Solving Nonograms with Genetic Algorithms and Wisdom of Artificial Crowds

James Culver
Department of Computer Engineering and Computer Science
Speed School of Engineering
University of Louisville, USA
jeculv02@louisville.edu

Mark Ford
Department of Computer Engineering and Computer Science
Speed School of Engineering
University of Louisville, USA
maford15@louisville.edu

Oleg Poyan
Department of Computer Engineering and Computer Science
Speed School of Engineering
University of Louisville, USA
o0poya01@louisville.edu

***Abstract*— Nonograms also known as picross and griddlers is a popular Japanese logic puzzle. An approach is introduced using a genetic algorithm and wisdom of artificial crowds to solve nonograms. Solving nonograms is an NP-complete problem. Genetic algorithms have a tendency to get stuck on local optimums and take many generations to improve further. The wisdom of crowds approach takes the best candidates from multiple runs to create a crowd of experts. These experts are used to aggregate a new solution. This solution often improves beyond the best expert in the crowd. The method proposed was able to successfully solve 10x10 simple nonogram puzzle, and also attempted to solve more difficult 35x47 puzzle.**

I. Introduction

A. Nonograms

Nonograms are a popular Japanese logic puzzle. It is also known as picross and griddlers. In order to solve the puzzle, the player must use constraints provided for each row and column to fill in a grid. The puzzle is considered solved once all constraints are satisfied. These puzzles can be considered a special case of a more general binary image reconstruction problem or discrete tomography [1]. It was also shown in [1] that GA algorithm for solving discrete tomography problems developed in [2] can be easily applied to domain of Nonogram puzzles.

Nonograms consist of a grid m x n in length and width. Each row and column has a sequence of numbers. The total amount of elements in a sequence is the number of groups required. The actual values for the elements denote the number of squares in the group. A group consists of a set squares which are filled. Groups must be separated by an empty square. The sequence of numbers for rows and columns are considered constraints and reduce the state space. Difficulty can be attributed to how restrictive these constraints are. The more restrictive the easier the puzzle. These puzzles are generally intended to be solved by hand. Finding a solution for Nonogram has been proven to be NP-Complete [3][4].

Figure 1(a) shows a representation of an unsolved nonogram and figure 1(b) represents the solution. The integers located to the left of the rows and the top of the columns represent how many boxes have to be filled in on the grid. For example, “3,1” means that there is a group of 3 cells that are filled in to the left of the group of 1 cell with at least one of more empty cells in between. The group always appear in the same order as the string reading from left to right. Nonogram puzzles may have multiple solutions.

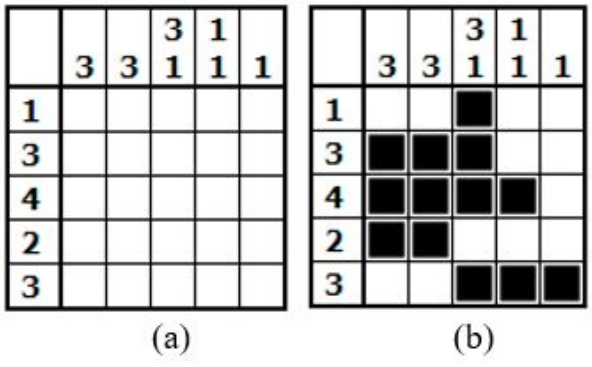

***Fig. 1** Example of a nonogram (a) unsolved puzzle and (b) solution.*

B. Genetic Algorithm

A genetic algorithm simulates a biological phenomenon where populations evolve over time to be better suited to their environment. “It is a probabilistic search algorithm based on the mechanics of natural selection and natural genetics” [5]. This is accomplished by assigning fitness to each member of a population and having the fittest survive and reproduce. This will cause the population’s fitness to improve. To preserve genetic diversity two methods are employed. First is mating which implements a crossover. Crossover results in the children to have a combination of genetic material from the parents. The second method is mutation. This is when the genetic material in an individual change. Without these two methods genetic diversity would diminish and the entire population would become clones. This would cause the population to get stuck on a local optimum if it has not found the solution. Each successive cycle of this process is known as a generation.

