Solving the Rubik's Cube using Simulated Annealing and Genetic Algorithm

Article *in* International Journal of Education and Management Engineering · January 2018

DOI: 10.5815/ijeme.2018.01.01

1 author:

Shahram Saeidi
Islamic Azad University Tabriz Branch
29 PUBLICATIONS **133** CITATIONS

I.J. Education and Management Engineering, **2017, 1, 1-12**
Published Online January 2013 in MECS (http://www.mecs-press.net)
DOI: 10.5815/ijeme.2013.01.01

Available online at http://www.mecs-press.net/ijeme

Solving the Rubik's Cube Using Simulated Annealing and Genetic Algorithm

Shahram Saeidi
Department of Industrial Engineering, Tabriz Branch
Islamic Azad University Tabriz, Iran
Sh_saeidi@iaut.ac.ir

Abstract

The Rubik's cube is 3D puzzle with 6 different colored faces. The classis puzzle is a 3x3x3 cube with 43 quintillion possible permutations having a complexity of NP-Hard. In this paper, new metaheuristic approaches based on Simulated Annealing (SA) and Genetic Algorithm (GA) are proposed for solving the cube. The proposed algorithms are simulated in Matlab software and tested for 100 random test cases. The simulation results show that the GA approach is more effective in finding shorter sequence of movements than SA, but the convergence speed and computation time of the SA method is considerably less than GA. Besides, the simulation of GA confirms the claim that the cube can be solved with maximum 22 numbers of movements.

Index Terms: Rubik's Cube, Simulated Annealing, Genetic Algorithm

1. Introduction

The classic Rubik's cube is mechanical 3D puzzle which is in-vented by a Hungarian sculpture and professor Erno Rubik in 1974. It was first knows as the Magic Cube and later is re-named to the Rubik's Cube to honor of its inventor. The standard cube has 6 faces each with 9 (3×3) facelets. The goal of solving the cube is having each face arranged with a specific color. The total number of permutations is about 43 quintillion (exact value is 43,252,003,274,489,856,000 which is known as God's number) turns that makes it as a NP-Hard combinatorial problem. If each turn takes one second of time, it will need about 1400 trillion years to perform all possible permutations. El-Sourani et al. imply an upper bound for the maximum number of needed movements to solve the puzzle which is equal to 22 movements. Some algorithms are proposed to decrease this upper bound [1].

Solving the cube is always considered in two aspects of time and the number of the movements which both are covered in this paper. El-Sourani et al. in [1] claim that there is an upper bound for number of movements needed for solving the cube. This bound was first reported as 27 which was next decreased to 22 movements. It should be mentioned that these values are obtained by empirical experiments without using a scientific or mathematical calculation. The proposed GA approach in this paper is designed for testing this hypothesis and

the simulations results confirm the claim. The effort for finding smaller upper bounds still continues by enthusiasts.

In this paper, two metaheuristic approaches based on GA and SA are developed for solving the Rubik's cube. The proposed algorithms are implemented in Matlab 2009a software and are compared using equal random inputs. The computational results show that the GA based approach lead to solutions with fewer numbers of movements than that of SA, although the computation time for finding such solutions is higher than SA. Considering the time aspect, the SA based approach solves the cube faster than GA approach, but offers more number of movements.

There are some known classic algorithms for solving the cube which mostly complete the cube layer by layer. These algorithms are hard to learn and remember the sequence of the movements, especially for a large size N×N×N (extended versions) of the Rubik's cube. Therefore, proposing heuristic approaches has raised the interest of the researchers and mathematicians which unfortunately few numbers of them are academically reported and published. For example Korf in [2] used Pattern Databases, Lee & Miller [3] used Best-Fast method, Kapadia et al. [4] used A* Algorithm and IDA* which is depth-first search oriented method as their graduate project, El-Sourani et al. [1] composed exact methods with an evolutionary approach for solving the Rubik's cube problem, as their graduate project. El-Sourani and Borschbach in [7, 8] proposed two evolutionary algorithms based on group theory and compared the efficiency of the developed methods. Some other heuristics are developed which none of them are published as an academic research work. Anil in [9] proposed a method for solving the cube using Genetic Algorithm which is based on Group theory analysis. This algorithm solves the cube suing more than 100 movements and is not fast enough regarding to the proposed GA algorithm in this paper. Mantere proposed a method based on the cultural algorithm for solving the cube [10]. Smith et al. [11] developed evolutionary approach using genetic programming.

