Solving nonograms using genetic algorithms

Alicja Bobko, Tomasz Grzywacz
Warsaw University of Technology
Faculty of Electrical Engineering
Warsaw, Poland
Email: tomasz.grzywacz@ee.pw.edu.pl

***Abstract*—This paper presents the possibilities of using classical genetic algorithm as a tool for solving logical images. Results are presented for simple objective function taking into account the quantity of 0's and 1's and the modified objective function basing on the differences in length of each filled block. Brute-force algorithm was used to illustrate the complexity of NP-complete problem.**

I. Introduction

Japanese nonogram is a picture logic puzzle which takes the form of $N \times M$ grid. Solving it consists of filling in appropriate cells of the grid in order to reveal the hidden picture. At the ends of each row and column there are clues which indicate the number of consecutive black squares. Small puzzles can

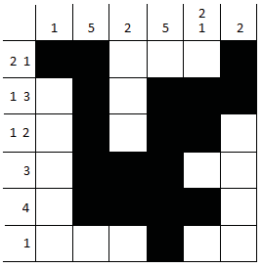

Fig. 1. Nonogram example

usually be solved by hand using simple logic rules, but when considering larger ones, it becomes more difficult since it is a NP-complete combinatorial optimization problem [3]. This paper presents the performance of Genetic Algorithm with binary encoding and different fitness functions as a stochastic computational approach to solve this problem.

II. The classic genetic algorithm

Genetic Algorithm (GA) is an optimization technique used mostly for large number of applications in optimization, control or signal processing. Operators in GA were inspired by mechanisms of natural selection and inheritance. They use the evolutionary principle of survival of the fittest individuals.

Fitness function is a measure of the adjustment of an individual in the population, in other words, it determines the chance of survival. Suitable formulation of the evaluation mechanism has a significant influence on the algorithm.

Overview of the algorithm:

1) Initiation random selection of initial population of chromosomes in the form of binary strings.
2) Evaluation of the adaptation of chromosomes in the population.
3) Selection of chromosomes.
4) Application of genetic operators: crossover, mutation.
5) Creation of a new population final population becomes the current one, then it returns to the second point of the algorithm.
6) Verification of the stop condition.
7) Selection of the best chromosome (the highest fitness function).

III. Simple fitness function

The objective function compares the resulting amount of 0's and 1's of each line (row or column of a nonogram) of the matrix X with the target solution, which is given as the conditions r_{ij} and c_{ij} (where $r_{ij}, i \in N, \forall j$, and $c_{ij}, k \in M, \forall j$ are the conditions respectively for rows and columns).

For example, in figure 1.: $r_{11} = 2$, $r_{12} = 1$, $r_{13} = 0$ etc., and $c_{11} = 1$, $c_{12} = 0$ etc. For such assumptions desired objective function takes the form [1]

$$f(X) = f_1(X) + f_2(X) + f_3(X) + f_4(X) \tag{1}$$

$$f_1(X) = \sum_{i=1}^{N} |\sum_{p=1}^{M} r_{ip} - \sum_{p=1}^{M} x_{ip}| \tag{2}$$

$$f_2(X) = \sum_{k=1}^{M} |\sum_{p=1}^{N} c_{kp} - \sum_{p=1}^{N} x_{kp}| \tag{3}$$

$$f_3(X) = \sum_{i=1}^{N} |(M - \sum_{p=1}^{M} r_{ip}) - (M - \sum_{p=1}^{M} x_{ip})| \tag{4}$$

$$f_4(X) = \sum_{k=1}^{M} |(N - \sum_{p=1}^{N} c_{kp}) - (N - \sum_{p=1}^{N} x_{ip})| \tag{5}$$

Function $f_1(X)$ determines the difference between the expected number and the current number of 1's in matrix X, $f_2(X)$ does the same for the columns. Further equations: $f_3(X)$ and $f_4(X)$ give the difference between the number of 0's for rows and columns. The solution represents binary matrix X minimizing function $f(X)$. The objective function has been transformed into a maximization problem:

$$g(X) = N \cdot M - f(X) \tag{6}$$

Results analysis:

978-1-5090-2800-9/16/$31.00

a) nonogram 4x4:

Constraints:
r = [0 1; 0 3;0 3;1 1]
c = [0 1;0 3;0 2; 0 3]

Parameters:
Population size: N = 30
Chromosome length: L = 16
Probability of mutation: 0.01

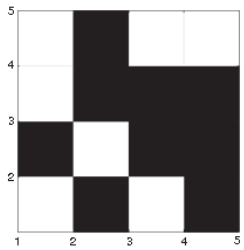

Fig. 2. Solution for maximized objective function for nonogram 4x4

Average time to obtain the correct solution was 0.226 sec. with 100% effectiveness for 300 iterations. As can be seen in figure 3 adaptation value quickly converges to the maximum.

