ISSN (Online) : 2278-1021
ISSN (Print) : 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 2, February 2015

Comparative study of Recursive and OpenMp Genetic algorithm in solving Towers of Hanoi

Prashanth V

UG Student, Computer Science, RV College of Engineering, Bangalore, India

Abstract: Towers of Hanoi is a standard planning problem in the field of computer science. It has applications in various fields of engineering. Solution to Towers of Hanoi is generally provided by method of recursion. Recursion is a functional programming concept widely used to solve variety of problems. Recursion has been providing solution to this problem most efficiently; hence other methods of solving have not been explored. In this paper we study artificial intelligence method of solving the problem. Artificial intelligence is a field of study which aims at creating intelligence in computer systems. Genetic algorithm is a well-known algorithm for optimization used in artificial intelligence. In this paper we have studied a method of solving Towers of Hanoi using genetic algorithm. Parallelism is achieved through OpenMp.We propose a comparative study of both Genetic algorithm and Recursive algorithm in solving Towers of Hanoi puzzle. Results for the same are presented.

Keywords: Towers of Hanoi, Genetic algorithm, Recursion, Roulette Wheel selection, OpenMp

I. INTRODUCTION

Artificial intelligence is an emerging field of study. If implemented efficiently, it is a field with potential to change the world. It is a dream of every engineer to build system which is intelligent. In this paper we study a method based on artificial intelligence in solving Towers of Hanoi [1]. The reason for study of this method is because it is a method which solves the problem without being explicitly programmed. Towers of Hanoi are a standard recursive problem in the field of computer science. It is well known mathematical puzzle invented by Edouard Lucas (hence also known as Lucas' Tower). The problem basically consists of three rods A, B and C. A known number of disks of unequal sizes are placed in one of the three rod (consider rod A). The problem is complete when all the disks are moved from one rod (rod A) to another rod (rod B) by making use of an auxiliary rod (rod B).The constraints for transfer of disks has following constraints.

- Only one disk can be moved at a time.
- No disk has to be placed on top of a smaller disk.
- Disk at the top of each rod has to be moved to either of other two rods. Each transfer of disk from one rod to another is known as a move.

The problem has to be completed in minimum number of moves without violating the constraints. A known recursive method provides solution in 2^n - 1 moves, where n is the number of disks. Thus number of moves for 3 disks is 7. We show that artificial intelligence method also solves the problem in the minimum number of moves on most of the occasions. The two algorithms are separately considered and analysed on basis of various factors in this paper. Results using both the algorithms in solving 3 disk and 4 disk problems are presented. OpenMp is used as a parallelizing tool to increase the efficiency of genetic algorithm. Genetic algorithm is a continuous algorithm whereas Towers of Hanoi is a discrete valued problem; hence the study of this algorithm can solve many other real world problems. It relates discreteness and continuity two contrasting fields.

II. RECURSIVE SOLUTION TO TOWERS OF HANOI

Recursion is a process of breaking an object into smaller objects of its own kind. Recursion is functional programming concept and is used very much in mathematics and computer science [2]. Objects are said to exhibit recursive behaviour if they satisfy two properties:

- A base case
- Rules that reduce the problem towards base case.

Towers of Hanoi are solved using recursion making use of dynamic programming principle. Consider the problem to consist of n disks numbered from 1 to n. Let A, B and C be the from peg, auxiliary peg and to peg respectively. Let the problem of moving n disks be represented as F(n). The problem is solved if top n-1 disks are moved from A to B, & the remaining nth disk is moved from A to C, then again the n-1 disks are moved again from B to C. Here the act of moving 1 disk is the base case and moving n-1 disks is the divided problem. Since F(n) represents the operation of moving n disks F(n-1) represents the act of moving n-1 disks. Hence the solution is obtained by moving n-1 disks twice (i.e. from A to B and B to C) and moving the nth disc once. We formulate the above as a recurrence relation.

$$F(n)=2F(n-1)+1$$

This recurrence relation on solving will give the minimum number of moves to solve the puzzle as $2^n - 1$.

The following procedure when written as a code is implemented in following way

```
void
towers (int n, char A, char C, char B)
{
   if (n == 1)  //base case
 {
      //print from peg to peg move
      return;
 }
   towers(n- 1, A, B, C); //1st recursive call
   //print from peg to peg move
   towers (n - 1, B, C, A); //2nd recursive call
```


DOI 10.17148/IJARCCE.2015.4209

ISSN (Online) : 2278-1021
ISSN (Print) : 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 2, February 2015

}
The above function towers calls itself repeatedly until certain conditions are satisfied. The base condition is given by n=1 i.e. number of disk is one. The solution for 4 or 5 rods towers of Hanoi problem cannot be solved by the same algorithm this is the major flaw in the recursive algorithm. Time complexity of algorithm is $O(2^n)$ or exponential. Most recursive algorithms have exponential time complexity hence it is very difficult to implement them in iterative methods.