2. The Proposed Approaches

In this section, first the Rubik's Cube and related operators are mathematically modeled. Next, the proposed Simulated Annealing and Genetic Algorithm approaches are described in details.

2.1. Modeling the Problem

In order to simplify the implementation and calculations of the operators on the problem, the 3D mechanical Rubik puzzle is mapped as a 2D equivalent numerical matrix as shown in Fig. 1.

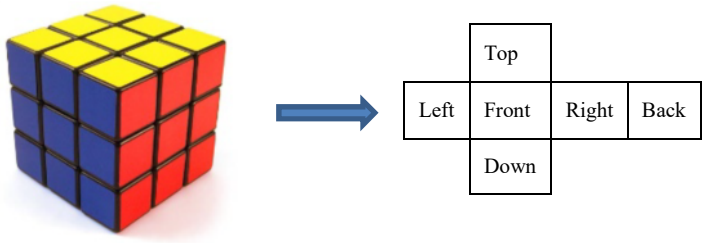


Fig. 1 2D mapping of the Cube

The Cube has six colored faces, each is made up 9 (3×3) small pieces. Each color is considered as an integer number in [1..6] range. Next, The top and down(bottom) faces of the cube in 2D representation, are moved to first and end positions of the middle row of the matrix, to have a homogenous array of integers, leading the 3x18 array named M_{ij} shown in Fig. 2.

Top			Left			Front			Right			Back			Down		
1	1	1	2	2	2	3	3	3	4	4	4	5	5	5	6	6	6
1	1	1	2	2	2	3	3	3	4	4	4	5	5	5	6	6	6
1	1	1	2	2	2	3	3	3	4	4	4	5	5	5	6	6	6

Fig. 2 Integer array representing the solved cube

2.2. Modeling the Operators

There are eighteen basic operators which can be applied on the Rubik's cube. These operators rotate one layer out of 9 layers of the cube (three layers at each dimension), clockwise or counter clockwise. Applying each of these operators moves some of the small colored pieces of the cube to new position which means some integer values in 2D representation will exchange location with some others. For example, rotating the front face of the cube in clockwise direction, exchanges the array cells as the sequences shown in Fig.3 in which $M(i,j)$ represents the ith row and jth column of the M_{ij} matrix.

$$
\begin{aligned}
&M(3,6) \rightarrow M(3,1) \rightarrow M(1,10) \rightarrow M(1,18) \rightarrow M(3,6)\\
&M(2,6) \rightarrow M(3,2) \rightarrow M(2,10) \rightarrow M(1,17) \rightarrow M(2,6)\\
&M(1,6) \rightarrow M(3,3) \rightarrow M(3,10) \rightarrow M(1,16) \rightarrow M(1,6)\\
&M(3,7) \rightarrow M(1,7) \rightarrow M(1,9) \rightarrow M(3,9) \rightarrow M(3,7)\\
&M(2,7) \rightarrow M(1,8) \rightarrow M(2,9) \rightarrow M(3,8) \rightarrow M(2,7)
\end{aligned}
$$

Fig. 3 Exchanging the array cells by rotating the front face clockwise

The related exchanges of all eighteen basic operators are calculated similarly. The input of the proposed algorithms is an messed up 3x18 matrix (demonstrating a shuffled 3D cube) and the main objective is sorting the matrix as Fig. 2 using a minimum length sequence of the operator (in GA approach) or at shortest processing time (in SA approach).

2.3. The SA Approach

The Simulated Annealing approach is a random search methods adopted from metallurgy introduced by Metropolis et al. [5] and Kirkpatrick [6]. The algorithm first heats up the heat bath (system) to melt the solid and next decreases carefully the temperature of the system until the particles arrange themselves specific in the ground state of the solid.

The SA algorithm starts with a randomly generated feasible solution for the problem (call x) and calculates its energy (fitness), f(x). Then a subsequent neighbor (state) y is generated by applying small changes on the current solution. If the energy (fitness) difference f(x)-f(y) is less than or equal to zero, the state y is accepted as the current state. Otherwise, the state y is accepted with the probability given by the equation 1.

$$\text{Probability} = e^{\frac{f(x)-f(y)}{K_B \cdot T}} \tag{1}$$

Where T is temperature of the system, and KB is physical constant called Boltzmann constant [7]. Considering the structure of the Rubik's cube, each of the eighteen operators mentioned in previous section, makes some changes on the current cube and generate a new cube, which can be considered as a neighbor of the current solution.