C. Wisdom of Crowds

Wisdom of crowds is a method of aggregating the collective wisdom of a population. It has been shown in [6] that the aggregate of proposed solutions from a group of individuals will usually yield better results than the single solution of an individual.

II. Prior work

It was showed in [1] that Nonogram puzzle is a special case of a more general Discrete Tomography problem. Discrete Tomography is a field that deals with reconstructing discrete images from its projection. In a black and white, n x m nonogram, it specifically deals with reconstructing a binary image from its two projections, such as row constraints and column constraints. Discrete tomography is a highly researched field, with many papers proposing various approaches to solve and model the problem [7, 8, 9]. There are many ways to sub-categorize the problem. For example [10] describes an algorithm to solve specific class of Discrete Tomography problem in polynomial time. The hv-convex polyominoes class of images describes all reconstructions that have black squares contiguous in all directions. In this paper, focus will be on the general Nonogram problem, that doesn’t assume any information about a reconstruction image, and may only have one solution.

There are many approaches to solve nonograms. Solvers can be found hosted online such as teal and others [11] [12]. These solvers however are limited in their ability to solve puzzles. Puzzles with multiple solutions or without restrictive constraints prove difficult for these solvers. This is due to the method being employed. The algorithms used iteratively searches and attempts to reduce the state space. This is done by using the constraints to determine the state of squares. If the algorithm can successfully determine a square or multiple squares each iteration it will be able to solve the puzzle. It is when it can no longer determine squares when it will fail. An example of constraints that would cause these solvers to fail are when all of the row and column constraints are 1.

Other approaches such as depth first search are guaranteed to find a solution [13]. However, once puzzles become large DFS has to search through an exponentially growing search space. The DFS algorithm complexity would be 2^n where n is the number of squares in the grid. This quickly becomes computationally prohibitive for larger puzzles.

Genetic algorithms are popular when attempting to solve nonograms. Often these are coupled with other search techniques to improve overall functionality such as a Taguchi-based genetic algorithm [14] or a memetic genetic algorithm [2]. This is due to the fact genetic algorithms by themselves often converge to a local optimum and can remain stuck for many generations. For larger puzzles, genetic algorithms often do not find the optimal solution [15]. Memetic genetic algorithm described in [2] adds a stochastic hill climb procedure in the crossover operator, to ensure the optimal solution in the current search space. This kind of algorithm is very computationally heavy for each crossover operation, but at the same time usually requires less population iterations. The nonogram can be converted into other problems such as a satisfiability problem and a genetic algorithm can be used [16].

Another approach is using an Intelligent Genetic Algorithm. This is the combination of a canonical genetic algorithm with specially tailored features in the genetic algorithm to enhance performance [17]. The latter approach was chosen in designing the genetic algorithm, and it is described in detail in the following section.

III. Proposed Approach

The methods proposed to approach this problem is genetic algorithms and wisdom of artificial crowds. Wisdom of crowds offers an approach which often outperforms the best results from GA run [18]. The algorithm is largely based on the works of [6,17]. These two approaches are combined, to solve a complex nonogram problem. It is important to select the proper parameters to ensure the genetic algorithm runs efficiently [19]. Parameters are empirically tested and prioritized for computational efficiency, result obtained, and population diversity.

A. Nonogram encoding for GA algorithm

Since the problem is a black and white image reconstruction problem, binary string encoding makes sense and will work, but better encodings were discovered. Such encoding is described in [17]. Instead of random generation of initial individuals, Information from constraints can be used to create segmentation lines that are solved in respect to either row numbers, or column numbers. Each segmentation line will have number of 1's equal to number of elements in the corresponding row constraint. Adjacent 1's are not allowed, and number of 0 's in the encoding corresponds to the actual number of 0's in binary encoding. Each block of consecutive black squares will be condensed to just 1 square.

