Learning Intelligent Genetic Algorithms Using Japanese Nonograms

Jinn-Tsong Tsai, Ping-Yi Chou, and Jia-Cen Fang

***Abstract*—An intelligent genetic algorithm (IGA) is proposed to solve Japanese nonograms and is used as a method in a university course to learn evolutionary algorithms. The IGA combines the global exploration capabilities of a canonical genetic algorithm (CGA) with effective condensed encoding, improved fitness function, and modified crossover and mutation. In this paper, the condensed encoding ensures that the chromosome is a feasible solution in all rows for Japanese nonograms. In the reconstruction process of a Japanese nonogram, the numbers in the left column are used as encoding conditions, and the numbers in the top row with the improved fitness function are employed to evaluate the reconstruction result. From the computational experiments, the proposed IGA approach is applied to solve Japanese nonograms effectively, with better results than using a CGA. The students of the Department of Computer Science, National Pingtung University of Education, Taiwan, have gained practical experience of applying evolutionary algorithms to solve Japanese nonograms using both the proposed IGA and a CGA. The students learn that the IGA can find the right solution of the puzzle effectively, but the CGA cannot.**

***Index Terms*—Condensed encoding, genetic algorithms (GAs), Japanese nonograms, pedagogical issues, teaching sequence.**

I. Introduction

EVOLUTIONARY algorithms (EAs) are an important skill for solving optimization problems. The majority of computer science departments in the world offer courses on EAs for their students to learn the optimization method. The most popular EA is a genetic algorithm (GA) [1], [2]. In optimization approaches, the GA-based approach demonstrated very promising results from experimentation and practice in many areas [3], [4]. In the past few years, GA has received much attention for solving combinatorial optimization, such as the traveling salesman problem (TSP) [5] and the job-shop scheduling problem (JSP) [6]–[8]. Solving these combinatorial optimization problems efficiently using the GA-based approach was regarded as a way to give students experience [9]–[13]. For instance, the TSP was used as an example in the course described in [14]. While the TSP and JSP are very famous benchmark problems as application examples for learning GA, they are difficult examples for a novice due to the permutation encoding and the infeasible solution. In Salcedo-Sanz *et al.* [15], the authors used Japanese puzzles to teach advanced features of the evolutionary algorithm. All the students agreed that they had understood concepts better as a result. Therefore, this paper describes the use of interesting Japanese nonograms as examples upon which young students can learn GA-based methods, employing an intelligent genetic algorithm (IGA) to solve the nonograms effectively. Pedagogical issues addressed are the following: 1) What impact does the proposed IGA have on learning and teaching? When the IGA is used, do teachers teach differently and do students learn more? 2) What can the IGA do that the canonical GA cannot do? What is the advantage to using the IGA? 3) Can "achievement" be measured in students and teachers when the proposed IGA is applied to solve Japanese nonograms?

Japanese nonograms are fascinating puzzles that can be found in several popular magazines. The simpler puzzles can usually be solved by simple logic reasoning on a single row or column. More difficult puzzles work on searching for contradictions. When the puzzle is simple or is a small set, it is solvable using simple logic rules and offers lots of fun. However, when the puzzle becomes more difficult or a larger superset, and becomes an NP-complete and constrained combinatorial problem, it is sometimes not easy for humans to solve [15]–[17]. Since the GA-based method has acquired a good reputation for solving NP-complete combinatorial problems in recent decades, Japanese nonograms provide challenging examples for learning and teaching the GA-based method.

The proposed IGA uses effective condensed encoding, an improved fitness function, and modified crossover and mutation to modify a canonical GA (CGA). These improved operators augment the abilities of the CGA to allow it to solve Japanese nonograms. In brief, the proposed IGA enhances a CGA by including the abilities of global exploration and robustness.