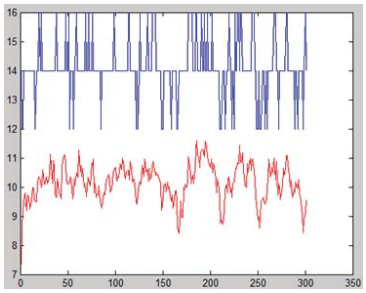

Fig. 3. Fitness function chart, Y-axis - value of fitness function (maximum - blue, average - red), X-axis - number of iterations

b) nonogram 5x5:

Constraints:
r = [2 0 0;3 0 0;3 1 0;1 1 0; 1 1 0]
c = [1 0 0;3 0 0;2 1 0;2 1 0; 1 1 1]
Parameters:
Population size: N = 40
Chromosome length: L = 25
Probability of mutation: 0.01

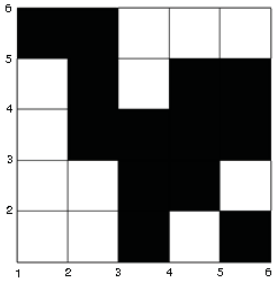

Fig. 4. Solution for maximized objective function for nonogram 5x5

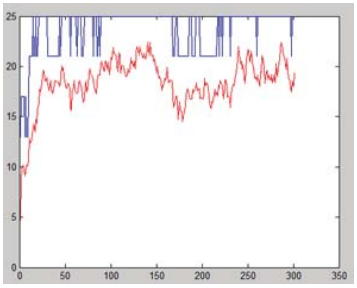

Fig. 5. Fitness function chart for nonogram 5x5

Average execution time: 0.426 sec. with 100% efficiency for 300 iterations and 30 runs of algorithm.

c) nonogram 10x10

Constraints:
r = [0 2; 1 1; 0 4; 2 1; 3 1; 0 8; 0 8; 0 8; 0 7; 0 5; 0 3]
c = [0 1; 0 2; 1 6; 0 9; 0 6; 0 5; 0 5; 0 4; 0 3; 0 4]
Parameters:
Population size: N = 90
Chromosome length: L = 100
Probability of mutation: 0.003

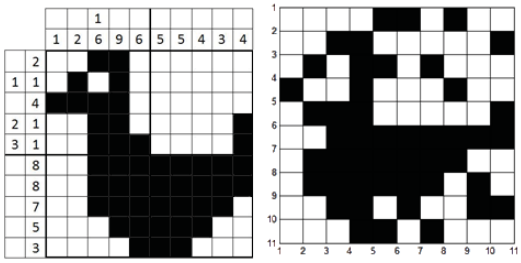

Fig. 6. The correct solution of nonogram 10x10 and solution for the highest fitness value equal to 94

The average execution time: 38.294 sec. with 100% effectiveness for 3000 iterations and 30 executions of algorithm and the average value of maximum adaptability for 30 runs equal to 90.842.

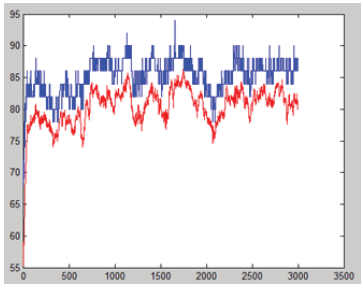

Fig. 7. Fitness function chart for run with the best results

Algorithm with a simple fitness function for larger sizes of nonograms do not converge to the maximum but remain at a relatively high level. The images shown in figure 6 are the results of maximized objective function and are not the correct solutions even though the objective function reached the expected value. This happens because the simple objective function defines only the numerical content of 0's and 1's in the individual rows and columns, not taking into account their continuity.