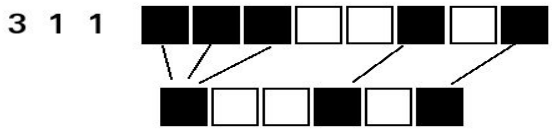

Fig. 2 *Condensed encoding example*

By using this encoding, grids are generated that have either all rows, or all columns adhere to the corresponding constraints. This also decreases the search space, as the problem is now only consists of one dimensional shifting of resulted segments in a way that would improve individual's fitness.

B. Initial Population

GA algorithm was designed that could generate initial populations with three different characteristics such as ROW_BIAS, COL_BIAS, RAND. By using the encoding described above, it is very easy to create images with either rows, or columns solved. This resulted in visible difference in initial populations, which will be referred to as ROW_BIAS and COL_BIAS. When generating population with respect to solved rows, it can be seen in Figure 3 that images appear to be more stretched out in one dimension. The opposite applies to COL_BIAS in Fig 4, with pictures appearing to be stretched out in the other dimension.

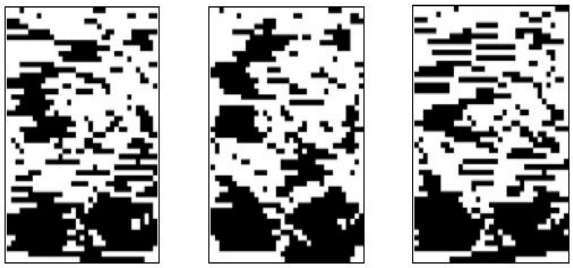

Fig. 3 *ROW_BIAS initial population individuals*

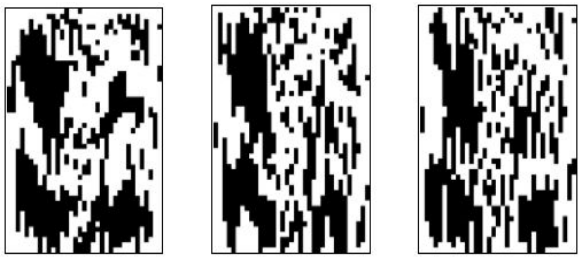

Fig. 4 *COL_BIAS initial population individuals*

These characteristics will produce two distinct populations, which is very useful for Wisdom of Crowds approach described later.

C. Fitness

The fitness value represents how close to a solution the individual is. Fitness of 0 represents an optimal solution. The fitness value is generated by totaling the number of incorrect squares and incorrect groups. The number of incorrect squares is calculated by counting the number of squares in the grid are filled in compared to the total number of squares in the corresponding constraint. The number of incorrect groups is calculated by finding the difference of consecutive black squares in a line and number of groups from the constraint. Each difference is multiplied by the corresponding weight parameter.

$$Fitness = square\_penalty * square\_weight + group\_penalty * group\_weight$$

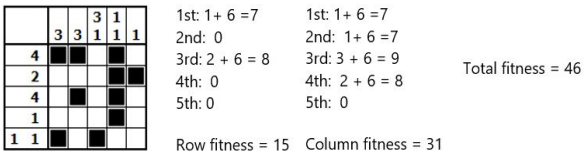

Fig.5 *Fitness example.*

D. Crossover

The GA will perform a crossover with two nonograms selected from a population by means of roulette wheel selection. To implement roulette wheel selection, the problem was converted from minimization to maximization. By estimating worst possible fitness and subtracting the current individual fitness value, problem was converted to maximization. Roulette wheel selection is important, as it allows for proportional probability distribution when selecting an individual. Once two parents are selected, it will randomly choose three points for crossover and combine the two arrays at these points to generate two offspring. Then the fitness value for each offspring is calculated and the highest fitness will be rejected while the lowest is added to the new population. The probability of crossover is defined by the CROSSOVER parameter, and set low. Note that if GA runs in ROW_BIAS mode, then crossover points will be randomly selected in the way that doesn't break current row's perfect fitness. The same applies to COL_BIAS mode.