II. Curriculum Content for Training Software Engineers

In this study, the Japanese nonograms are adopted as interesting curriculum content to train computer science students in using the GA-based method. Internationally, Japanese nonograms are very popular games, with a grid of N rows and M columns with numbers placed, for example, on the left column and the top row as shown in Fig. 1. The numbers represent the number of blocks that must be filled in the grid, and the order in which they appear. The blocks have to be separated by at least one blank block if there are two or more numbers [15]–[17]. For example, a clue of "2 3 2" in the sixth row of Fig. 1 means that, in that row, there are sets of two, three, and two filled blocks, in that order, with at least one blank block between successive groups. In order to solve a puzzle, the player needs to determine which blocks are going to be boxes and which are going to be spaces.

Manuscript received January 18, 2011; revised April 05, 2011; accepted May 12, 2011. Date of publication June 13, 2011; date of current version May 01, 2012. This work was supported in part by the National Science Council, Taiwan, under Grant NSC99-2221-E153-001.

The authors are with the Department of Computer Science, National Pingtung University of Education,Pingtung 900, Taiwan (e-mail: jttsai@mail.npue.edu.tw).

Digital Object Identifier 10.1109/TE.2011.2158214

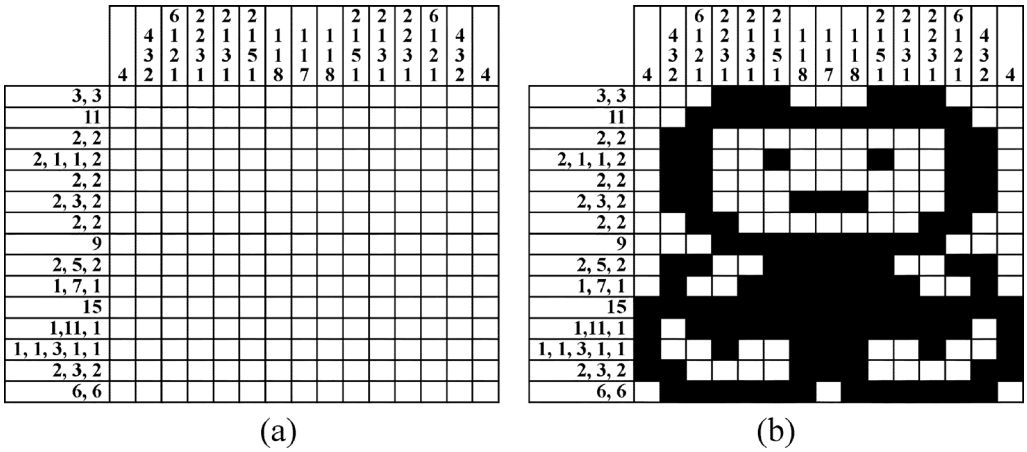

Fig. 1. (a) Japanese nonogram. (b) Its corresponding solution "Bear."

A step-by-step sequence to teach students GA-based methods by using Japanese nonograms is the following.

Step 1) Practice Japanese nonograms to make the students understand that they are working with a combinatorial problem.

Step 2) Study encoding and fitness function in Japanese nonograms for the future application of the GA-based method, as shown in Section III.

Step 3) Apply the CGA with the binary encoding and the improved fitness function to solve a Japanese nonogram, as shown in Section IV.

Step 4) Use the proposed IGA with effective condensed encoding, improved fitness function, and modified crossover and mutation to solve a Japanese nonogram, as shown in Section V.

Step 5) Guide the students through experiments and comparisons to determine the advantages of the proposed IGA and its operators, as shown in Section VI.

III. Encoding and Fitness Function in Japanese Nonograms for IGA and CGA

This section describes how to encode a Japanese nonogram and make the fitness function in evaluating achievement for the IGA and the CGA. The condensed encoding and the improved fitness function are created for the proposed IGA, while the binary encoding and the improved fitness function are adopted for the CGA.

A. Condensed Encoding and Improved Fitness Function for the IGA

In reconstructing a Japanese nonogram, the numbers in the left column are used as encoding conditions, and the numbers in the top row are employed to evaluate the reconstruction result.