The fitness function in the proposed algorithm is defined as total number of small colored pieces of all the faces, which are not located at their correct position. Thus, the problem is formulated as a minimization combinatorial problem in which, the fitness of the final solution (solved cube) is equal to zero. By defining binary matrix U_{ij} [3×18] as:

$$U(i,j) = \begin{cases} 1 & if\ M(i,j) = \left\lceil \frac{j}{3} \right\rceil \\ 0 & therwise \end{cases} \qquad i=1,2,3;\ J=1,2,\ldots,18 \tag{2}$$

So, the proposed fitness function for a given cube (or equivalent M_{ij}) is defined as equation 3.

$$f(cube)= \sum_{i=1}^{3}\sum_{j=1}^{18} U(i,j) \tag{3}$$

The SA algorithm gets a randomly messed up Rubik's cube as initial solution, and calculates its fitness. At the worst case, the fitness will be 54-6=48. This is because the cube has 54 small pieces and the middle square of each face, is always located at the correct position and will never change. Next, the main loop starts whit the initial temperature is set to 5000 and continues until the best solution is found, or the temperature is almost reached to zero. The percentage value of decreasing the temperature is considered as α=0.05. At each temperature, the inner loop randomly generates the neighbors of the current solution by applying one of the eighteen operators and calculates the fitness of the new solution. The pseudo code of the proposed SA approach is given in Fig. 4.

1. *Begin*
2. *Initialize Temp;* initial probability function which should be around 1. and tend to zero
3. *Get input cube, called x;*
4. *Compute f(x) using equation 3;* cost function evaluated
5. *Repeat until frozen*
   a. *Do N times* 1000
      i. *y := FindNeighbor(x);*
      ii. *Delta := f(y)- f(x);*
      iii. *if Delta≤0 then x:=y ; (Accept y)*
      iv. *else if U(0,1) < $e^{-(Delta)/Temp}$ then x:=y; (Accept y)*
      v. *else reject y;*
   b. *End Do* delta ?
6. *Temp := Temp ×θ;* cooling rate=0.05 no more like .99 or .95
7. *The Solution will be best so far;*
8. *End*

Fig. 4 The pesudo code of the proposed SA

Inside the inner loop, we move from the current solution *x* to its neighbor solution *y* and calculate the fitness. The neighbor solution is a point in feasible solution space which is obtained by making small changes in the current point (solution). Considering the Rubik's cube structure, these changes are performed by applying either of eighteen operators on the current cube. If the fitness of the new solution is better (less) than that of the current solution, it will be accepted and set as the current solution. Otherwise the solution is not ignored forever and it may be accepted with probability calculated by equation 1 in order to prevent the local optimum trap.

2.4. The GA Approach

The GA method is a population based approach and is made up some chromosomes each of which representing a solution of the problem being solved. Generally, the first generation of the chromosomes is created by random and during the execution iterations, next generations are made by applying crossover and mutation operators. It is very important to define and design the structure of the chromosomes in order to optimize the efficiency of the GA method. The structure of the chromosomes in proposed GA is considered as an integer vector of length k which is shown in Fig. 5.

1	2	...	k-1	k
Op#	*Op#*	...	*Op#*	*Op#*

Fig.5 The structure of the proposed chromosome

The values of the genes in the chromosome represent the operation number (Op#) which is suggested to be applied on the cube. So, a chromosome demonstrates a sequence of operators for solving the cube if *k* is large enough. In proposed GA method, the length of the chromosome first was considered as k=50 and all input cubes were easily solved using maximum 50 numbers of movements. Next experiments were designed by decreasing the length of the chromosome and the chance of solving the cube was also decreasing as for k=22 (El-Sourani claim) the proposed approached was 43% successful in solving the cube. It should be mentioned that the proposed GA was unsuccessful in solving the cube for k<22 which informally confirms the claim.

The success rate of the proposed GA for solving the cube using sequence of movements of length *k* is given in table 1 for different values of *k*.

