An Evolutionary Approach to Tower of Hanoi Problem

Chapter · January 2015

DOI: 10.1007/978-3-319-12286-1_10

2 authors, including:

Jie Li
Cspace Intelligent Technology Co.,Ltd.
6 PUBLICATIONS **21** CITATIONS

SEE PROFILE

An Evolutionary Approach to Tower of Hanoi Problem

Jie Li, Rui Shen

Department of Computer Engineering
Shandong Aerospace Electro-Technology Institute
Yantai, China

jie_yi_ehw@163.com

Abstract: The Tower of Hanoi problem is an ancient and interesting topic. In this paper, we presented an evolutionary algorithm approach for searching the solutions of the problem. We use a direct encoding and apply mutation only in the evolution. Experimental results are reported and show that the proposed method is capable of finding solutions for the problem of multiple pegs.

Keywords: Tower of Hanoi; Evolutionary approach; multiple pegs problem

1 Introduction

The Tower of Hanoi problem, proposed by Lucas over a hundred years ago [1], is a well-known game that transferring a number of disks to a goal position. In the game, there exists three vertical pegs, and a player is given a certain number of disks (typically 3) of mutually different diameters place in small-on-large ordering on a peg. The task is to get from a given initial state to a target state by moving a single disk from the top of the peg to the top of another possible one, while obeying the following rules:

(1) Each time only one disk is moved;
(2) Only the topmost disk can be moved;
(3) At any moment, a disk cannot reside on a smaller one.

The Tower of Hanoi problem has been widely discussed in [2], [3], [4] and is being applied in many fields. In computer programming course, it is the most popular example for recursive programming [5], often comparing with the iterative solution, and showing that the recursive one is short and elegant. In psychology researches, the Tower of Hanoi is used as a tool to investigate how humans develop their problem solving ability [6], [7]. Many science and engineering activities require planning. Solving the Tower of Hanoi problem is a good example for modeling the process of planning, for instance, robot task plan [8], and unmanned vehicle route planning [9].

Work on this problem still goes on, studying properties of solution instances, as well as variants of the original problem. A natural extension of the original problem is obtained by adding pegs. In this work, we applied evolutionary algorithm to search adequate solutions for the Tower of Hanoi problem. Each particular moving is as-

signed by a specific integer number, which denotes a gene in EA. Thus, a series of disks moving steps can be described by a string of certain fixed length. All possible candidates are composed of such strings. The evolutionary algorithm checks the validation of the strings, and finally finds the solution. This work focuses on the problems of 3, 4 and 5 pegs. Experiment results show that our approach is able to solve the Tower of Hanoi problem of 3-peg with 5 disks, 4-peg with 8 disks and 5-peg with 9 disks.

The rest of the paper is organized as follows: Section 2 presents the detail of the non-determination approach. Section 3 describes the evolutionary algorithm applied in the work. Section 4 shows the experimental results. Section 5 gives the conclusion of the paper.

2 proposed method

2.1 Problem description

The classic Tower of Hanoi problem has three pegs, denoted as A, B, C, and n ($\geq$1) disks of different size. All disks initially rest on the source peg in a tower in small-on-large ordering, generally with the largest disk at the bottom, the second largest one above it, and so on, with the smallest one at the top. The objective is to transfer the tower from the source peg to the destination peg, with legal moves. The legal move is considered as the operation that can transfer the topmost disk from any peg to another without breaking the rules mentioned above.

Fig. 1 shows the initial and goal states of a 3-disk classical Tower of Hanoi problem. There are 3 disks, D1, D2, and D3 of increasing size. Initially, all the disks are on stake A, the source peg, while the target peg is stake C.

For the version of multi-peg Tower of Hanoi problems, the goal of the game is the same - moving the tower from the source to the destination.