A line segmentation $L_i = (l_{i1}, l_{i2}, \ldots, l_{ij}, \ldots, l_{iM})$ is defined corresponding to a row of a Japanese nonogram, where $i \in [1, N]$. The numbers in the line segmentation correspond to the alternating black and white segments. The encoding is $l_{ij} = 0$ if it is white, and $l_{ij} = 1$ if it is black. Then, the black segments with condensed encoding are considered to be implicitly expressed. Therefore, the condensed encoding only shows the actual white pixels and the condensed black pixels. The advantage of the condensed encoding is that it ensures that the numbers of the line segmentation are mapped to be a feasible solution in all rows. In the proposed IGA, a chromosome is generated by the condensed encoding in all rows of a puzzle and one that is feasible in all rows but not necessarily in columns. In addition, the search space size for the IGA would reduce to $\prod_{i=1}^{N} R_i$, where R $(= C^{\text{zero no.}+1}_{\text{condition no.}})$ is the number of possible solutions in each row.

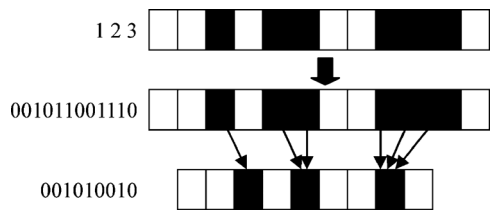

Fig. 2. *(top)* Actual line, *(middle)* corresponding encoding line, and *(bottom)* condensed encoding line.

For example, as shown in Fig. 2, the numbers of the actual line are "1 2 3," and the alternating black and white segments are shown in the line. The middle line presents its corresponding encoding "001011001110." The condensed encoding is "001010010," as shown in the bottom line, as the black segments are suggested without being directly expressed. In the example of the condition numbers "1 2 3," the search space size is $C_3^{6+1} = 7!/4!\,3! = 35$.

When the numbers in the left column are used as condensed encoding conditions, there are no unfeasible solutions in all rows. The improved fitness function is defined to compare, in each column, the desired numbers of 1's with the actual numbers of 1's of a given solution X. The desired numbers of 1's in the top row of a puzzle are described to be c_{ij}, where $i \in [1, M]$ and j is the condition number. For Fig. 1 as an example, the condition in column 1 has one number, which represents $c_{11} = 4$.

The conditions in column 2 have three numbers, which represent $c_{21} = 4$, $c_{22} = 3$, and $c_{22} = 2$. The improved fitness function is defined as follows:

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

where M is the number of columns and Q is the number of inconsecutive 1's in each column of a given solution X. $f(X)$ only takes into account the difference between the desired numbers of 1's and the actual numbers of 1's of a given solution X in all columns because there are no differences in all rows due to conducting condensed encoding. The objective is to minimize $f(X)$ to zero.

The advantage of the improved fitness function is not only to compute the numbers of 1's, but also to consider the order of 1's in all columns. For example, suppose the desired numbers are 2 and 1 in a column, such that $c_{11} = 2$, $c_{12} = 1$, and the rest $c_{1j} = 0$. If the actual number and the order of 1's in a column have three possible solutions (e.g., "0 1 1 0 1 0," "1 0 1 0 1 0," and "1 0 0 1 1 0"): $x_{11} = 2$, $x_{12} = 1$, presented in the first solution; $x_{11} = 1$, $x_{12} = 1$, $x_{13} = 1$, shown in the second one; and $x_{11} = 1$, $x_{12} = 2$, described in the third one. The fitness value for each possible solution is given in the following: $f(X)_{\text{first}} = |2-2| + |1-1| = 0$, $f(X)_{\text{second}} = |2-1| + |1-1| + |0-1| = 2$, and $f(X)_{\text{third}} = |2-1| + |1-2| = 2$. It can be observed that the first solution corresponds to the desired solution, and the others do not. From this case, it can be seen that the three possible solutions have the same numbers of 1's and 0's. If the objective functions given by Salcedo-Sanz *et al.* [15] are used to evaluate the result, the evaluation values are equal to zero in the three possible solutions; it cannot differentiate between them. Therefore, it can be shown that the accuracy of the proposed fitness function is better than that of the objective functions given by Salcedo-Sanz *et al.* [15], which only consider the difference between the desired numbers of the 1's and 0's and the actual numbers of the 1's and 0's of a given solution X in all rows and columns.

