ISBN 978-93-5156-328-0
International Conference of Advance Research and Innovation (ICARI-2014)

Maze using Hybrid Genetic Algorithm

Bhawana Singh*, Anil Kumar Pandey
Department Of Computer Science and Engineering, Invertis University, Bareilly, Uttar Pradesh, India

Article Info

Article history:
Received 29 December 2013
Received in revised form
10 January 2014
Accepted 20 January 2014
Available online 1 February 2014

Keywords
Hybrid Genetic Algorithm
Disjoint-set data structure
Maze structure
Genetic Algorithm

Abstract

Genetic concept does not measure well with complexity. That is, where the number of elements which are exposed to mutation is large there is often an exponential increase in search space size. This makes it extremely difficult to use the technique on problems such as designing an engine, a house or plane. In order to make such problems tractable to evolutionary search, they must be broken down into the simplest representation possible. And another problem of complexity is the issue of how to protect parts that have evolved to represent good solutions. In this paper describes the approach of solving Maze problem with Hybrid Genetic Algorithm. A method for generating the maze structure has also been described.

1. Introduction

In real-world, a sequence of decisions is required to solve a problem, in which each decision leads us to some path. Maze structures come under such kind of problems. A maze is a grid-like two-dimensional area of any size. It is usually rectangular in shape. It consists of cells (formally bounded space). A cell is an elementary maze item which is interpreted as a single site. The maze contains different types of obstacles. Its complexity is determined by the number of cells, number of walls, number of hallways, dead-ends and the distance between the start-finish in the maze structure [3].
Some sources interchangeably refer to mazes also as labyrinths. The word labyrinth has a deep historic background in ancient Crete. It was a structure elaborated by Daedalus for king Minos. Its purpose was to hold back the Minotaur, a deadly creature, which was killed by Young Theseus of Athens who entered the labyrinth with a sword and a ball of string. He traced the path out of the labyrinth by unwinding the strings as he went along. The term labyrinth is associated with the construction which leads from starting point to the goal state by taking tortuous path but requires no actual decision [3].

The paper describes the method to generate a maze structure which has a unique solution. It also describes the method to find the optimum path for the mazes having multiple paths to the dead end.

Corresponding Author,
E-mail address: bhawana3022@yahoo.com

Fig: 1. The Cretan labyrinth

2. Related Work

Genetic algorithm, an evolutionary heuristic method for finding optimum solution, had been used to give solution of the rectangular maze. The GA is a stochastic global search method that mimics the metaphor of natural biological evolution (based on the Darwin's principle of origin of species by means of natural selection). GA's are invented by John Holland in 1960's, who wrote the first book on Genetic Algorithms 'Adaptation in Natural and Artificial Systems' in 1975. GAs operates on a population of potential solutions applying the principle of survival of the fittest to produce better and better approximations to a solution. At each generation, a new set of approximations is created by the process of selecting individuals according to their level of fitness in the problem domain and breeding them together using operators borrowed from natural genetics. This process leads to the population of individuals that are better suited to their environment than the individuals that they were created from, just as in natural adaptation.

Tremaux algorithm uses recursive backtracking procedure to find path from inside the maze to the goal position outside the maze. Dead End filling algorithm fails to get the optimum path when there exist more than one solution paths in the maze which otherwise works well in the case when the maze have multiple dead ends. A trivial method, referred as random maze solver, uses an unintelligent robot or a mouse which travels entire maze in random way but it is a slow method to find the solution. There are various other methods such as Left wall follower, Use of imaging, Partition based maze solving but no specific method gives solution to all the types of mazes [3].

Though genetic algorithms are good heuristics which returns near to optimum solution, huge evaluations for generations and populations sometime are time-consuming. To account some of the defects and employ the advantages of the GA, Hybrid Genetic Algorithm has been used.

3. Methodology Used

3.1 Disjoint-sets

In this paper a method, called Disjoint-sets, has been used to generate a puzzle maze. A puzzle maze has a choice of paths, some has dead-ends and some lead us round in a circle. A disjoint-set data structure maintains a collection S= {$S_{1,}$ $S_{2,}$ S_k} of disjoint dynamic sets. Each set is identified by a representative, which is some member of the set. If there are two sets S_x and $S_{y,}$ x is not equal to y, such that S_x= (3, 4, 5, 6, 7) and S_y= (1, 2) then these sets are called disjoint sets as there is no element which is common in both sets. In other words, Disjoint-sets are the data structure for problems requiring equivalence relations, that is, are two elements in the same equivalence class. An equivalence relation is a relation R such that the following hold:

- R is reflexive: (aRa) for all a belongs to S
- R is symmetric: (aRb) <-> (bRa)
- R is transitive: (aRb) and (bRc) -> (aRc)

Steps involved in generating a maze are as follows:
Step 1: Initialize a grid of rxc squares, where r denotes the number of rows and c denotes the number of columns.

```
Initialize( int r, int c)
{
    maze = new int [r*c];
    for (int e = 0; e < r*c; e++)
            maze[e] = e;
}
```

Step 2: Start with the entire grid subdivided into squares.

