Advanced Genetic Approach to Maze Solver

**ABSTRACT**

*The maze traversal problem has been an interesting challenge in computer intelligence. Recent works have shown that Genetic Programming can be used for computer based problem solving such as maze traversal. This paper focuses on exploiting relevant information about the maze traversal problem to improve the learning and to reduce the training time to solve mazes using genetic programming. This paper seeks to improve the performance of existing algorithm. Appropriate changes have been made in the genetic operators so as to come up with new path searching methodology. The proposed algorithm was implemented on 10 randomly created 2D rectangular mazes each of size 30×30 with different complexity.*

**Keywords**

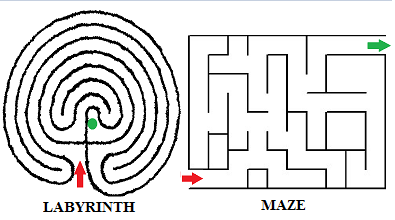
Genetic Algorithm, Maze structure, Slime-Mold, Maze runner, Depth-First search

**1. INTRODUCTION**

*The central challenge and common goal of computational intelligence and machine learning is to get a computer to solve a problem without explicitly programming it. This challenge envisions an automatic system that must routinely achieve satisfactory solution to a given problem at levels that equals or exceeds the human level of performance.*

*Genetic programming has achieved this goal with an automated solution to a problem. It has produced results that are comparable with human produced results in areas such as: control, design, gaming and pattern recognition. One such problem in which genetic approach is widely used these days is maze traversal problem.*

A maze is a grid like structure of any size, usually rectangular in shape, which consists of cells as the elementary unit [1]. *The concept of maze traversal came up since a long time.* Mazes are believed to be originated in ancient Greece, when the first labyrinth (similar to maze but not a maze) was built to hold a deadly creature, Minotour. *In the long run, many people have developed and demonstrated various techniques to solve maze problems. As the technology evolved further, mazes have been applied more often within the real life, which brought up the need to have automated mazes.*



*Various algorithmic methods have been proposed previously to solve various types of mazes*. ***Wall following technique*** uses wall as reference for its movement while a ***random mouse algorithm*** is a slow algorithm which travels a maze randomly to reach the destination. ***Tremaux algorithm*** uses a recursive backtracking method to find the path and ***Dead-end filling method*** requires prior knowledge of maze and is not effective in case of maze with multiple paths. Various *other methods have been studied in several works to solve maze problem* [1][3][4]*.*

*In this paper, genetic algorithm is chosen to solve the maze due to its efficiency and simplicity. The paper introduces new modified genetic operators, which seeks to improve the performance of existing algorithm.* Mazes considered in this paper are randomly generated 2D computerized mazes of size 30×30. Slime Mold algorithm is introduced here to determine the route to the destination point.

**2. METHODOLOGY USED**

**2.1 Maze creation**

*The 2D maze representation has been used for this problem. The maze is represented as a square matrix with each cell classified as an obstacle, pathway or goal. 10* Randomly generated mazes each of size 30×30 are used to generalize the test results. *The size of the maze is kept the same in all the mazes so as to maintain uniformity in population (number of chromosomes).* Depth-First search technique used to generate the random mazes is described below:

1. Initialize the maze

2. Select any cell as starting cell.

3. Push the position of current cell into the stack.

4. For each cell in maze repeat 5 and 6.

5. Find all possible neighbor cells which are unvisited from current cell.

6. If found, select any one of them randomly, move to it and push its position into the stack. Else 6.

7. Pop from the stack and Backtrack to last visited cell.

[End of step 2 loop structure]

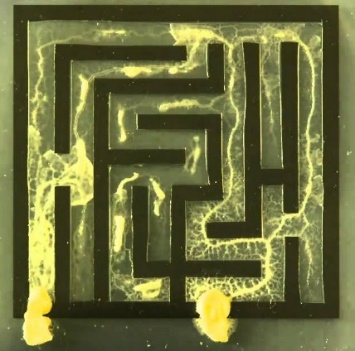
8. End

**Algo. 1. Maze generation using Depth-First search [7]**

**2.2 Slime-Mold algorithm**

*Slime-Mold algorithm is used as a supporting algorithm to evaluate the fitness of an individual in the population. Slime-Mold is a single cell organism having no brain or nervous system but is capable of making surprisingly sophisticated decisions. It can mimic the man-made transport network, find a shortest route through a maze to the food, and find the healthiest food from the menu [8].*

*Slime-Mold, when exposed to a maze, first extends its tendrils through the maze leaving behind a translucent slime. It then retracts every tendril that does not find the food and grow exclusively on the leftover slime avoiding retracted path* [9]. Slime-Mold algorithm used in this paper is explained below:

****

**Fig. 3. Physarum polycephalum**

1. Set the initial tendril to the current position of the runner;

2. Until destination not found do 3, 4 and 5;

3. For each moving tendril do 4 and 5;

4. N=cells unvisited from current position of moving tendril;

5. If (N==True)

{A. Spread tendril in each unvisited cell;