In terms of teaching, when the improved fitness function was proposed in this study, teachers could understand its utility for solving Japanese nonograms and were able to teach it to students. In turn, students were able to learn the importance of the fitness function for solving the optimization problem and could select an appropriate fitness function and adopt it to solve Japanese nonograms.

B. Encoding and Improved Fitness Function for the CGA

In the reconstruction process, a possible encoding can be accomplished by using a binary string of length $N \times M$. With this encoding, the search space size of a Japanese nonogram is equal to $2^{N \times M}$ for the CGA.

The improved fitness function is also used in the CGA. A binary string, generated at random, is employed for a puzzle. Therefore, the difference between the desired numbers of the 1's and the actual numbers of the 1's of a given solution X in all columns and rows must be considered. The desired numbers of 1's in the puzzle are c_{ij} in columns and r_{ij} in rows. The improved fitness function is redefined as follows:

$$f(X) = \sum_{i=1}^{M} \sum_{j=1}^{Q} |c_{ij} - x_{ij}| + \sum_{i=1}^{N} \sum_{j=1}^{Q} |r_{ij} - x_{ij}|, \quad (2)$$

where M is the number of columns, N is the number of rows, and Q represents the number of inconsecutive 1's in each column and row of a puzzle solution X. The objective is to minimize $f(X)$ to zero.

IV. Using the CGA to Solve Japanese Nonograms

This section describes the operators and steps of the CGA; those for the IGA follow in the next section.

A. CGA Operators

1) Initialization: In the initialization process, each chromosome is described by a binary vector of $N \times M$ bits. All $N \times M$ bits are initialized randomly.

2) Crossover Operation: The standard one-cut-point crossover is applied in the CGA. For each pair of selected coupled chromosomes, it randomly selects one cut-point and exchanges the right genes of two chromosomes after the cut-point to generate a pair of their offspring.

3) Mutation Operation: The mutation operator in the CGA alters one or more genes in a chromosome with a probability of mutation. If a gene of 0 is determined for mutation, it would be flipped into 1, and vice versa.

B. Steps in the CGA Approach

Step 1) Set the parameters: population size p_{s}, crossover rate p_{c}, and mutation rate p_{m}.
Step 2) Initialize.
Step 3) Perform a roulette wheel for selection.
Step 4) Conduct the one-cut-point crossover operation. Selected coupled chromosomes are determined to perform the crossover by depending on the crossover rate p_{c}.
Step 5) Carry out the mutation operation changing from 0 to 1, or vice versa. A gene is determined to perform the mutation depending on the mutation rate p_{m}.
Step 6) Generate the offspring population.
Step 7) Sort the fitness values in increasing order among parents and offspring populations.
Step 8) Select the better p_{s} chromosomes as parents of the next generation.
Step 9) Has the stopping criterion been met? If yes, then go to Step 10. Otherwise, return to Step 3 and continue through Step 9.
Step 10) Display the optimal chromosome and fitness value.

V. Using the IGA to Solve Japanese Nonograms

A. IGA Operators

1) Generation of Initial Population: The condensed encoding scheme mentioned above is used to reproduce an initial chromosome by randomly generating 0's and 1's based on the condensed encoding rule. In each section (that is, each row of a puzzle) of a chromosome, none of the 1's are adjoining.

The initialization procedure produces p_s chromosomes by the following algorithm, where p_s denotes the population size.

Algorithm

Step 1: Generate the genes of a section (that is, a row of a puzzle) of a chromosome by randomly using 0's and 1's. The number of 1's is based on the condensed encoding rule, but no 1 is next to another. In each section, the numbers of 0's are equal to $M -$ (sum of the numbers shown in the left column of a puzzle), where M is the number of columns of a puzzle.

Step 2: Perform Step 1 N times to create the genes for N sections of a chromosome, where N is the number of rows of a puzzle.

Step 3: Carry out the above two steps p_s times. p_s possible solutions of a puzzle are produced.