IV. Modified fitness function

The modified function gives results in the form of the solution taking into account the continuity of the blocks [2]

$$f(X) = f_1(X) + f_2(X) \tag{7}$$

$$f_1(X) = \sum_{i=1}^{M} \sum_{j=1}^{Q} |c_{ij} - x_{ij}| \quad (8)$$

$$f_2(X) = \sum_{i=1}^{N} \sum_{j=1}^{Q} |r_{ij} - x_{ij}| \quad (9)$$

The function returns the absolute value of the difference between the expected and received blocks.

Fig. 8. Expected form of the row and below the one received for analysis

For the situation shown in figure 8:

$$f_2(X) = |2-1| + |3-2| + |1-2| = 3 \quad (10)$$

As previously, the objective function has been converted to the problem of maximizing the function $g(X)$:

$$g(X) = 2 \cdot N \cdot M - f(X) \quad (11)$$

Results analysis:
Tests were performed with the same parameters as in the simple objective function.

a) nonogram 4x4

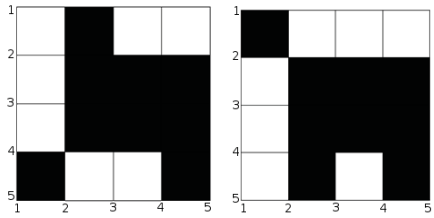

Fig. 9. Correct solutions obtained after application of modified fitness function

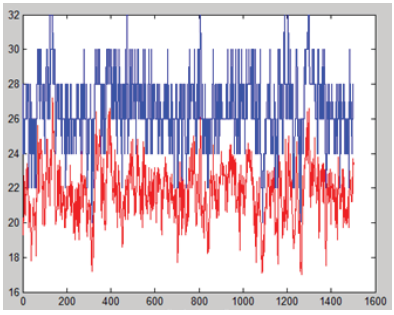

Fig. 10. Fitness function chart for run with the highest results for nonogram 4x4

The correct result was obtained for 100% of the runs. Modified objective function allows to receive accurate solutions, as in the brute-force method. An average time to reach the maximum value by function 11 was 2.978 sec. It is doubled because the differences between the length of blocks and the constraints are checked twice for the entire matrix (for all rows and columns). An extreme case is a matrix of 1s compared with a matrix of 0s, then the objective function reaches the minimum – zero.

b) nonogram 5x5

For 30 runs of algorithm the maximal adjustment, equal to 50 was received only once. The correct solution is shown in figure 3.4. The exact moment when the function reached this value can be observed at the fitness function chart (fig. 12). At this point a mutation may have occurred that allowed to come out of the local maximum.

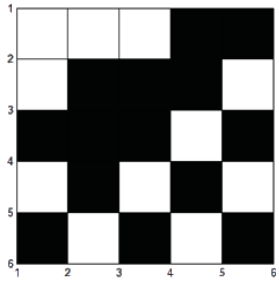

Fig. 11. The best results for nonogram 5x5

Values of the adjustment on the roulette wheel were averaged so individuals did not introduce any significant changes in the objective function. The search space has been extended

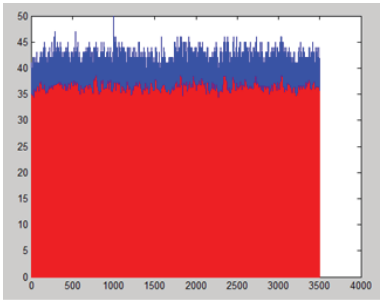

Fig. 12. Fitness function chart for preceding nonogram

to 3500 iterations and the probability of mutation increased to 0.05 because such parameters achieved the best results. Average execution time for that number of iterations was 30.027 sec. and average maximum adjustment was 46.67.

c) nonogram 10x10

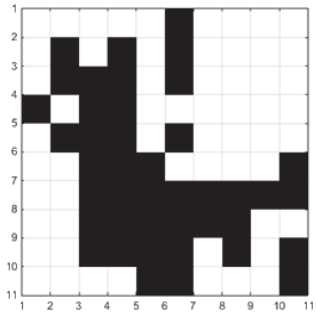

Fig. 13. Solution with the objective function reaching 160 for the nonogram size of 10x10

Average execution time was 156.82 sec. and average maximum adjustment was 160.2. The modified objective function did not cope with the problem-solving of nonogram of larger size, due to the computational complexity of the problem.

