Slovak University of Technology
Faculty of informatics and information Technology
Ilkovičova 2, 842 16 Bratislava 4

Artificial Intelligence

Assignment 3
Zen Puzzle Garden
Zenova zahrada

Sultan Numyalai

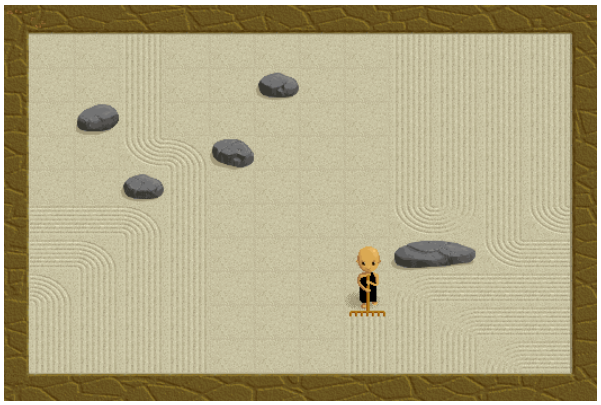
Instructor: Ing. Ivan Kapustík
Study Field: INFO3
Year: 2. Bc
Academic Year: 2017/2018

Semesster:Summer

Contents

1 Assignment

Zenová záhradka je plocha vysypaná hrubším pieskom (drobnými kamienkami). Obsahuje však aj nepohyblivé väčšie objekty, ako napríklad kamene, sochy, konštrukcie, samorasty. Mních má upraviť piesok v záhradke pomocou hrablí tak, že vzniknú pásy ako na nasledujúcom obrázku.



0	0	1	0	0	0	0	0	10	10	8	9
0	0	1	0	0	K	0	0	10	10	8	9
0	K	1	0	0	0	0	0	10	10	8	9
0	0	1	1	K	0	0	0	10	10	8	9
0	0	K	1	0	0	0	0	10	10	8	9
2	2	2	1	0	0	0	0	10	10	8	9
3	3	2	1	0	0	0	0	K	K	8	8
4	3	2	1	0	0	0	0	5	5	5	5
4	3	2	1	0	0	0	11	5	6	6	6
4	3	2	1	0	0	0	11	5	6	7	7

Abstract.
The problem is to Zen Puzzle garden as represented in the above pictures visually and logically , The dimension of the basic board is 10X12. The map is represented by an int 2D array in which numbers greater than 0 represents raking of the garden and 0 represents sand and -1 represents the rocks in the garden through which the alien can't come through. The Aim of the assignment is to solve the problem by an Evolutionary algorithm.

2 Solution:

Data Structures and it's functions

Controller – Contains main function of the program and after compiling the class, the program reads map from the file and makes logical representation in the Garden class and then through the class EvoluteSearch finds it's solution

Coordinate – represents and provides the raking in the garden, contains Rows, columns previous coordinates of the predecessor. And the direction in which the alien is raking

EvoluteSearch – This is the most important and base class of the program which solves the problem of Zen's Garden and contains the method genetic(), Containing attributes:

Garden, Num of genes, Population ,TOURNAMENT_SELECT,POP_SIZE = 100;
GENERATIONS = 10000;
MIN_MUT_RATE = 0.05f;
MAX_MUT_RATE = 0.80f;

```
      CROSS_RATE = 0.90f
Functions used in the program:
```
generateChromozome() – returns array of genes which contains random numbers from 1 to size of perimeter of the map and has 50% Probability of being negative, and then we create individuals from the random genes.

doTournament(int[]) – generates 2 or 3 according to it's argument from the random numbers which are from a to Population size which are indexes for fitness array, The function which calls this function finds out the best fit pair from the individuals

Garden – This Class makes the data structure for the Zen's Garden Which reads the file and makes 2d array of the garden. Sets number of rows, colums , Rocks, parameter of the rectangle and required number of raking.

Functions

rakeGarden(int[], boolean) – This function is used for raking the sands in the map, in which first makes a copy of the map and then use the new map for raking. This function get's parameter array of the chromosomes and then for each gene it specifies starting position for the alien, and gets it's direction from another function getDirection.

Before starting raking the sand, the function controls whether in the starting position is free and doesn't contain a rock or raked sand.

Then the function controls that the alien shall not be at the end of the garden by the function inMapBounds.

And then controls that there is no rock in the actual position which is represented in the map by a -1 or a number greater than 0.

Which means that the sand has been already raked.

If true, the we call the function changeDirection, which get's as a parameter actual coordinates of the chromosome, and changes the direction. And if the function doesn't find such a direction then clears the whole path and assign value 0 to the path of the previous coordinate.