2) Offspring Generation by Modified Crossover: The crossover operator performs a single-point operation to maintain a high diversity of chromosomes. In Japanese nonograms, a standard single-point crossover would produce unfeasible solutions in rows when using the condensed encoding scheme because it randomly swaps genes. In this paper, the modified crossover operator forces operations between individuals, that is, swaps between rows instead of between genes. They thus keep the feasible solutions in each row.

For each pair of coupled chromosomes selected for crossover, a random integer number p is generated from $[1 \ldots N]$, where N is the number of rows of a puzzle. The number p indicates the position of the crossing point. Two chromosomes $(R_1 R_2 \ldots R_p R_{p+1} \ldots R_N)$ and $(S_1 S_2 \ldots S_p S_{p+1} \ldots S_N)$ are replaced by a pair of their offspring: $(R_1 R_2 \ldots S_p S_{p+1} \ldots S_N)$ and $(S_1 S_2 \ldots R_p R_{p+1} \ldots R_N)$, where, for example, R_1 is the first-row condensed encoding of one puzzle and S_1 is from the other puzzle. The crossing point in the crossover operator is a row number instead of a gene number, thus the feasibility in rows is ensured.

3) Mutation Operation: The modified mutation operator also works on the individual row. Once a gene of 0's or 1's is mutated, it shifts to another position on its individual row, but with no adjoining 1's. This ensures the feasibility of the rows.

B. Steps in the Proposed IGA Approach

Step 1) Set the parameters: population size p_s, crossover rate p_c, and mutation rate p_m.

Step 2) Initialize the population by using the condensed encoding. Then, convert the condensed encoding at each chromosome to its corresponding encoding and compute its fitness value.

Step 3) Perform a roulette wheel for selection.

Step 4) Conduct the modified crossover by forcing operation between individuals. Selected coupled chromosomes are determined to perform the crossover depending on the crossover rate p_c.

Step 5) Carry out the modified mutation by shifting a gene of 0's or 1's to another position on its individual row. However, none of the 1's can adjoin. A gene is determined to perform the mutation, depending on the mutation rate p_m.

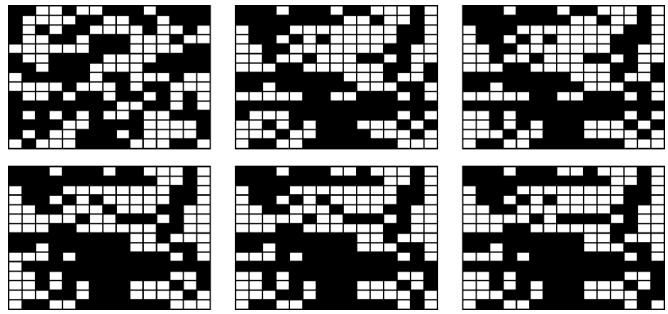

Fig. 3. Evolution in pursuing the best solution "Bear" using the CGA.

Step 6) Generate offspring population.

Step 7) Sort the fitness values in increasing order among parents and offspring populations.

Step 8) Select the better p_s chromosomes as parents of the next generation.

Step 9) Has the stopping criterion been met? If yes, then go to Step 10. Otherwise, return to Step 3 and continue through Step 9.

Step 10) Display the optimal chromosome and fitness value.

VI. Educational Aspects in Computational Results and Comparisons

Pedagogically, the goal of this work is to make students realize that they are working with the GA-based method and a combinatorial optimization problem, the Japanese nonogram "Bear." This puzzle is easy to understand, but very hard to solve. The steps described below guide the students through experiments and comparisons to determine the advantages of the IGA and its operators. The same nonogram "Bear" is adopted to test the proposed IGA and the CGA and to compare their performance. First, the students use the CGA to solve the puzzle. Then, they employ the proposed IGA with its operators as mentioned above to implement the same puzzle. The details of experiments and comparisons follow.