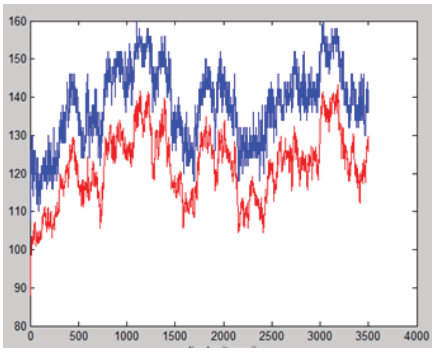

Fig. 14. Fitness function chart for nonogram in figure 13

V. Veryfying the continuity of rows and columns

The verifying function checks the continuity of filled cells in each row (or column) and compares it with the constraints i.e. the expected value. This method is based on finding numbers of indices of empty cells in analysed row (column) of matrix. For the found indices, the previous index (n) value incremented by one must be substracted from the next (n+1) to obtain a vector, for example, figure 15:
$[2-(1+1), 5-(2+1), 6-(5+1), 8-(6+1), 11-(8+1)] = [0, 2, 0, 1, 2]$

Before comparing the resulted vector with the constraints vector all of the 0's should be removed. It is only possible to compare vectors of the same length, so the next step consists of converting them into the form allowing the comparison. To align both vectors, corresponding number of 0's is added at the end of the shorter one.

Fig. 15. Sample column of nonogram along with the indices of empty cells

The objective function reaches zero if all of the rows and columns are constructed correctly (vectors are equal).

VI. Illustrating the scale of the problem

Since solving nonogram is a NP-complete combinatorial optimization problem, finding all of the solutions is only possible by checking the whole search-space. A traditional simple brute-force method has been applicated to illustrate how it will cope with the problem. There is no standard algorithm as each problem is different. The general idea is based on analysis of all of the states in an orderly manner. If the algorithm is well constructed each of the states is subjected to analysis exactly once. Implementation of brute-force is simple and finding t he s olution i s g uaranteed. T his method is usually used when the size of the problem is small, when there exists a possible heuristic approach to reduce the set of solutions to a reasonable size or when simplicity is more important than the speed of the algorithm. The main idea is to convert the following decimal numbers within the range of all combinations (for nonogram size 4x4: from 0 to 65 535) into a binary vector in which the length is the number of cells in the matrix. The task was conducted only for 4x4 and 5x5 dimensions, since the method requires large computational memory and time of the execution grows exponentially.

TABLE I Results of brute-force method

Size	No. of combinations	Average execution time
4x4	65535 (2^{16})	187.47 sec.
5x5	35554432 (2^{25})	1012 min $\approx$ 16.9 h

Brute-force method is inefficient even for nonograms sized 5x5. Despite the 100% effectiveness the search-space is enormous and grows exponentially. Solving this puzzle manually takes incomparably less time which is contrary to the objectives of the experiment.

VII. Conclusion

Only for nonograms sizes 4x4 and 5x5 with simple fitness function author's implementation of the genetic algorithm coped with optimization in shorter time. In other cases Matlab optimization tool gave better results in terms of time and quality of the solution. Although optimization problem with well-defined objective function does not seem to be a complicated issue itself it should be remembered that standard technical problems appear to have much smaller number of variables. For nonograms 4x4 and 5x5 with chromosome length 16 and 25, respectively, algorithm gave the correct answer, but in the case of 10x10 with chromosome length of 100 none of the methods did give expected result.

References

[1] Sancho Salcedo-Sanz, Jose A. Portilla-Figueras, Emilio G. Ortz-Garca, ngel M. Prez-Bellido, Xin Yao, *Teaching Advanced Features of Evolutionary Algorithms Using Japanese Puzzles*, IEEE Transactions on Education, Vol. 50, no. 2, 2007.

[2] Tsai Jinn-Tsong, *Solving Japanese nonograms by Taguchi-based genetic algorithm*, 2012.

[3] Ueda Nobuhisa, Nagao Tadaaki, *NP-completeness results for NONOGRAM via Parsimonious Reductions*, Technical Report, Department of Computer Science, Tokyo Institute of Technology, 1996.