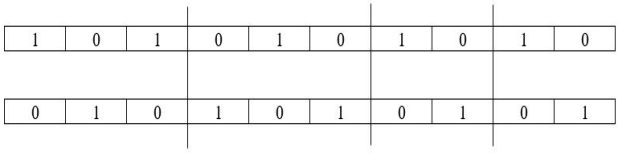

Fig. 6 *Representation of crossover points for parents*

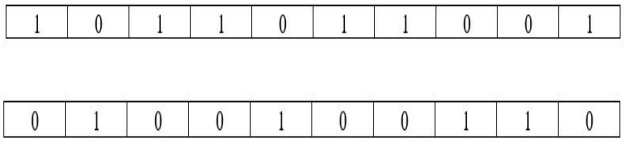

Fig. 7 *Offspring generated by parents*

E. Mutation

After each crossover operation, there is chance offspring will undergo a mutation. It is possible to alter the mutation rate. Probability of mutation was set to 0.01. This value for probability allowed for some mutation to occur but were unlikely to undo improvements made. If a line is selected to be mutated, then it will randomly choose an index and will flip a bit at that position to the opposite value. Figure 8 depicts a mutation. It is important to have a mutation function as this one of the methods of ensuring genetic diversity.

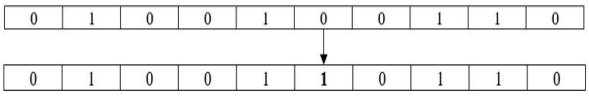

Fig. 8 *Example of mutation*

F. Wisdom of Crowds

Wisdom of crowds is a collective opinion of a group of individuals, rather than that of a single expert. This phenomenon is thoroughly studied in [21]. In [6] author applies principles of wisdom of crowds to domain of NP-hard problems. The idea of Wisdom of Crowds approach is to generate wise crowd that has to satisfy four criteria such as cognitive diversity, independence, decentralization, aggregation [6, 21]. It is important to create a wise crowd, or the result of aggregation will not yield any good results. Using the condensed encoding and penalty weights described earlier, it is possible to create six diverse individual experts. To create these individuals, GA is run six times, each time feeding it parameters that correspond to specific characteristics. In Figure 9 examples of a crowd generated is shown.

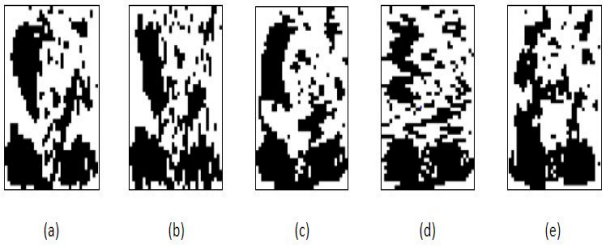

Fig. 9 *Example of a crowd: (a) COL_BIAS_GROUP, (b) COL_BIAS_SCATER, (c) ROW_BIAS_GROUP, (d) ROW_BIAS_SCATER, (e) RAND_GROUP*

In addition to feeding the GA different initial population, varying penalty weights were also utilized. By making group penalty weigh more during individual fitness calculation, GA will favor individuals with least amount of incorrect groups, which in turn results in individuals (a), (c), (e). By weighting square penalty heavily, results will favor individuals with least amount of total incorrect squares and will result in individuals that have images appear scattered. Combined with different

initial population generation methods, the algorithm can come up with up to 6 distinct experts, that were generated independently from each other, and all having their own area of expertise. Once a crowd is generated, each individual's result is aggregated. Aggregation happens on square by square basis. A variable THRESHOLD is assigned a percentage. If the crowd is in agreement on a square beyond the THRESHOLD, that square is included on the resulting grid. Anything in between is undecided, and needs to be handled, by some other post-process step.

IV. Experimental Results

A. Data

Constraints for two puzzles were used. One is of a 10 x 10 grid depicting a face and the second is a 35 x 47 grid depicting a panda. Additional random puzzles were created but were difficult to determine where deficiencies were in the solutions.