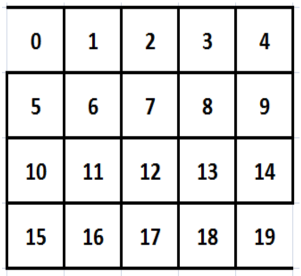

Fig: 2. Grid of 4x5 squares

Step 3: Represent each square as a seperate disjoint set.

Example-
{0} {1} {2} {3} {4} {5} {6} {7} {8} {9} {10} {11} {12} {13} {14} {15} {16} {17} {18} {19}

Step 4: Repeat the following algorithm:

1. Randomly choose a wall between two cells.
2. Check whether the two cells are adjacent:
   a. Let c denote the number of columns, r denotes the number of rows and i denote the cell.
   b. If (i<c) then there is no cell above the cell i.
   c. If (i>=(r*c-c)) then there is no cell below the cell i.
   d. If (i%c==0) then there is no cell to the left of cell i.
   e. If ((i+1) %c==0) then there is no cell to the right of cell i.

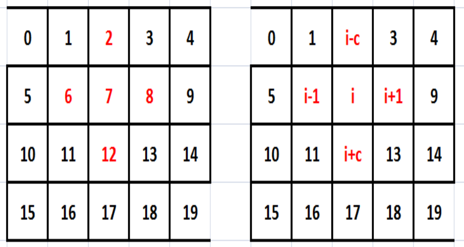

Fig: 3. Adjacent cells

3. Check whether the two cells are disjoint. That is, if Find(x)! = Find(y), where x and y are two cells, then they are said to be disjoint.

```
int find( int i )
{
    return maze[i];
}
```

4. If sets are disjoint then knock the wall = union the sets.

```
void UnionSets( int i, int j )
{
    if(isadjacent(i,j)==TRUE)
    {
```

```
                rooti=find(i);
                rootj=find(j);
                for (int k=0; k<r*c; k++)
                    if (maze[k] == rootj)
                        maze[k] = rooti;
        }
    }
```

Maze can also be represented as a graph {V,E}, where V is the collection of rooms and E is the connections between rooms (initially all closed). E’ is the collection of connections to knock down,where E’ $\subseteq$ E, such that a unique path connects any two rooms. Instead of using trees with pointers from each node to its children, a tree with a pointer from each node to its parent is used. Each subset is an up-tree with its root as its representative member. A Weighted Union is applied in which the root of the larger tree is made the new root. This often cuts down on height of the new up-tree.

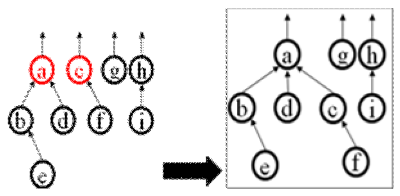

Fig: 4. Weighted Union

Path Compression is another heuristic that is used during FIND operation to make each node on the find path point directly to the root. It does not change any ranks.

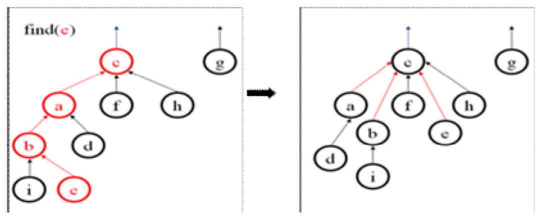

Fig: 5. Example of Path Compression

3.2 Hybrid Genetic Algorithm

The GA and the traditional optimization algorithms face the difficulties of a long trial-and error process in finding a better set of initial design variables or slow convergence. Therefore, how to select better initial values of the design variables is a critical step for those traditional methods. As for using the GA, it has the advantage of working in a random population. Even though a GA can locate the solution in the whole domain, it does not solve constraint problems easily, especially for exact constraints. In order to overcome these difficulties, a new hybrid optimization procedure, which combines the GA with traditional optimization methods, has been used. In the first step of the procedure, the GA is applied to provide a set of initial design variables, thereby avoiding the trial process; thereafter, traditional algorithms are employed to determine the optimum results. This hybrid algorithm, which can be termed a *Hybrid Genetic Algorithm* (HGA), is more effective than traditional algorithms [1]. Steps involved are as follows:

Step 1: Produce initial population in random, let it be (t), and initialize t=0.
Step 2: Encode the string into binary (or other forms based on the encoding methods).
Step 3: Calculate fitness of the individuals in the population.
Step 4: Select the best fit from current population.
Step 5: Perform crossover on the selected strings.
Step 6: Perform mutation.
Step 7: Increment t=t+1.
Step 8: Check if generation is complete or not. If no, then goto step 4.
Step 9: Initialize initial population t1 to Z.
Step 10: Perform optimization by using the traditional method with initial design variable Z.
Step 11: Check if converged or not, that is, a better solution is found or not. If yes then exit, else assign Z to Z_0 and goto step 10.

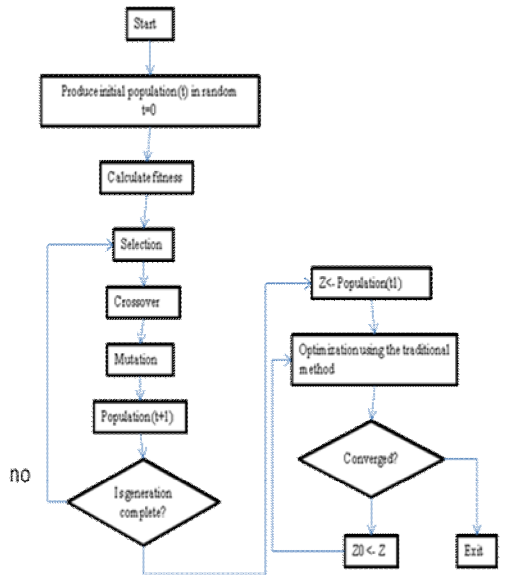

Fig: 6. Flowchart of HGA

Various components of genetic algorithm are explained as follows [3]:

1. Encoding

The chromosome uses decimal numbers from 0 to 3 (including both) for representing the path from one cell to another. To get the optimal path the chromosome length is kept at the minimum value of the size twice that of the size of the maze and it is further planned to increase over the generations gradually.

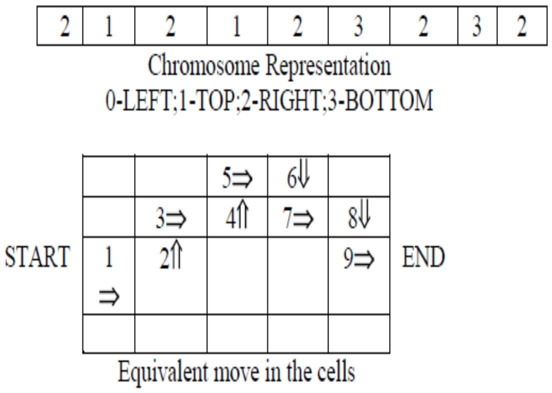
2 1 2 1 2 3 2 3 2
Chromosome Representation
0-LEFT;1-TOP;2-RIGHT;3-BOTTOM
START
END
1 ⇒
2⇑
3⇒
4⇑
5⇒
6⇓
7⇒
8⇓
9⇒
Equivalent move in the cells

Fig: 7. Sample chromosome and its equivalent move representation.

2. Fitness Function

The fitness function used for the genetic algorithm takes the use of the move made by the chromosome mapping in order to reach near to the finish cell.

FV= ((CC-SC)/ (FC-CC))*100

Where, FV- fitness value, CC- current cell, SC- start cell, FC- finish cell.

3. Crossover

Two parents are selected from parent population and two off-springs are generated by performing addition and subtraction operators on the respective genes in the parent. First child is the result of addition operation whereas second child is the result of subtraction operation. The "n modulo 4" is taken on all genes in resultant offspring to restrict the gene value within required range of value (0-3).

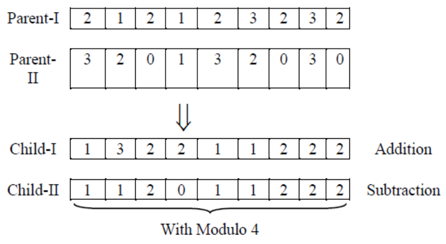
Parent-I 2 1 2 1 2 3 2 3 2
Parent-II 3 2 0 1 3 2 0 3 0
Child-I 1 3 2 2 1 1 2 2 2
Addition
Child-II 1 1 2 0 1 1 2 2 2
Subtraction
With Modulo 4

Fig: 8. Crossover

4. Mutation

Random-bit mutation has been performed in which a random number (0-3) is replaced at randomly selected place in chromosome.

