AI DOCUMENTATION

|  |  |  |  |
| --- | --- | --- | --- |
| القسم: | السنة: | :ID | الاسم: |
| CS | 3 | 201900708 | محمد علاء الدين محمد رشدي |
| CS | 3 | 201900678 | محمد سعيد محمد عبدالرحمن |
| CS | 3 | 201900583 | كريم محمد حسني حمادي |
| CS | 3 | 201900713 | محمد عماد حنفي عبدالفتاح |
| CS | 3 | 201900701 | محمد عبدالفتاح شعبان عبدالفتاح |
|  |  |  |  |

shorturl.at/bsBDV

***Project idea in details:***

* + 1. The Main idea of N-Queen is Putting N number of Queens on a N\*N Board in away where no 2 Queens meet in the same column ,row or diagonal
  1. ***Similar applications*** :
     1. Go , sudoku , Chess , NPuzzle

***Approaches we used:***

* ***Differential Evolution approach:***

it is a special case of Genetic Algorithms in which it focus on converging every generation of solutions (Parents) into a specific target and the more generations we go and the more population we have , will help the solutions gets to the target faster and more accurately helping us to find the peak of our Minima .Differential Evolution was occupied in our NQueens project by creating NP Vectors of Dimension 1\*N , All elements in each vector is distinct with values 0<N , each value in the vector represents the row where the Queen is currently at , while the index represents the column where the Queen is currently at.

After creating the random population , the whole population is inserted into the Fitness function which measures how many Queens are facing each other diagonally and then we sort the population according to the Fitness Value .

Then the process of converging the populations into the target starts by doing CrossOver , Mutation ,Selection over each parent in our population.it start looping over each parent(P4) in the population and then picking 3 different random parents (P1,P2,P3) and then creates the Mutant Child based on the following equation

Mutant child = P1+F\*(p3-p2) where F is a random weight (Mutation process) and then start creating Trial child by mixing both the P4 and mutant child according to a random rate CO (Crossover process) and then select between P4 and Trial child whoever has the lower fitness that help the population converge to the Minima (Selection process ) and do the same operation for specific no. of generations or there are no more better solutions (Converged).

* ***BackTracking Approach :***

It is a recursive approach which helps to find all the solutions possible on the board by trying all the possible combinations . By putting the first Queen on the first row ,first column and start processing from this position by accessing the next row and try to find a valid column then accessing the next row and try to find a valid column so on … then the moment

an answer is found , it is added to a set of unique solutions ,then it returns to try the rest of columns then backtrack to try more and more combinations of rows and columns.

***Development platform:***

Language : Python

Libraries : 1)Queue

2)Threading

3)Time

4)random

5)Randint

6)Numpy

7)pygame

8)pygame\_Gui

9)matplotlib.pylpot

*Proposed Solution/Main Features:*

Main Functionality of the program is to find a solution for the N-Queen Problem in which you have to put N number of queens on a N\*N chessboard where no 2 queens can check each other.

The Solution is found through 2 Algorithms, Differential Evolution and Backtracking.

The program finds the solution and then introduces a few features to the user to choose from, either to switch back and forth between the different solutions of both algorithms, or to Compare between the initial population and the last generation generated by the Differential evolution algorithm to see how efficient it was, lastly he can chooses to compare between both algorithms ( Differential Algorithm and Backtracking ).