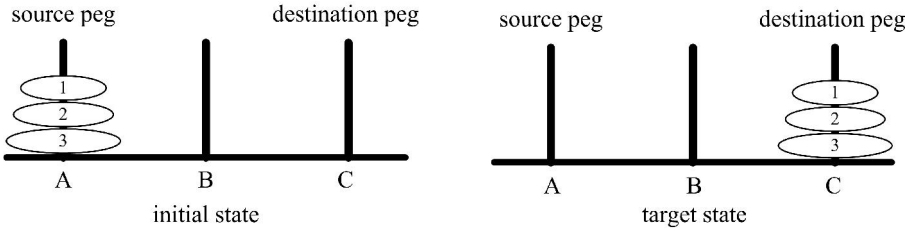

Fig. 1. The basic three-peg Tower of Hanoi Problem

2.2 Solution encoding

Let $M(n, p)$ denote the sequence of moves to solve the Tower of Hanoi problem with n ($\geq$3) disks and p ($\geq$3) pegs. And m_i denotes the ith move that transfer a disk from one peg to another. Then, a candidate plan for solving the Tower of Hanoi problem will be as follows:

$$M(n, p) = \{ m_0, m_1, m_2, \ldots, m_i, \ldots, m_{k-1} \}, \quad (1)$$

where k is the length of the planning.

Each single operation in the plan is encoded as a gene, and the sequence of moves of a candidate planning $M(n, p)$ is mapped as a combination of genes as the component of an individual, which the population is composed of.

In this paper, a direct way to represent an individual is used to encode each move by an integer, thus an individual is mapped as a string of integers. Take Fig.1 as an example. Let

1) 0 encodes the move from A to B
2) 1 encodes the move from B to A
3) 2 encodes the move from A to C
4) 3 encodes the move from C to A
5) 4 encodes the move from B to C
6) 5 encodes the move from C to B
7) 6 encodes No Action

The chromosome of a possible solution with a length of 7 for the problem can be as follows:

$$M(3, 3) = \{2, 0, 5, 2, 1, 4, 2\}.$$

This represents the sequence of operations that successfully transfer the three disks from A to C:

{A->C, A->B, C->B, A->C, B->A, B->C, A->C}

It is the simplest encoding way. However, the drawback is obvious: this encoding method do not guaranty all operations are valid in every possible state, and thus may result in an invalid solution that violates the three rules, even with the final goal state. For example, one possible sequence of moves could be as follows:

$$M(3, 3) = \{0, 0, 0, 4, 4, 4, 6\},$$

Which means a series of actions: {A->B, A->B, A->B, B->C, B->C, B->C, NA}.

The final result of these operations is correct. However, the repeat of moving disks from A to B introduces states with a larger disk reside on a smaller one, which breaks the 3rd rule. And thus, it is an illegal solution. So examination is necessary for solutions in the process of evaluation to ensure the validity of the final result.

Allowing the length of individual variable makes the evolutionary approach design more flexible but too complex. In this work, the length of individual is set with a specific fixed value for each evolution process.

2.3 Selection and mutation

Selection. The initial population is generated randomly in the beginning of evolution. After evaluation, an elitism selection is employed to keep the best individual for generating the next population. If there finds an individual scores the same best fitness as the current best individual, the current best one will be replaced by the individual with a probability of 50%.

Mutation. Suppose the population contains S individuals. In a traditional way, a new population is generated by performing mutation operation on the best individual selected in the previous generation S-1 times in the beginning of each generation. In this paper, a hierarchical mutation mechanism is employed to create the new population. A new population, except the best individual, is divided into q groups. Each part contains the same number, say r, of individuals. In other words,

$$S - 1 = \sum_{i=1}^{q} r_i \tag{2}$$

The r individuals in the first group are directly generated from the best individual by mutation. Each single mutation on the best individual creates a new individual. Every gene of the parent has equal probability of being mutated. In each mutation operation, a new integer number that encodes a possible operation is randomly chosen to replace the previous one - the old gene.

For the second group, every individual in this group is produced from the one and only one individual in the first group. Since the two groups are of the same size, the one-to-one method can create the new group easily and quickly. Similar operation is taken to generate the 3rd group, and so on, until the generation of the whole new population has completed, as Fig.2 shows.

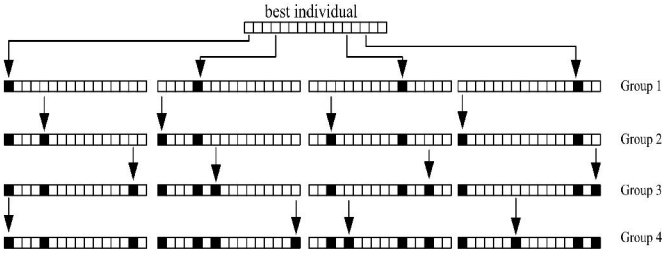

Fig. 2. Hierarchical mutation.