A log file was created to capture the statistics for each generation as the algorithm ran. The metrics captured were current generation's best, average, worst, and median fitness, and standard deviation. An image generation library Pillow was used to draw the grids for the initial population, final generation, and wisdom of artificial crowds.

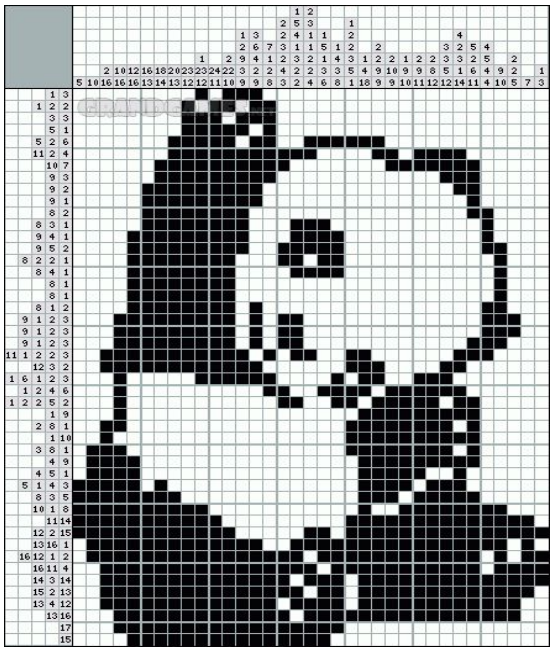

***Fig. 11**: 35x47 Panda.* Source: en.grandgames.net

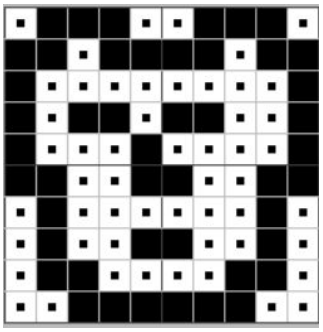

***Fig. 10**: 10x10 Face.* Source: www.pro.or.jp/~fuji/java/puzzle/nonogram/knowhow0-3-eng.html

B. Results

Software was run on Intel Core i5 processor (4.2 GHz), Ubuntu 17.10 OS. The algorithm can successfully solve 10x10 simple nonogram. Wisdom of Crowds performed as expected, even when not a single GA run produced and optimal solution, WoC seemed to remedy that most of the time, and result in optimal solution. The genetic algorithm by itself was usually unable to obtain an optimal solution. The solution each run provided was significantly improved over the initial population. Additionally, the genetic algorithm quickly converged while greatly improving fitness.

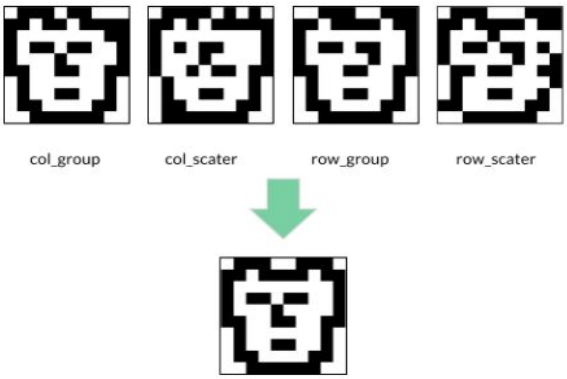

***Fig. 12**: 10x10 WoAC aggregate result*

In chart 1, average performance of a single GA run is shown. It takes around 30 iterations for a population of 100 to converge to a really good fitness values. Average running time of GA under these conditions is 0.26 seconds

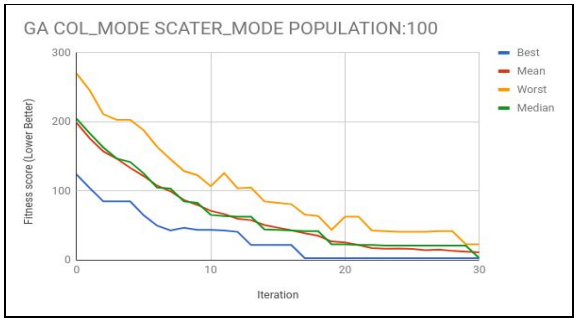

***Chart 1**: 10x10 average GA performance*

As it can be seen from the Chart 2, 7 out 10 times WoAC managed to get an optimal solution.

