gradient can be found to approximate solutions to the problem with more efficient search techniques.

To solve the problem using a genetic algorithm, first random chess board configurations were created and saved as the population (see Figure 1). Then natural selection was performed on the population by looking at a subset of chess configurations (size=4) and picking the one configuration with the highest fitness value. This was repeated, until the new population was the same size as the initial population. The fitness value of a chess board constellation was calculated by determining the total number of queen attacks in a method called fitness() as follows:

$$fitness(state) = 28 - count\_queen\_attacks$$

A state with fitness 28 was found to be a goal state, because no queen attacks were registered. After the fittest individuals were picked out of the population, recombination was performed by picking two individuals and cross-paring their states at a random cross-over point. This created new variants, which can also be referred to as children. Lastly mutation was performed on the children to increase the variability and scan through a bigger search space by randomly repositioning one queen in the individual state.

The fittest individual out of a subset was chosen for the creation of the new population instead of a probabilistic average of many states. This was done to decrease the computational complexity of the algorithm, as the fitness function was determined to be the bottleneck in regard to time complexity. Similarly to that argumentation, two parents instead of multiple parents were chosen for the crossover, to reduce the time complexity of this computational step. The base version of the algorithm is able to find a solution to the 8 queens problem in less than 30 seconds. In the fastest recorded attempt a solution was found after only 2 seconds. On average, the base version takes 8.59 ± 3.92 seconds to find a solution (5 runs conducted).

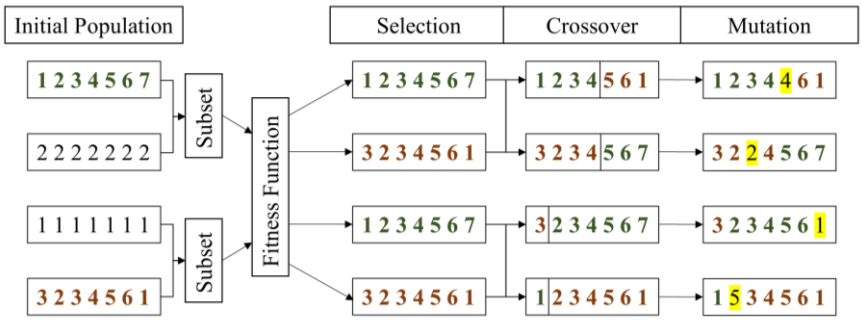

***Figure 1**: Illustration of the genetic algorithm implemented to solve the 8 queens puzzle. Initial population is split into subsets (right) before the fitness function is applied to choose the fittest individuals to build a new population in the selection step. New variants are then introduced into this population in the crossover and mutation steps. The optimal subset size was determined to be 4. The optimal crossover and mutation rates were determined to be 1 and 0.2 respectively.*

Extension

The base version algorithm discussed above was extended in multiple ways to achieve a significantly better performance. First, the code was modified to be able to solve puzzles of any size n other than 8. Additionally, multiple crossover and mutation passes were added to one algorithm iteration to increase the child population variability and potentially boost the performance.

On top of that, the initialized states where chosen not in a random fashion anymore but modified so adjacent queens would not be right next to each other. This caused the starting states to already have significantly higher average fitness compared to random states and led to the algorithm converging in a solution faster.

Finally, ideas from the simulated annealing algorithm discussed in the lecture were used to start the search with high mutation and crossover rates, which where then reduced strongly just like the temperature variable cooled down in simulated annealing. It was assumed that this would help the genetic algorithm explore a higher search space by creating more variation in the populations early on in the search but help reduce variability by the end of the search to converge in a solution. The variation in mutation and crossover rates was achieved by multiplying both rates with the temperature, which was calculated as:

$$temp = \frac{d}{\log(iteration_{count})}, temp \in [0,1]$$

In this case, d is a parameter that can be tuned to control how fast the cooling will proceed.[1] The higher the value of d, the slower will the temperature cools down.[1] A value of d=2 yielded the fastest results for the 8 queens problem.

The extended version finds solutions for the 8 queens problem significantly faster than the base version (1.49 ± 0.98 seconds) but lacks the ability to find many different types of solutions in a fast matter. Attempts to find all solutions for any board constellation failed. It is assumed that the non-random starting setting is responsible for this outcome. The extended version can find solutions for other boards $n < 8$ in less than 30 seconds. Further experiments could be conducted to find out, what type of starting states result in the most effective search through the complete search space.

References

[1] Bertsimas, D., Tsitsiklis, J., 1993. Simulated Annealing. *Statistical Science*, 8(1), 10-15.