In the example of Fig.2, the size of the population is $S = 17$. The rest individuals, excluding the best one, are separated into $q = 4$ groups, each group contains $r = 4$ individuals. An arrow line shows the offspring is produced by whom and a black block indicates a mutated gene. Note that the gene to be mutated is randomly selected without any additional rule, so the existence of duplication of individuals is allowed.

The intension of this mutation mechanism and the selection of q and r are trying to find a path towards the balance between exploitation and exploration, while hoping there will be a positive impact on the overall execution time of this approach.

Evaluation. The goal of the proposed approach is to find a solution that satisfies the following two conditions: first, the sequence of operations leads the system from the initial state to the goal state; second, no invalid operation is allowed in the solution. The algorithm encourages individuals in the situation of less invalid moves in the sequence of operations, and later the first invalid move arises.

Therefore, the fitness function consists of three components: the score of the final state f_s, the number of invalid operations f_e, and the location of the first invalid move appears f_f.

The f_s evaluates the final state quality of a solution how well it fits to the goal state. The algorithm follows the operations from the beginning to the end of a solution to check its final state. The calculation of the score is related to the number of disks on the destination peg and the order of these disks. A better solution results in a higher fitness. The value of f_s depends on the number of pegs and disks.

While following the operations, the algorithm goes through every operation from the first move and checks if it is a valid operation. No matter what answer it is, the state is changed to what it will be, according to the behavior of the operation. If it is a valid operation, the values of f_e and ff remain unchanged. Otherwise, the value of f_e increases by 1. If it is the first invalid operation, the value of f_f is saved according to the location it appears, as follows:

$$f_f = max\_len - first\_error\_location \quad (3)$$

Where *max_len* is the length of the solution. And *first_error_location* is the location counted from the left side to the right side of the operation sequence. The later the first invalid operation appears, the less the f_f is. In the beginning of evaluation, the f_f is set to 0 initially. The fitness function is formulated as follows:

$$fitness = a \times f_s - b \times f_e - c \times f_f \quad (4)$$

Where a, b and c are weights. One can use them to emphasize one of the three factors by selecting adequate values, if it is necessary. In this paper, the three weights, a, b and c are all fixed as 1.

3 Evolutionary algorithm

3.1 Algorithm description

In this work, a modified CGP algorithm [10] is used as the evolutionary algorithm for searching the solution of the Tower of Hanoi problem. The basic procedure of the standard CGP algorithm is described as follows [11]:

1. Generate initial population
2. Evaluate fitness of genotypes in population
3. Promote fittest genotype to new population
4. Fill remaining places in the population with mutated versions of the parent
5. Return to step 2 until stopping criterion matched（reached the maximum generation number or found a valid solution）

Comparing with the conventional CGP, the modified version applies a hierarchical mutation mechanism, which enables the algorithm to search in a larger area with fewer amounts of genetic operations [12]. There is a slight difference in the evolutionary algorithm between this work and in [12]. The population size varies as the

complexity of the problem increase. And no taboo strategy is used further, as can be seen from Fig.2.

3.2 Building blocks

As tradition way goes, we choose a set of integer numbers as the building blocks for evolution. However, it is quite different from that of evolvable hardware. In evolvable hardware evolution, once the function set is selected, it remains unchanged no matter how complex the target circuit will be. Nevertheless, in the Tower of Hanoi problem the available type of operation grows as the member of pegs increases. The following equation describes the relation between the number of pegs and the number of available operations:

$$op = \mathrm{p}_m^2 = m \times (m-1) \tag{5}$$

Where m is the number of pegs, and op denotes the number of the available operations. Table 1 gives the building block set applied in this work.

For 3-peg Tower of Hanoi problem, the available building blocks in Table. I are those from No.1 to No.7. For 4-peg problem, the available range extends to No.13 and to No.21 for 5-peg problem. The table can be easily expanded as the peg number increases.