B. set those cells visited; }

Else

{a. Retract tendril to last visited junction which has more than one possible path present ;}

[End of step 3 loop]

[End of step 2 loop]

6. Retrace every tendril except the one which traced right path;

7. Move the runner along the left over tendril;

8. End;

**Algo. 2. Shortest Path using Slime-Mold Algorithm**

**2.3 Genetic Algorithm**

*Genetic algorithm is an evolutionary algorithm* *based on the principle of genetics and Darwin’s principle of natural selection [6]. In this paper, it is used as a search and optimization technique. A GA allows individuals of a population to evolve under specific selection rule. And only those individuals with higher fitness are selected that maximizes the “fitness” of the solution.*

*Genetic algorithm is a heuristic method that returns near to optimal solution. Here, GA starts by tracing all the possible paths to reach the destination point. It then evaluates each individual solutions from the population (using the fitness test) to determine which one is better for solving the problem. Good individual, that solves the maze in the shortest time, would be ranked more prior. Next, these individuals would reproduce with each other to produce a new population.*

*The reproduction occurs in such a way that highly ranked individuals reproduce more frequently. So, it can be expected that the best individual in the new population would be better than the best individual of the previous population. Over time the population would improve and the best individuals in successive generation would be a more efficient solution to the problem.*

Basic Genetic Algorithm is shown in Fig. 5.

1. Initialize the population;

2. Generation=0;

3. While (termination condition) do 4,5,6 and 7

4. Evaluate each individual;

5. Select the best individuals and produce the offsprings;

6. Replace the offprings with the weak individuals of the current population;

7. generation++;

[End of step 2 loop]

8. End;

**Fig. 5. Genetic Algorithm**

2.3.1 Chromosome Representation

Each chromosome in the population represents a runner with random solution to the maze which may or may not solve the maze completely. The chromosome is an array with each element (gene) *being* represented by a set of 4 tuples (D1, D2, D3, D4). Each tuple represents the direction the runner can choose *to run* at any junction. The directions are randomly generated by different combinations of 4 possible directions (L=Left, R=Right, D=Down, U=Up) *and is considered to be prioritized. The first element, which is considered to have the highest priority, is chosen at a node (say n1) and the possible path is found out. If it does not solve the maze then the next element of the tuple (at node n1) is considered. To get the optimal path, the length of chromosome* is chosen as 900, equal to the total number of cells in the maze.

2.3.2 Fitness function

*Once the initial random population of runners has been created, each individual needs to be assessed for their fitness.* ***The fitness value represents the quality measure of an individual, which helps the algorithm to select individuals with better gene to produce new individuals for next generation.***

*Fitness test is a problem specific test. In this case, a smaller fitness value indicates a better individual. The chromosome with minimum fitness value will be chosen to create the optimal solution.*

Fitness of a runner is *measured by* *the sum of* the moves the runner runs *towards* the destination and the distance between that point and the destination. Fitness value describes both of these parameters for any runner and can be calculated as

Fitness (runneri ) = movesi + Slime\_mold ( desti )

Where,

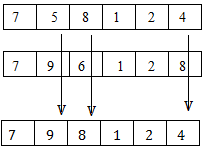
Movesi - represents the number of moves the ith runner run towards the destination

Slime\_mold (desti) -function returns the distance to the destination from the current position.

2.3.3 Crossover

*Crossover is a genetic operator that is a source of new and eventually better individuals. It generates variants in the new population by mixing the elements (genes) of two individuals. In this technique, genetic materials from two individuals (called parents) are mixed to form a new individual (called offspring) for a new generation (new population). The probability of selection of an individual for crossover is proportional to its fitness.*

*If both parents have same gene then the gene is copied to the child.* If the gene value in parents differs, one parent is randomly chosen as dominant and its gene is copied to the child.

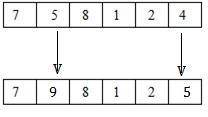


**Fig. 6. Crossover operation**

2.3.4 Mutation

*Mutation is also a typical genetic operator that randomly varies the genes within an individual. It randomly selects one or more genes from an individual and changes its value to a possible random value.*

Mutation prevents the population from falling in local optima and allows searching other parts of search area.

**Fig.7. Mutation**

**2.4 Enhanced Genetic Implementation**

***The*** *scope* ***of genetic algorithm lies in the fact that*** *it is* ***capable of finding global optimum in multi-model space.***

Basic genetic operators were found to *produce noticeably less optimum result* with very slow convergence. Moreover, crossover and mutation operator *does not* *always* ensure the production of offspring with good genetic material. The main idea of improving genetic operators is to enable the population to enjoy faster convergence and to *assure* offspring *with* good genetic material.

Modified crossover operator starts with the comparison of every gene of two parents. If the two corresponding genes are same then gene is copied to offspring. If the gene values are distinct, offspring is produced with respect to each parent’s gene and evaluated. The gene from a parent which produces better offspring is selected. Complete offspring is produced at the end of the process.