Table 1. Success rate of proposed GA with regard to the length of the chromosome

Value of *k*	*K*≥50	40	30	25	22	*K*<22
Success Rate	100%	91%	73%	44%	38%	0

The population size at every generation is considered as 100 chromosomes. The simulation results show that increasing the population size, increases the computation time of the execution, but does not lead to considerable improvement at successful rate of solutions. The chromosomes at the first generation are created randomly and the value of each gene is randomly selected between [1..18] to identify the movements. It is also considered that the value of two consecutive genes should not be set to opposite movements because it neutralizes the operator's influence. After generating the chromosomes, the fitness of them is calculated using equation 3.

The chromosomes of the current population are selected using roulette wheel method with probability α=80% and offsprings (children) are generated using one point crossover operator. If the fitness of the children is better than that of the parents, they will be moved to the next generation. The mutation is another operator which is applied on β=20% of the population and changes the value of a randomly selected gene such that the new value is not the opposite movement of the previous and next gene.

The elitism concept is also considered in proposed GA that transfers top 10% chromosomes of the current generation to the next generation. The pseudo code of the proposed GA is given in Fig. 5.

1. *Begin*
2. *Get initial input cube.*
3. *Generate initial population of size PopSize.*
4. *Do while termination condition is not met*
    a. *Calculate the fitness of chromosomes in current population.*
    b. *Select PopSize/2 pair of population using roulette wheel method.*
    c. *Apply Crossover operator with probability α. Calculate children's fitness, if acceptable, add them to mate pool.*
    d. *Apply mutation operator with probability β. If the fitness is modified, add the mutated chromosome to mate pool.*
    e. *Add top 10% of the current population to mate pool.*
    f. *Sort the mate pool in decreasing order, truncate the mate pool and replace as the current population, empty the mate pool.*
5. *Select the topmost chromosome of the population as the final solution and report the sequence of operators.*
6. *End.*

Fig 5. The pseudo code of the proposed GA

3. Simulation and Computational Results

The proposed methods are implemented in Matlab 2009a software using a personal computer with 4.2 GHz of processor and 2GB of RAM, running Microsoft Windows 10 and the simulation results are reported in the following subsections.

3.1. The Proposed SA results

The initial temperature value should be chosen as the value of the equation 1 is calculated around 1 [6]. Hence, considering the difference domain between the fitness of two neighbor solutions, this value is set to 1000. The cooling rate of the temperature is set to $\theta=0.05$ by try and error.

The proposed SA is executed for200 different initial cubes and it was successful solving the cube on all the test cases. The number of needed movements and the computation time of the best and the worst cases in seconds are given in table 2.

Table 2. Computation results of proposed SA for 200 experiments

Criteria \ Case	Best Case	Worst Case	Average
Computation time (seconds)	18.12	68.47	36.31
# of movements	157	358	219

The convergence diagram of the best and the worst cases related to table 1 are also depicted in figures 6 and 7 respectively. The vertical axis shows the value of fitness function.

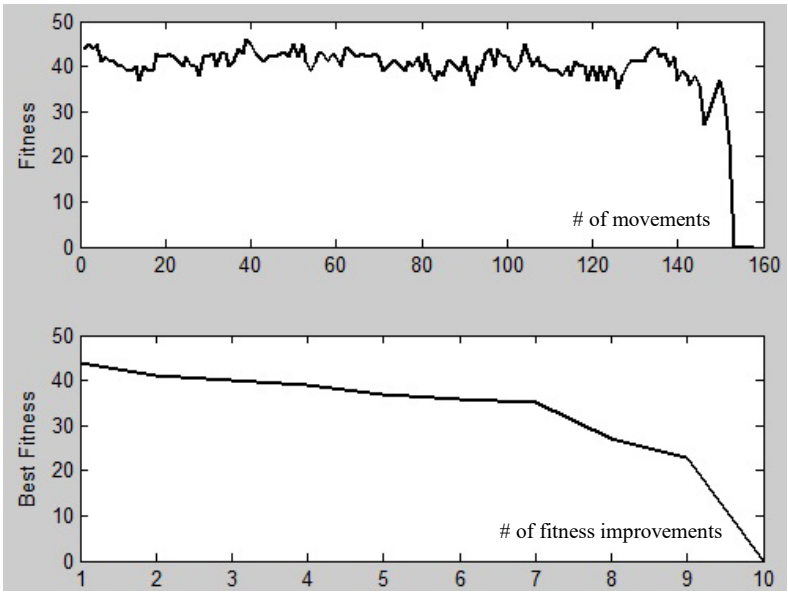


Fig. 6 Total fitness (a) and best fitness (b) of the best case

The main reason of fluctuations at (a) diagram in figures 7 and 8 is because of the nature of the movement operators. Applying an operator on the cube, may move a facelet to its correct position, but change the correct

position of some other pieces at the same time. Besides, one movement can move some pieces (maximum 9 pieces) to their correct position and lead to salient decrement of the fitness.