First, the CGA is used to solve the nonogram "Bear," as shown in Fig. 1. It includes a binary encoding, the improved fitness function, a roulette wheel for selection, a canonical one-cut-point crossover, and a mutation changing from 0 to 1 or vice versa. For the nonogram "Bear," the length of a binary vector is 225, and the search space size is 2^{225} $(= 5.392 \times 10^{67})$. The following evolutionary environments are used: the population size p_s is 100, the crossover rate p_c is 0.9, and the mutation rate p_m is 0.05. The stopping condition is that the number of function evaluations is 100 000. The best fitness value, the average fitness value, and the standard deviation of the fitness value in 30 independent runs for the puzzle "Bear" by using the CGA are respectively 42, 58, and 8. As an example, the evolutions in pursuing the best solution "Bear" are shown in Fig. 3. It can be seen that the CGA cannot converge to the exact solutions of the puzzle within the stopping condition, mainly because the CGA cannot find the right directions to follow in the huge search space size of 2^{225}.

Then, the proposed IGA is used to fulfill the nonogram "Bear." Under the condensed encoding, all chromosomes are feasible in rows but not in columns. The search space size in "Bear" is 6.214×10^{20}. In the IGA, the condensed encoding, improved fitness function, a roulette wheel for selection, and modified crossover and mutation are used to implement the puzzle. The evolutionary environments for the IGA are the

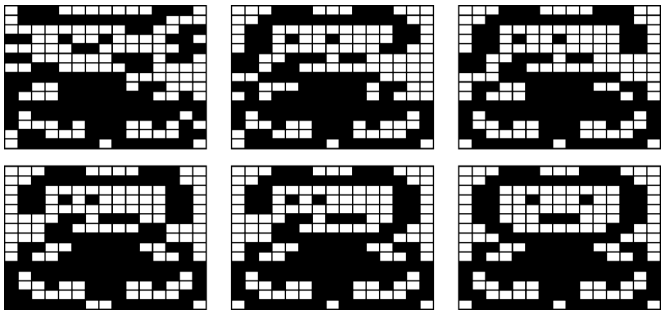

Fig. 4. Evolution in pursuing the best solution "Bear" using the IGA.

same as those for the CGA. The best fitness value, the average fitness value, and the standard deviation of the fitness value in 30 independent runs for the puzzle "Bear" are respectively 0, 4, and 4. Fig. 4 shows the evolution in pursuing the best solution "Bear." From the computational results and from Figs. 3 and 4, it can be seen that: 1) the proposed IGA can obtain the correct puzzle solution, but the CGA cannot; 2) the proposed IGA gives a better fitness value and a better average fitness value than does the CGA; 3) the proposed IGA gives a smaller standard deviation of the fitness value compared to the CGA, hence the proposed IGA approach has a more robust and stable solution quality. These results all indicate that the IGA, in general, gives a better quality solution than the CGA. Hence, it can be concluded that the proposed IGA is a viable high-quality approach for solving the Japanese nonogram "Bear," and that students can learn that it can effectively find the right puzzle solution while the CGA cannot.

The CGA and the IGA computational processes described here were taught to the 15 students in the course "Genetic Algorithms" in the Department of Computer Science, National Pingtung University of Education, Taiwan. The teacher explained the GA concepts and the steps in the solution. Then, the students, as homework, used the CGA and the IGA. Each student carried out two exercises, discussed their results and drew comparisons, and got the same results as described above. All the students and their teacher agreed that they had understood many evolutionary concepts in solving Japanese nonograms, the bottleneck in the CGA solution, and the advantages of the IGA such as the condensed encoding, improved fitness function, modified crossover and mutation, and so on. They also agreed that Japanese nonograms are both interesting problems and real-world applications for learning evolutionary algorithms. In addition, the students and teachers said they would be motivated to enhance the CGA by including more advanced and efficient techniques in its operators. The improved CGA could effectively solve similar optimization problems.

VII. Conclusion

This paper proposes the use of the IGA to teach evolutionary algorithms to university students and to solve Japanese nonograms as the application cases in an algorithms course. The proposed IGA includes condensed encoding, an improved fitness function, and modified crossover and mutation. In addition, it can reduce the search space size for Japanese nonograms to $\prod_{i=1}^{N} R_i$ instead of $2^{N \times M}$. From the computational experiments, the students believe that the proposed IGA, in general, gives a better quality solution than does the CGA. They also learn that, in some cases, improved operators need to be incorporated into the CGA to solve real-world problems.