III. OPENMP GENETIC ALGORITHM METHOD

Genetic algorithms are well known evolutionary algorithm used in global optimization problems. Genetic algorithms are comparatively slow in solving problems hence they need to parallelize for greater efficiency. Genetic algorithm was initially implemented in C language by Tim.M.Jones [11]. Modified version of this algorithm using OpenMp[4] is implemented in this paper. OpenMp is an API that supports multi-platform shared memory multiprocessing programming in C, C++ and FORTRAN. Parallelization of code using OpenMp has increased efficiency and has better memory utilization than dual population parallel genetic algorithm. The OpenMp library can be imported into program including <omp.h> header file

Solving Towers of Hanoi using evolutionary computation has many steps.

Step1. First we have to represent the sequence of moves in the form that can be evolved by the genetic algorithm. We considered all the moves that are possible. Since we consider the problem with 3 pegs, the maximum number of moves possible are 3!= 6. Though we consider all the moves some of them are not feasible, these moves are removed or avoided by genetic algorithm when it is simulated. The pegs are represented as 0 1 and 2 respectively, hence move from peg 0 to 1 is encoded as 01 similarly all moves are represented in the solution space. Hence total number of moves is represented from 0 to 5. The output to the problem is printed as 01 02 12 and so on, 01 depicts that a disk is moved from peg 0 to 1.

Step2. We consider 2 structures one to represent the pegs. The pegs structure contains a pegs array to depict the pegs, and a count value to count number of disks in each peg. Then another structure is used to provide the solution it contains the current fitness value, the sequence of moves and also a variable to count number of legal operations.

Step3. We then define the values for maximum number of generations for the genetic algorithm to evolve and maximum number of operations. The mutation and crossover probability also have to be defined or set to default values. We have to take care that number of operations must in the order of the number of disks taken; if too large values are taken minimum number of moves might be very different from what we expect. If we define the operations too less the algorithm takes large amount of time to compute. These are known as under fitting and over fitting in optimization.

Step4. We split population into two arrays one to store the current generation and another array to store the next generation. Hence solution is represented as double dimensional array of the form solution[2][Population_size].

Step5. After the basic initialization and defining of default values, we enter the main loop of genetic algorithm. First in the main function we begin by randomly generating and initializing the population. We then compute the fitness of the randomly initialized population. Initially the population are set to zero. We then call the genetic algorithm function.

Step6. The genetic algorithm function receives current population as input. Then the algorithm walks through entire population and selects two parents based on their fitness value. Parents with highest fitness have higher probability of getting selected. Then crossover is achieved between the 2 parents by swapping the tails at appropriate points. We also check that number of moves is minimum, when crossover is achieved. Crossover includes both possibility of mutation and selection based on the probability. Then the genes are copied into the child to continue the evolution process.

Step7. We used a roulette wheel [5] based selection method for selecting the fittest parent. This function walks through the population by comparing population's fitness value with the randomly generated value. If the random value is less than the fitness value of population, then the value is selected and propagated, i.e. less fit members are allowed to propagate. If no member is selected the random value is returned.

Step8. Then finally we considered the fitness function. The fitness function starts of by initializing the Towers of Hanoi simulation. Then it is iterated to perform the specified move. If an illegal move is attempted the illegal move counter is incremented, but the move is not performed. We found out the fitness value by assign a constant value to each disk that is on the correct peg. Then this value is subtracted from number of illegal moves so that no illegal moves are attempted.

Step9. The above algorithm when simulated provides the solution to Towers of Hanoi problem. In certain cases exact solution is not obtained, this is mainly due to less population diversity. This problem can be corrected by running the code many times or providing an automatic restart mechanism.

Step10. The main genetic algorithm function continues execution until there is adequate diversity in population or until the fitness value reaches an expected value [11].

DOI 10.17148/IJARCCE.2015.4209

ISSN (Online) : 2278-1021
ISSN (Print) : 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 2, February 2015

Step11. Once the algorithm is coded we parallelized it using OpenMp. #pragma omp parallel is the directive used for parallelizing. The entire main loop of the program is included inside the directive. This directive achieves parallelization by splitting the process into number of available threads. There is an associated thread id with each thread. Considerable improvement in performance is achieved in this. The code segment for parallelization is as follows

```
int threads, tid;/*declare thread and thread id variables
/* split a team of threads giving them their own copies of variables */
#pragma omp parallel private(threads, tid)
  {

  /* thread number is generated and stored in tid */
  tid = omp_get_thread_num();

  /* master thread id is checked and total number of threads is printed. This action can be performed only by master thread */
  if (tid == 0)
    {
    threads = omp_get_num_threads();
    printf("Number of threads = %d\n", threads);
    }
}
```

In general running time of genetic algorithm depends on the population size and also on the methods of implementation. Genetic algorithm is generic and can be used for all problems solving in general. The above algorithm can be used to solve 4 pegs or 5 pegs problem conveniently by just making use of appropriate solution space.