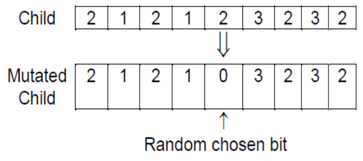

Fig: 9. Mutation

4. Recursive Maze Algorithm

The traditional method used in the HGA is backtracking to perform optimization. Backtracking is a methodical way of trying out various sequences of decisions, until we find one that "works". Recursive maze algorithm is one of the good examples for backtracking algorithms. In fact Recursive maze algorithm is one of the most available solutions for solving maze.

As explained above, in maze we have to travel from the starting point to ending point. The problem is to choose the path. If we find any dead-end before ending point, we have to backtrack and change the direction. The direction for traversing is North, East, West and South. We have to continue "move and backtrack" until we reach the ending point.

Assume that we are having a two-dimensional maze cell [WIDTH][HEIGHT]. Here cell [x][y]=1 denotes wall and cell [x][y]=0 denotes free cell in the particular location x,y in the maze. The first step is to make the boundary of the two-dimensional array as 1 so that we won't go out of the maze, and always reside inside the maze at any time.

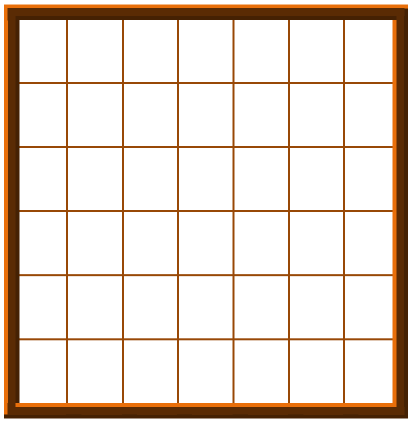

Fig: 10. Example Maze

Now start moving from the starting position (since the boundary is filled by 1) and find the next free cell then move to the next free cell and so on. If we reach a dead-end, we have to backtrack and make the cells in the path as 1(wall). Continue the same process till the ending point is reached.

5. Conclusion

The proposed method is useful in finding the optimum path for the maze structure. Hybrid genetic algorithm is more efficient than traditional algorithms. Further other methods can be applied for generating the maze which has multiple paths to the dead-end. In this paper disjoint-sets have been used to generate the maze structure that gives a unique solution. This method can also be used for more complex maze structure with different geometrical structure than rectangular structure.

References

[1] P. Guo, X. Wang, Y. Han, “The Enhanced Genetic Algorithms for the Optimization Design”, School of Civil and Architectural Engineering, Liaoning University of Technology, Jinzhou, China, 2010 3rd International Conference on Biomedical Engineering and Informatics (BMEI 2010).

[2] T. Pasquier, J. Erdogan,”Genetic Algorithm Optimization in Maze Solving Problem”, Institut Superieur d'Electronique de Paris

[3] N. S. Choubey, “A-Mazer with Genetic Algorithm”, MPSTME, SVKM’s NMIMS, Shirpur, Maharashtra, India, International Journal of Computer Applications (0975-8887) 58 (17), November 2012

[4] T. Sukumar, Dr. K. R. Santha, “Maze Based Data Hiding Using Back Tracker Algorithm”, Department of IT, SVCE and Department of EEE, SVCE, Anna University, India, International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622, 2 (4), July-August 2012, pp. 499-504.

[5] D. G. Sotiropoulos, E. C.Stavropoulos, M. N. Vrahatis, “A New Hybrid Genetic Algorithm For Global Optimization”, Department of Mathematics, University of Patras, Greece, Nonlinear Analysis Theory, Methods & Applications, 30 (7), pp. 4529-4538,1997.

[6] Maze Generation, ece.uwaterloo.ca.

[7] Maze classification, www.astrolog.org/labyrnth/ algrithm.htm

[8] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, “Introduction to Algorithms”, 2nd Edition, Pearson Education.

[9] Aho, Ullman, Hopcroft, “Design and Analysis of algorithms”, Pearson Education.

[10] Union-Find Algorithm, www.cs.princeton.edu /~rs/AlgsDS07/01UnionFind.