could this be an argument for my proposed column / row check

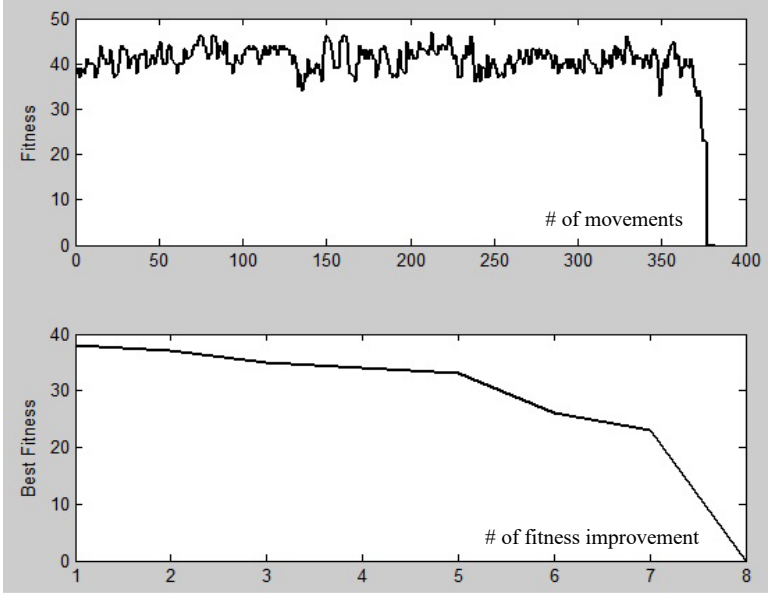


Fig. 7 Total fitness (a) and best fitness (b) of the worst case

3.2. The Proposed GA Results

In order to compare the proposed algorithms, the proposed GA is also simulated for the same experiments test cases. The simulation parameters are listed in table 3. The results show that the GA approach can lead to shorter solution considering number of the movements, although the execution time for finding such solutions as high.

Table 3. Simulation Parameters of Proposed GA

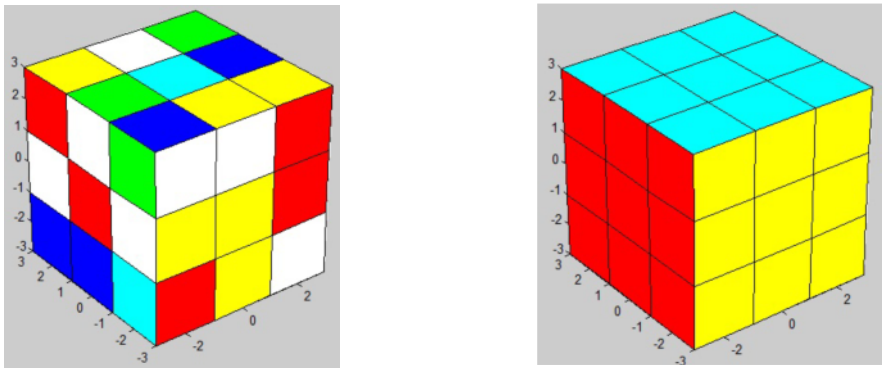
Parameter	Value
Population Size	100
Crossover Rate	0.8
Mutation Rate	0.2
Chromosome Size	22 up to 50
Number of experiments for each chromosome size	200
Number of iterations until final solution	2562 Generations for 22 movements 37 Generations for 50 movements

The simulation results of the best and the worst cases with regard to the number of the required movements for solving the cube are given in table 4.

Table 4. Number of required movements and mean execution time of 200 experiments

Criteria \ Case	Best Case	Worst Case
Required number of movements	22	50
Average Execution time (Second)	3135.12	68.25

As shown in table 4, the GA approach has taken a long time (3135.12 seconds) for finding short solutions of length 22. It should be mentioned that the GA method was able to find such solutions only in 43% cases (86 out of 200 experiments) and the reported time is the average of these successful cases. The figure 9 shows the input and output of the algorithm for one of these starting from the fitness equal to 38.



