Maryam Ibrahim Mahmoud
March 2024

Applying the Genetic Algorithms to the 8 Queens problem

Introduction

The 8-queens problem involves placing eight queens on an 8×8 chessboard in such a way that no two queens threaten each other. This problem is challenging due to its large search space and possibilities. Genetic algorithms offer an effective approach to solving the 8-queens problem by exploring the solution space and converging towards optimal solutions.

The implementation

The 8 Queens problem is solved by the combination of the genetic algorithm's functions (presented in Report 1 under the title of Genetic Algorithms For Optimization). It is solved by representing candidate solutions (initial population) that are represented as a combination of integers (represents the placement of each queen), a fitness function that acts as the guideline for the operations, and all other functions as selection, crossover, and mutation that will at the end lead to the final optimal solution.

The following code will illustrate how the combination is performed:

```
1 usage
def queens8(pop_size, max_gen, pc=7, pm=0.01):
    population= init_pop(pop_size)
    best_fitness_overall= None
    for i_gen in range(max_gen):
        fitness_vals=calc_fitness(population)
        best_i=fitness_vals.argmax()
        best_finess=fitness_vals[best_i]
        if best_fitness_overall is None or best_finess > best_fitness_overall:
            best_fitness_overall=best_finess
            best_solution=population[best_i]
        print(f'\ri_gen ={i_gen:06} -f={-best_fitness_overall:03}',end=' ')
        if best_finess==0:
            print('\nfound optimal solution')
            break
        selected_pop= selection(population, fitness_vals)
        population=crossover_mutation(selected_pop,pc,pm)
    print()
    print(best_solution)
```

Maryam Ibrahim Mahmoud
March 2024

This function takes four arguments: 1. population size, 2. Maximum number of generations (that determine how iterations will be performed), 3. probability of cross-over, and 3. Probability of mutation.

First, it initializes the population using the initial population function. Then, it initializes the best fitness found to keep track of the value of the best fitness found overall. Second, it iterates through a loop in the max. number of generations times and in this loop all the other functions of the genetic algorithms will be placed to be performed on the whole population.

The operation operates as follows:

1. Calculate the fitness values of all individuals in the population using the calc_fitness function.
2. Find the index (best_i) of the individual with the highest fitness value using argmax() method.
3. Update best_fitness_overall and best_solution if the current best fitness value is better than the overall best fitness value found so far.
4. Print information about the current generation number *number of trials (i_gen) and the best fitness value found so far (-best_fitness_overall).
   The - is used to indicate fitness, where lower values are better.
5. If the best fitness value is 0 (indicating that an optimal solution has been found), print a message indicating that an optimal solution has been found and break out of the loop.
6. Select a new population (selected_pop) for the next generation using the selection function, based on the fitness values of individuals in the current population.
7. Create a new population for the next generation by applying crossover and mutation operations (crossover_mutation) to the selected population.

After completing the whole process, whether you have reached the maximum number of generations or found the optimal solution) you'll:

8. Print a newline character to move to the next line after printing the progress information.
9. Print the best solution found (best_solution), which represents the arrangement of queens on the chessboard that according to the best fitness value found during the genetic algorithm execution.

Maryam Ibrahim Mahmoud
March 2024

The 8 queens function will be tested as follows:

```
functions.queens8 (pop_size=100, max_gen=10000, pc=0.7, pm=0.01)
```

Where you'll input the values of the population size, max generations, probability of crossover, and probability of mutation as preferred. You won't need a print statement because it is already included in the function.

Note that the output will be different each time you run the code. Sometimes, it'll be found after a small number of iterations, other times it will take a large number of iterations or even won't be able to find the optimal solution. This is mainly because the Genetic Algorithm is mainly based on probabilities and random selections.

Examples of the possible outputs:

```
 [1 7 3 0 0 2 4 2]
 [1 6 3 0 0 4 4 7]
 [1 7 0 3 0 6 5 5]]
i_gen =007187 -f=000
found optimal solution

[2 5 1 6 0 3 7 4]
```

```
 [3 2 5 2 0 2 6 0]
 [3 2 5 6 2 5 6 3]
 [6 7 4 2 0 7 5 0]]
i_gen =000040 -f=000
found optimal solution

[1 6 4 7 0 3 5 2]
```

```
  [2 0 7 1 7 5 4 3]
  [2 0 7 1 7 5 7 3]
  [6 3 4 1 3 0 4 3]]
 i_gen =009999 -f=002
 [2 7 4 1 5 0 6 3]
```

when an optimal solution isn't found.

conclusion

In summary, the queens8 function combines the execution of a genetic algorithm to solve the 8-queens problem, printing progress information and the best solution found during the process.