Table 1. Building block set

No.	Building block	Operation	No.	Building block	Operation
1	0	A->B	*12*	11	C->D
2	1	B->A	*13*	12	D->C
3	2	A->C	*14*	13	A->E
4	3	C->A	*15*	14	E->A
5	4	B->C	*16*	15	B->E
6	5	C->B	*17*	16	E->B
7	6	NA	*18*	17	C->E
8	7	A->D	*19*	18	E->C
9	8	D->A	*20*	19	D->E
10	9	B->D	*21*	20	E->D
11	10	D->B			

4 Experimental results

To evaluate the proposed approach, a series of tests on the Tower of Hanoi problem with 3-peg, 4-peg and 5-peg was performed. Each experiment ran multiple times and the results are reported here.

4.1 Three-peg Tower of Hanoi problem

The minimum number of moves to reach the goal state has been proved to be 2^n-1 [13], where n is the number of disks. In some of the tests, we set the size of individuals to the length which is slightly longer than 2^n-1. In others, we set the size to the length of the optimal solution, 2^n-1. We performed 50 runs in each case of disks (from 3 disks to 5 disks). Table 2 shows the experimental results.

For 6-disk problem, the length of an optimal solution increases to 63. Ten runs for 6-disk problem were performed. However, no solution was found within 6 hours in the tests.

4.2 Four-peg Tower of Hanoi problem

For the number of pegs greater than 3, things become difference. As for the case of 4-peg Tower of Hanoi problem, the size of possible states for an *n* disks problem increases to 4^n. The Frame-Stewart algorithm offers a way to find presumed-optimal solution [14]. It has not been proved to be optimal yet, but most assume that the solutions of Frame and Stewart are correct.

According to the Frame-Stewart algorithm, the length of the presumed optimal solutions for 4-peg of 3, 4, 5, 6, 7, 8 disks are 5, 9, 13, 17, 25, 33 [15], respectively. We used these presumed results to define the individual length in parts of the tests.

4.3 Five-peg Tower of Hanoi problem

For 5-peg Tower of Hanoi problem, situation is similar. The length of the presumed optimal solutions for 5-peg of 5, 6, 7, 8, 9 disks are 11, 15, 19, 23, 27 [15], respectively, according to the Frame-Stewart algorithm. We used these presumed results to define the individual length in parts of the tests.

For the 5-peg of 9 disks problem, the presumed optimal solution length is 27. We set the length of individual to 35, and tried 5 runs to search the solutions. Only one valid solution with a length of 32 was found, and the time cost was 2205 seconds.

Table 2. Experimental results of 3-peg problem

Disks	Runs	Length setting of individual	Successful ratio	Average execution time (s)	Optimal solution length[1]
3	50	12	100%	0.137	7
	50	7	100%	0.85	7
4	50	24	100%	21.6	17
	50	15	100%	22.8	15
5	50	40	100%	1842.4	31
	50	31	100%	2419.2	31

"Optimal solution length" is the length of the remained moves in the best solution after removing the "NA" operations.

Table 3. Experimental results of 4-peg problem

Disks	Runs	Length setting of individual	Successful ratio	Average execution time (s)	Optimal solution length
3	30	8	100%	0	5
	30	5	100%	0.49	5
4	20	16	100%	29.0	10
	20	9	100%	37.2	9
5	20	28	100%	195.9	20
	20	13	80%	245.1	13
6	10	20	100%	39.3	17
	5	17	80%	386.5	17
7	10	30	100%	513.4	28
	5	25	100%	815.0	25
8	10	40	100%	2020.3	35
	5	33	80%	5632.2	33

Table 4. Experimental results of 5-peg problem

Disks	Runs	Length setting of individual	Successful ratio	Average execution time (s)	Optimal solution length
5	10	16	100%	2.1	14
	10	11	100%	36.8	11
6	10	20	100%	19.3	17
	5	15	100%	177.4	15
7	10	19	100%	264.2	19
8	5	23	100%	3829.0	23
9	5	35	20%	-	32

4.4 Summary and discussion

The simple evolutionary method employed a fixed length strategy and applied only mutation as its evolutionary operator. The experimental results show that the proposed approach can evolve solutions for multiple pegs of the Tower of Hanoi problems. However, this method is not guaranteed to find a valid solution as the complex-

ity of the problem increases. And the execution time it costs in this situation is longer than other deterministic algorithms.

One possible way to improve the performance of our approach is to adopt certain heuristic method to guide the behavior of the evolutionary algorithm. For example, to trace the state of the disks on the top of each pegs [16] and guide the selection of reasonable evolutionary operation. Of course, it will introduce additional memory and computation time cost.