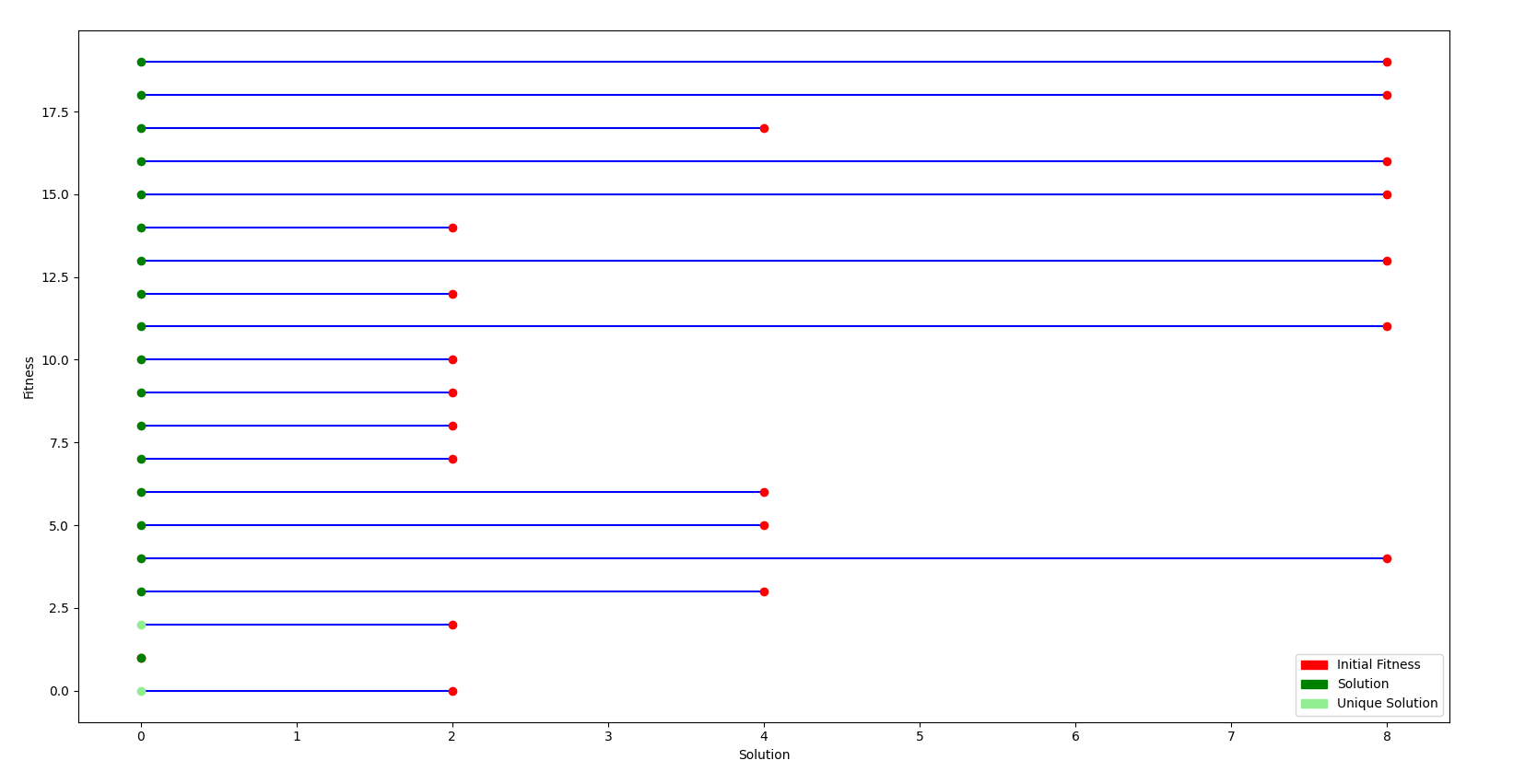
***Experiments & Results :***

1. 4 Queen Problem ( 300 Generation, 20 Population ).

* Solution

Graphical user interface, chart, treemap chart

Description automatically generated

* Generations Comparison

1. 5 Queen Problem ( 300 Generation, 40 Population ).

* Solution

Chart, treemap chart

Description automatically generated

* Generation Comparison

Chart

Description automatically generated with low confidence

1. 8 Queen Problem ( 300 Generation, 40 Population ).

* Chart, treemap chart

  Description automatically generatedSolution
* Generation Comparison

A screenshot of a computer

Description automatically generated with low confidence

**Finally,** this is the Comparison plot of the 3 examples we just demonstrated, The comparison is between the 2 algorithms

( Purple ->Differential , Black -> Backtracking )

With time on the y-axis and number of solutions on the x-axis.

Chart, line chart

Description automatically generated

***Flowcharts And Diagrams:***

General Diagram:

Graphical user interface

Description automatically generated with medium confidence

Diagram

Description automatically generated

***Graphical user interface

Description automatically generated with medium confidence***

***Graphical user interface

Description automatically generated with medium confidence***

***Analysis:***

The backtracking algorithm is clearly superior in every possible way ( time and number of solutions ).

However on high values of N the differential algorithm may get better time complexity than the backtracking, since the backtracking tries to find every possible solution.

That being said the differential plot may be a little confusing because it shows only the unique solution but actually it get probably as many solutions as the backtracking but most of them aren’t unique.

* ***Advantages and disadvantages.***

|  |  |  |
| --- | --- | --- |
|  | Differential | Backtracking |
| Speed | Slower on small values, but faster on high values | Fast on Small values of n |
| Number of Solutions | Doesn’t Guarantee All solutions | Guarantee All Solutions |
| Order of Solutions | Not ordered | Ordered |

* ***Future Work.***

In the future we will try to improve the differential evolution by eliminating Solutions that are repeated and make the algorithm only generate unique solutions.

We can improve backtracking by using heuristics that improve how we search and try to make the search space as small as possible.