Mutation operator is all about bringing new genetic material in the population which allows GA to achieve global optima. Any improper change in the mutation operator and in its randomness may lead the population to local optima. Improved mutation operator allows only good genetic material to exist in the population. Gene selected for mutation is set to a random value. If mutated gene increases the fitness of the individual then the change is kept else the previous gene value is restored. Each time mutation occurs, genes are selected randomly, set to a random value and only good genetic material is allowed to exist. Such modification in the mutation operator leads to the faster convergence as well as prevent the occurrence of the local optima.

Individuals in the population are arranged according to their fitness. The best individual (with lowest fitness value) is placed in the beginning and the worst is at last. Individuals with good genetic material are selected for crossover and produced offspring are placed in the population by replacing individuals from the last. This protection mechanism ensures the existence of good genetic material in the population.

1. Select par1 and par2

2. for each gene in (par1, par2)

3. if(par1[gene]==par2[gene])

Then child[gene]=par1[gene];

Else

{ child[gene]=par1[gene];

X=fitness(child);

Child[gene]=par2[gene];

If(fitness(child)>X)

Then Child[gene]=par1[gene];

}

[end of step 2 loop]

4. Exit

**Fig.8. Improved Crossover Operator**

1. for individual par

2. x=fitness(par);

3. randomly select genes from par

4. for each selected gene

5. par[gene]= random[gene];

6. if(fitness(par)>x) restore previous gene value.

[end of step 4 loop]

7. Exit

**Fig.9. Improved Mutation Operator**

**3. EXERIMENTAL RESULT**

The experiment was tested with 10 randomly created mazes each of size 30×30. Solving maze with only the basic operators encountered very slow convergence. Fig. 10 shows the experimental results with basic genetic operators. Very less improvement was observed in best individuals of populations in case of basic genetic operators.

The smart genetic operator has shown much better results (Fig. 11). Faster convergence and noticeable improvement in individual’s fitness in lesser number of generations was observed with smart genetic operators. Fig. 11 also shows that almost for every maze the best result is achieved in less than 1000 generations. Fig 12 clearly described the significant improvement achieved by improving the traditional Genetic Algorithm which allows the population to obtain best fitness in much lesser Generations.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Maze Number | Initial Fitness Value | Traditional G.A. Result | | Improved G.A. Result | |
|  |  | Best Fitness value | Achieved at Generation No. | Best Fitness value | Achieved at Generation No. |
| 1 | 1208 | 1012 | 148 | 1012 | 58 |
| 2 | 1148 | 892 | 4496 | 700 | 15 |
| 3 | 1092 | 888 | 4696 | 888 | 15 |
| 4 | 1188 | 732 | 4226 | 732 | 54 |
| 5 | 916 | 348 | 45 | 348 | 9 |
| 6 | 1420 | 1152 | 991 | 1152 | 1613 |
| 7 | 1216 | 948 | 3973 | 948 | 40 |
| 8 | 1456 | 1208 | 4849 | 1208 | 629 |
| 9 | 1156 | 884 | 3833 | 884 | 68 |
| 10 | 1264 | 998 | 1171 | 988 | 31 |

**4. CONCLUSION**

Genetic algorithm was implemented and found to be effective in solving maze problems. Significant improvement in Genetic algorithm was achieved by using smart genetic operators. The smart algorithm converges much faster than basic algorithm and approaches to the best solution quickly in lesser number of generations.

Further improvements will have to be made in genetic algorithm to completely solve more complex maze problems. To find the shortest route to the destination, it could be useful to generate the initial population in which each individual has a solution of maze and evolve the population to get the best solution. Modified crossover operator must be used which could ensure the production of best offsprings from the available genetic material of the parents. More sophisticated mutation operator should be used to produce best possible mutated child with the mechanism to prevent to occurrence of local optima.

**5. REFERENCES**

[1] Choubey N.S., “A-Mazer with Genetic Algorithm”, International Journal of Computer Applications, Vol. 58, pp. 48-54, November 2012.

[2]. The Labyrinth Builders website. [Online]. Available: <http://www.labyrinthbuilders.co.uk/>

[3]. Jianping Cai, Xuting Wan, Meimei Huo, Jianzhong Wu. An Algorithm of Micro Mouse Maze solving. 10th IEEE International Conference on Computer and Information Technology (CIT 2010), 2010.

[4]. Choubey N.S. & Sonawane S. R., “Comparative Study of various maze solving algorithms”, International Conference in Recent Trends, (i-CORT2012) , IOKCOE, Pune, 2012.

[5] DARWIN C., 1859, The origin of species by means of natural selection, 1859.

[6]. Randy L. Haupt and Sue Ellen Haupt, “Practical Genetic Algorithms”, John Wiley and Sons, 2004

[7]. “Maze creation algorithm- depth first search”. [Online]. Available: <http://www.algosome.com/articles/maze-generation-depth-first.html>.

[8]. “Why slime molds can solve maze better than robots”. [Online]. Available: <http://io9.com/5950984/why-slime-molds-can-solve-mazes-better-than-robots>

[9]. “How Brainless Slime Molds Redefine Intelligence”. [online]. Available:<http://www.scientificamerican.com/article/brainless-slime-molds/>