Then the Alien enters the garden with the raking number and starts raking the garden along the direction. At the end of each succefull raking the function increases the counter and starts another gene.

After raking the whole garden and the sands, the function is called with print Boolean as true and call the function printMap and writes the solution on the console.

getDirection(int) – this function get's an int parameter which is the nuber of the actual gene finds out the direction and starting position across the garden on which the alien can start raking.

R = 10 (#rows) dirR (direction row)
C = 12 (#columns) dirC (direction column)
HP = 22 (half perimeter)
CH = (44,44) (gene number)

Starting Coordinate
Ending coordinate

Direction
CH(10) <= S(12)
R=0 C=CH-1 dirC=1 dirR=0 Right
0 9 1 0
CH(15) <= PO(22)
R=CH-S-1 C=S-1 dirC=0 dirR=-1 Up
2 1 1 0 -1
CH(25) <= PO+R (32)
R=CH-HP-1 C=0 dirC=0 dirR=1 Down
2 0 0 1
CH > PO+R(32)
R=R-1 C=CH-HP-R dirC=-1 dirR=0 Left
9 3 -1 0

changeDirection(Coordinate, int, int[][]) – this function chages the direction of raking in case of collision, has the coordinates of the chromosome which needs to be changed, number of gene, 2D Map.
First creates a new coordinate and un-numbered and finds out whether is the direction vertical or horizontal. According to the direction makes another coordinate but with different direction, through which the alien can come, which means should be un-raked sand. If there are two such ways through which the alien can come, then the alien selects the path according to the number in each coordinate, if the doesn't a direction through which the alien can come, the function returns null.

Following is output for one of the main tests,

```
Generation:   114. Max raked:  114   Mutation rate: 0.41


 10  10   5   8   8   4   4   2  11  11   7   9
 10  10   5   8   8  -1   4   2  11  11   7   9
  5  -1   5   5   5   5   4   2  11  11   7   9
  5   5   5   5  -1   5   4   2  11  11   7   9
  5   5  -1   5   5   5   4   2  11  11   7   9
  4   4   4   4   4   4   4   2  11  11   7   9
  2   2   2   2   2   2   2   2  -1  -1   7   7
  3   3   3   3   3   3   3   3   3   3   3   3
  1   1   1   1   1   1   1   1   1   1   1   1

  6   6   6   6   6   6   6   6   6   6   6   6
```

3 Algorithm

Algorithm starts by initializing the chromosomes into a 2D array of populations by the function generateChromosome(), then the function runs for the selected Generations to find out the solution.

1. For each individual the program calls the function rakeGarden, which through the chromosome of the actual individual rakes the garden and finds out how large area or sand was able to rake to the fitness array.

2. In the fitness array we look for the most successful individual or so called individual with the highest rakeAmount in each generation and we write it's value to max and it's index to Imax.

3. If max is equal to the required number of raking in the garden which was previously calculated, then we have the solution, and then we call the function rakGarden with parameter with the population which is in the index iMax and print Boolean true, and print's the solution. And print's out the sequence of the genes of the chromosomes as well.

4. If the program doesn't find the solution in the current generation and the value of muteRateAct is not greater the maxRate, then we increase the mutation rate about 0,01, if it's greater than max then we set it to Min-mutation rate. Which affects creating the generations.

5. We deculare individuals which represents the index of the best fit chromosomes from 2 or 3 random individuals and sending them to the function doTournament and then we copy them to the 2D array called children.

6. then we do the cross-over with 90% probability.

For each of the genes from those selecte 2 indexes in the array children we do mutation with the mutation rate 0.02 to 0.3

7. generating random mutation numbers from 1 to size of map which has probability of 0.5 to be negative, then we find this number in all the genes of the chromosomes.

8.if we find the number we assigned above then we change the number of the gene with the actual number , if doesn't find the number then we ex-change the number of the actual gene with mutNum.

9. after creating all the successors we write the number of the best individual for the next generation in the first index of the children array. Then we write the children array in the previous population array.

4 Conclusion

The Program is written in the programming language Java, including 4 classes for realization of the program . and is tested for different input or gardens.

Tournment_selection is implemented in two different ways which in the first one selects from two different individuals and in the second one from 3 different. Which in the second way finds out better than the first.

Generations, finding the best solution and raking amount is directly proportional to size of generations.

Population size, is indirectly proportional to that of making generation.

Cross-rate, to find out optimal solution the program would have 1 cross-rate.

5 References

1. *Kinnear, K. E.*: Advances in Genetic Programming. Cambridge: MIT Press, 1994.
2. *Ackermann, J.*: Robuste Regelung. Berlin, Heidelberg, New York: Springer-Verlag, 1993.
3. MIT Lecture on evolutionary algorithms