IV. RESULTS AND DISCUSSION

The above considered methods are implemented in Intel core i5 processor and 8 GB RAM. The following results were obtained when preformed on 3 and 4 disks problem.

Run no	Recurs-ion 3 disks	GA 3 disks	OpenMp GA 3 disks	4 disks Recur-sion	4 disks GA	4 disks Open Mp GA
1	0.0002	4.56	2.98	0.0001	1.37	1.51
2	0.0002	22.16	1.38	0.0002	31.1	0.62
3	0.0001	7.03	12.18	0.0002	2.47	4.31
4	0.0002	11.30	0.66	0.0002	1.42	0.73
5	0.0002	54.38	2.51	0.0002	11.6	5.27
6	0.0001	7.98	0.46	0.0002	8.69	2.81
7	0.0002	26.02	13.55	0.0002	6.76	0.87

Table -1: Running time of Towers of Hanoi in seconds.

It is seen that recursive algorithm has running time of order 0.0002 seconds. Whereas running time for genetic algorithm is of the order of 7-8 seconds. The parallel genetic algorithm has higher efficiency than normal genetic algorithm in most of the cases. The above results are obtained with setting of probability of crossover as 0.09 and probability of mutation as 0.01. Number of operations is set as 30 and number of generations is set as 100000. The running time is computed by using clock() function available in standard library.

V. CONCLUSION AND FUTURE WORK

From the above results presented it can be concluded that speed of execution of Parallel genetic algorithm is much higher than normal genetic algorithm. Average time for 3 disk execution is over 15 seconds in case of normal genetic algorithm but this time reduces to up to 5 seconds in case of OpenMp genetic algorithm. However in case 4 disk Towers of Hanoi the time variation is not much, since the number of operations is less. Recursive algorithm is most efficient and its speed is very much higher compared to genetic algorithm. Hence we can conclude that Genetic algorithm provides an inefficient solution when compared to recursive algorithm. Recursion method is unique for the problem and the algorithm has to change in case of 4 peg problem. The programmer had to think the logic for solving the problem. Genetic algorithm is advantageous in these cases, since it is an artificial intelligence method and provides an intuitive solution. The speed of execution of the algorithm can improved even further using GPU and high processing power systems. Thus parallel genetic algorithm can be used for all kinds of applications provided high processing power systems are available. They are efficient in solving small input problems. The future work can be done in this regard to improve speed of execution and make the algorithm work for large input problems. Hence solving problems using artificial intelligence methods is region where much research can be done.

ACKNOWLEDGEMENT

I would like to thank my teachers, parents and my friends for helping me out with this paper. It wouldn't have been possible without them.

REFERENCES

[1] Mindless intelligence method for solving Towers of Hanoi by TSAO Minhe, Kao Weiven, CHANG Albert Chinese journal of mechanical engineering vol 22.no 2, 2009.
[2] Mathematics for Computer Science by Srini Devadas and Eric Lehman 6.042/18.062J.
[3] T. Park, K. Ruy, ―A Dual-Population Genetic Algorithm for Adaptive Diversity Control‖, IEEE transactions on evolutionary computation, vol. 14, no.6, pp 865-883, Dec.2010, pp 865-883. DOI:10.1109/TEVC.2010.2043362
[4] http://openmp.org, June 2013.
[5] Genetic Algorithm Programming Environments Jose Ribeiro Filho, Cesare Alippi and Philip Treleaven Department of Computer Science – University College London.
[6] http///;www.wikipedia.com
[7] [Banzhaf 1998] Banzhaf, W., Nordin, P., Keller, R.E., Francone, F.D., Genetic Programming: An Introduction: On the Automatic Evolution of Computer Programs and Its Applications, Morgan Kaufmann, 1998.

ISSN (Online) : 2278-1021
ISSN (Print) : 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 2, February 2015

[8] [EvoNews 1999] "Professor Hans-Paul Schwefel talks to EvoNews."1999.Available online at: http://evonet.lri.fr/evoweb/news_events/news_features/article.php id=5

[9] [Fogel 1966] Fogel, L.J., Owens, A.J., Walsh, M.J. Artificial Intelligence through Simulated Evolution. Wiley, New York, 1966.

[10] [Levenick 1991] Levenick, James R. "Inserting Introns Improves Genetic Algorithm Success Rate: Taking a Cue from Biology." Proceedings on the Fourth International Conference on Genetic Algorithms, 1991.

[11] Jones, T. (2009). *Artificial Intelligence.* A Systems Approach. Jones and Bartlett Publishers, LLC. MA, USA.