(a) (b)

Fig.9 (a) Input cube, (b) Solved cube using 22 movements

The corresponding convergence diagram is depicted in Fig. 10 in which the horizontal axes shows number of the movements. The vertical axes shows the value of fitness function which has reached to zero at the 22nd movement.

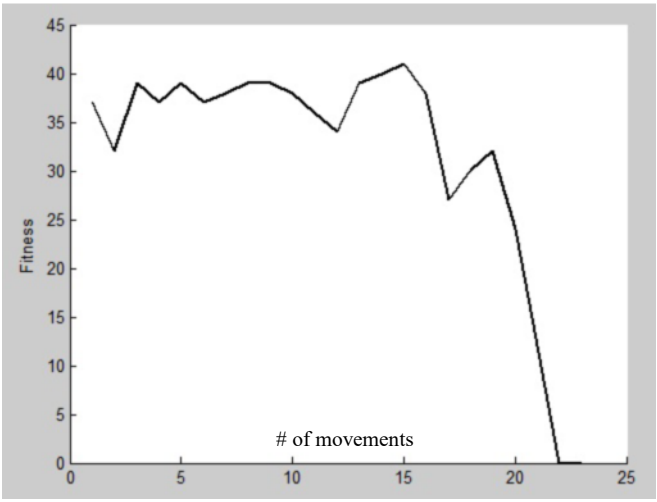


Fig. 10 The fitness function of solving the cube using 22 movements

4. Conclusions

In this paper, the Rubik's cube problem is considered for solving and due to the complexity and NP-Hard nature of the problem, two meta-heuristic algorithms based on Simulated Annealing and Genetic Algorithm are proposed for solving the cube. The proposed algorithms are simulated in MATLAB software using a personal computer with 4.2 GHz processor and 2GB of RAM running Microsoft Windows 10 for 200 different experiments.

The Computational results show that the SA approach performs faster than the GA method for solving the cube and consists less iterations and rapid convergence. Nevertheless the GA approach is more successful in finding short solutions, that is, the number of required movements for solving the cube is less. Besides, the several experiments of the GA method, confirms the claim that the cube requires at most 22 numbers of movements to be solved.

References

[1] El-Sourani, N., Hauke, S., Borschbach, M. An evolutionary approach for solving the Rubik's cube incorporating exact methods. *EvoApplications*, Part I, LNCS 6024, 2010, 80-89.

[2] Korf, R.E. Finding optimal solutions for Rubik's cube using pattern databases, *American Association for Artificial Intelligence*. (www.aaai.org), 1997.

[3] Lee, D., Miller, W. A prolog program and demonstration of an efficient heuristic search method, *Ziff-Davis magazine*, 2(**4**), 1997.

[4] Kapadia, M., Deshpande, M.V., Umale, J. Rubik's heuristic, *Mumbai, India, Electro Info-Com*, 2007.

[5] Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P. *Optimization by Simulated Annealing*, Science 220, 1983.

[6] Aarts, E.H.L., Korst, H.H.M. Simulated Annealing and Boltzmann Machines, *Wiley*, Chichester, 1989.

[7] El-Sourani, N., Hauke, S., Borschbach, M. Design and Comparison of two Evolutionary Approaches for Solving the Rubik's Cube, Parallel Problem Solving from Nature, PPSN XI, 2010, 442-451.

[8] El-Sourani, N. Design and Benchmark of different Evolutionary Approaches to Solve the Rubik's Cube as a Discrete Optimization Problem. Diploma Thesis, WWU Muenster, Germany, 2009.

[9] Anil, K. S., Solution for Rubik's Cube by Using Genetic Algorithm, *International Journal of Engineering Sciences & Research Technology* , 2015, 636-641.

[10] Mantere, T., Solving Rubik's cube with cultural algorithm, unpublished, 2012.

[11] Smith, R. J., Kelly, S., Heywood, M. I., Discovering Rubik's Cube Subgroups using Co-evolutionary GP: A Five Twist Experiment, GECCO '16 Proceedings of the Genetic and Evolutionary Computation Conference, 2016, 789-796.

Author(s) Profile

Shahram Saeidi (born April, 1973), received his PhD in Industrial, in 2009 from Mazandaran University of Science and Technology,. He received his BSc in 1996 in Computer Engineering from Shahid Beheshti University. He is currently an Assistant Professor at Industrial Engineering department in Islamic Azad University, Tabriz branch. His research interest includes optimization, scheduling, operation management and production systems.