In the process of evolution, some candidates go into a state where most of the disks reside on the target peg in correct ordering, while the largest disk and the second largest disk are left on other pegs. Before putting the largest and the second largest disks on the target peg, all the disks on the target peg should have to be moved away firstly. This situation may cause it more difficult to reach the goal state. Modification of the evaluation should be done to let the fitness function to assign more score to the behavior that arranges the largest and the second largest disks.

Also can be seen from the experimental results, reasonable selection for the length of solutions will be a help to reduce the evolving time. However, the relationship between the length and the execution time is unclear yet.

5 Conclusion

In this paper, a non-deterministic approach for evolving solutions of the Tower of Hanoi problem is presented. The proposed method uses a simple and direct encoding, and thus candidates with invalid operations are allowed during evolution. The algorithm applies mutation only in the evolution process. The results of experiments show that our approach is capable of finding valid solutions for some cases. However, as the complexity of the problem increases, the approach experiences difficulties in finding solutions within a certain time. Possible suggestions on improving the search performance are discussed. We also set the individual length to a number smaller than the one given by the Frame-Stewart algorithm and try to exam whether the EA can find a legal solution. But so far it is not succeed. Future works will focus on these issues.

6 References

1. Hinz, Andreas M. "The tower of Hanoi." Enseign. Math 35.2 (1989): 289-321.
2. Er, M. C. "A representation approach to the tower of Hanoi problem." The Computer Journal 25.4 (1982): 442-447.
3. Hinz, Andreas M. "Shortest paths between regular states of the tower of Hanoi." Information sciences 63.1 (1992): 173-181.
4. Klavžar, Sandi, and Uroš Milutinović. "Graphs S (n, k) and a variant of the Tower of Hanoi problem." Czechoslovak Mathematical Journal 47.1 (1997): 95-104.
5. Hayes, P. J. "Discussion and correspondence A note on the Towers of Hanoi problem." The Computer Journal 20.3 (1977): 282-285.

6. Chi, Michelene TH, and Robert Glaser. Problem-solving ability. Learning Research and Development Center, University of Pittsburgh, 1985.
7. Spitz, Herman H., Nancy A. Webster, and Suzanne V. Borys. "Further studies of the Tower of Hanoi problem-solving performance of retarded young adults and nonretarded children." Developmental Psychology 18.6 (1982): 922.
8. Cambon, Stéphane, Fabien Gravot, and Rachid Alami. "A robot task planner that merges symbolic and geometric reasoning." ECAI. Vol. 16. 2004.
9. McDermott, Patricia L., Thomas F. Carolan, and Mark R. Gronowski. "Application of Worked Examples to Unmanned Vehicle Route Planning."The Interservice/Industry Training, Simulation & Education Conference (I/ITSEC). Vol. 2012. No. 1. National Training Systems Association, 2012.
10. Jie Li, and Shitan Huang. "Evolving in extended hamming distance space: hierarchical mutation strategy and local learning principle for EHW."Evolvable Systems: From Biology to Hardware. Springer Berlin Heidelberg, 2007. 368-378.
11. Miller, Julian F., and Peter Thomson. "Cartesian genetic programming."Genetic Programming. Springer Berlin Heidelberg, 2000. 121-132.
12. Jie Li, and Shitan Huang. "Adaptive salt-&-pepper noise removal: a function level evolution based approach." Adaptive Hardware and Systems, 2008. AHS'08. NASA/ESA Conference on. IEEE, 2008.
13. Romik, Dan. "Shortest paths in the Tower of Hanoi graph and finite automata." SIAM Journal on Discrete Mathematics 20.3 (2006): 610-622.
14. Klavžar, Sandi, Uroš Milutinović, and Ciril Petr. "On the Frame–Stewart algorithm for the multi-peg Tower of Hanoi problem." Discrete applied mathematics 120.1 (2002): 141-157.
15. Houston, Ben, and H. Masun. Explorations in 4-peg Tower of Hanoi. Technical Report TR-04-10, 2004.
16. Klavzar, Sandi, Uros Milutinovic, and Ciril Petr. "Combinatorics of topmost discs of multi-peg tower of Hanoi problem." Ars Combinatoria 59 (2001): 55-64.