WoAC Run	Best Individual	Mean	Worst Individual	Median Individual	Std. deviation	Aggregation Result
0	1	9.083333333	24	6	8.9673	0
1	2	9.5	24	6	7.0947	2
2	1	9.25	23	8	6.4316	0
3	2	12.583	25	10	9.253	0
4	1	8.691	24	4	8.0490	0
5	1	7.1833	25	4	7.1830	0
6	1	11.3583	24	10	8.1465	3
7	0	7.9166	26	6	7.7183	0
8	0	9.475	25	8	8.7552	2
9	1	7.0166	21	6	5.4340	0

***Chart 2**: 10x10 average WoAC performance*

A larger 35x47 puzzle was attempted, but the algorithm did not find a solution. Every run of GA for a population of a 1000 would eventually converge and get stuck at the local minima.

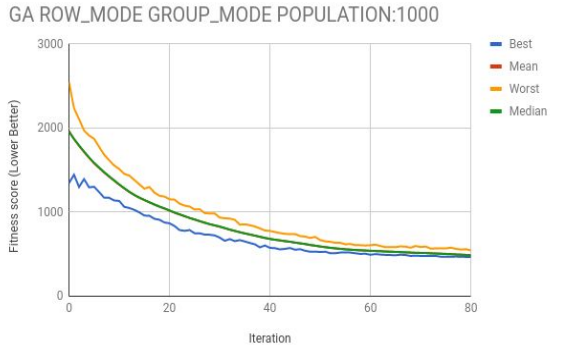

***Chart 3:** 35x47 nonogram stats*

For the 35 x 47 puzzle wisdom of crowds has difficulty determining some of the squares. The squares highlighted in pink in Figure 13 are undecided. These squares do not meet the threshold of agreement and cannot be decided by the crowd. However, the image created does show some resemblance to the solution. It takes around 2-3 hours to obtain a solution of similar quality.

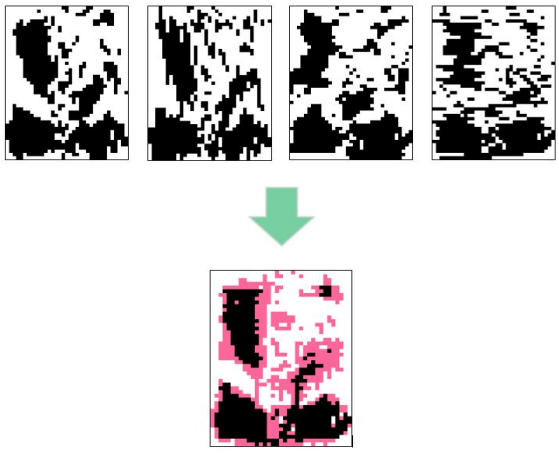

***Fig. 13**: 35x47 WoAC aggregate result*

V. Conclusions

By using genetic algorithms and wisdom of artificial crowds a 10x10 nonogram was solved. By assigning priority to group weight or square weight based on the number of incorrect squares and groups it was observed to vary the results of the genetic algorithm. When group weight was prioritized the results tended to have results that were optimized toward groupings and vice versa for square weight. When generating multiple solutions with varying priorities a variety of solutions are produced. This is ideal when using wisdom of crowds. Having a varied population of experts gives good results when aggregating.

Some possible improvements could modify properties of the genetic algorithm. If the genetic algorithm can further improve upon the solutions then the expert crowd would improve further. One method would be to vary the probability for crossover. This would help to preserve genetic diversity while also sustaining convergence [20]. Changing the mutation function in the genetic algorithm to use an intelligent mutate where groups are shifted around rather than flipping individual squares would help preserve the fitness of individuals in the population. Thirdly, including more experts in the population to have a larger crow of experts would likely improve the results. For wisdom of crowds, determining a method to handle undecided squares would be beneficial for larger, less agreed on solutions.

VI. References

[1] Batenburg, Kees Joost, and Walter A. Kosters. "A discrete tomography approach to Japanese puzzles." *Proceedings of the 16th Belgium-Netherlands Conference on Artificial Intelligence (BNAIC)*. 2004.

[2] Batenburg, Kees Joost. "An evolutionary algorithm for discrete tomography." *Discrete applied mathematics* 151.1 (2005): 36-54.

[3] Ueda, Nobuhisa, and Tadaaki Nagao. "NP-completeness results for NONOGRAM via parsimonious reductions." *preprint*(1996).

[4] van Rijn, J. N. *Playing games: The complexity of Klondike, Mahjong, nonograms and animal chess*. Diss. Master's thesis, Leiden Inst. of Advanc. Computer Science, Leiden Univ, 2012.

[5] Kumar, Manoj, et al. "Genetic algorithm: Review and application." *International Journal of Information Technology and Knowledge Management* 2.2 (2010): 451-454.

[6] Yampolskiy, Roman V., and Ahmed El-Barkouky. "Wisdom of artificial crowds algorithm for solving NP-hard problems." *International Journal of Bio-Inspired Computation* 3.6 (2011): 358-369.

[7] Ryser, Herbert J. "Combinatorial properties of matrices of zeros and ones." *Classic Papers in Combinatorics*. Birkhäuser Boston, 2009. 269-275.

[8] Gardner, Richard J., Peter Gritzmann, and Dieter Prangenberg. "On the computational complexity of reconstructing lattice sets from their X-rays." *Discrete Mathematics* 202.1-3 (1999): 45-71.

[9] Herman, Gabor T., and Attila Kuba, eds. *Advances in discrete tomography and its applications*. Springer Science & Business Media, 2008.

[10] Barcucci, Elena, et al. "Reconstructing convex polyominoes from horizontal and vertical projections." *Theoretical computer science* 155.2 (1996): 321-347.

[11] Nonogram Solver, http://a.teall.info/nonogram/

[12] Griddlers Solver, http://www.griddler.co.uk/Solve.aspx

[13] Jing, Min-Quan, et al. "Solving Japanese puzzles with logical rules and depth first search algorithm." *Machine Learning and Cybernetics, 2009 International Conference on*. Vol. 5. IEEE, 2009.

[14] Tsai, Jinn-Tsong. "Solving Japanese nonograms by Taguchi-based genetic algorithm." *Applied Intelligence* 37.3 (2012): 405-419.

[15] Bobko, Alicja, and Tomasz Grzywacz. "Solving nonograms using genetic algorithms." *Computational Problems of Electrical Engineering (CPEE), 2016 17th International Conference*. IEEE, 2016.

[16] Wiggers, Wouter, and Willem van Bergen. "A comparison of a genetic algorithm and a depth first search algorithm applied to Japanese nonograms." *Twente student conference on IT*. 2004.

[17] Tsai, Jinn-Tsong, Ping-Yi Chou, and Jia-Cen Fang. "Learning intelligent genetic algorithms using Japanese nonograms." *IEEE Transactions on Education* 55.2 (2012): 164-168.

[18] Yampolskiy, Roman V., Leif Ashby, and Lucas Hassan. "Wisdom of artificial crowds—a metaheuristic algorithm for optimization." *Journal of Intelligent Learning Systems and Applications* 4.02 (2012): 98.

[19] Deb, Kalyanmoy, and Samir Agrawal. "Understanding Interactions among Genetic Algorithm Parameters." *FOGA*. 1998.

[20] Srinivas, Mandavilli, and Lalit M. Patnaik. "Adaptive probabilities of crossover and mutation in genetic algorithms." *IEEE Transactions on Systems, Man, and Cybernetics* 24.4 (1994): 656-667.

[21] Surowiecki, James. The wisdom of crowds. Anchor, 2005.

[22] Nonograms puzzle, https://www.puzzle-nonograms.com/