References

[1] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Reading, MA: Addison-Wesley, 1989.

[2] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*. Berlin, Germany: Springer-Verlag, 1996.

[3] J. T. Tsai, T. K. Liu, and J. H. Chou, "Hybrid Taguchi-genetic algorithm for global numerical optimization," *IEEE Trans. Evol. Comput.*, vol. 8, no. 4, pp. 365–377, Aug. 2004.

[4] J. T. Tsai, J. H. Chou, and T. K. Liu, "Tuning the structure and parameters of a neural network by using hybrid Taguchi-genetic algorithm," *IEEE Trans. Neural Netw.*, vol. 17, no. 1, pp. 69–80, Jan. 2006.

[5] P. Larranaga, C. M. H. Kuijpers, R. H. Murga, I. Inza, and S. Dizdarevic, "Genetic algorithms for the travelling salesman problem: A review of representations and operators," *Artif. Intell. Rev.*, vol. 13, pp. 129–170, 1999.

[6] L. Wang and D. Z. Zheng, "A modified genetic algorithm for job shop scheduling," *Int. J. Adv. Manuf. Technol.*, vol. 20, pp. 72–76, 2002.

[7] T. K. Liu, J. T. Tsai, and J. H. Chou, "Improved genetic algorithm for job-shop scheduling problem," *Int. J. Adv. Manuf. Technol.*, vol. 27, no. 9–10, pp. 1021–1029, 2006.

[8] J. T. Tsai, T. K. Liu, W. H. Ho, and J. H. Chou, "An improved genetic algorithm for job-shop scheduling problems using Taguchi-based crossover," *Int. J. Adv. Manuf. Technol.*, vol. 38, no. 9–10, pp. 987–994, 2008.

[9] Y. J. Cao and Q. H. Wu, "Teaching genetic algorithm using MATLAB," *Int. J. Elect. Eng. Educ.*, vol. 36, pp. 139–153, 1999.

[10] Y. H. Liao and C. T. Sun, "An educational genetic algorithms learning tool," *IEEE Trans. Educ.*, vol. 44, no. 2, pp. 20–30, May 2001.

[11] E. Bütün, "Teaching genetic algorithms in electrical engineering education: A problem-based learning approach," *Int. J. Elect. Eng. Educ.*, vol. 42, no. 3, pp. 223–233, 2005.

[12] A. Venables and G. Tan, "A "hands on" strategy for teaching genetic algorithms to undergraduates," *J. Inf. Technol. Educ.*, vol. 6, pp. 249–261, 2007.

[13] J. Li and Z. Zhang, "A learning tool of genetic algorithm," in *Proc. 2nd Int. Workshop Educ. Technol. Comput. Sci.*, 2010, vol. 1, pp. 443–446.

[14] G. Pataki, "Teaching integer programming formulations using the traveling salesman problem," *SIAM Rev.*, vol. 45, no. 1, pp. 116–123, 2003.

[15] S. Salcedo-Sanz, J. A. Portilla-Figueras, E. G. Ortiz-Garcia, A. M. Perez-Bellido, and X. Yao, "Teaching advanced features of evolutionary algorithms using Japanese puzzles," *IEEE Trans. Educ.*, vol. 50, no. 2, pp. 151–156, May 2007.

[16] J. Benton, R. Snow, and N. Wallach, "A combinatorial problem associated with nonograms," Artificial Intelligence Dept., Stanford Univ., CA, Tech. rep., 2005.

[17] K. J. Batenburg and W. A. Kosters, "Solving nonograms by combining relaxations," *Pattern Recogn.*, vol. 42, no. 8, pp. 1672–1683, 2009.

Jinn-Tsong Tsai received the B.S. and M.S. degrees in mechanical and electromechanical engineering from National Sun Yat-Sen University, Taiwan, in 1986 and 1988, respectively, and the Ph.D. degree in engineering science and technology from National Kaohsiung First University of Science and Technology, Taiwan, in 2004.

He is currently an Associate Professor with the Department of Computer Science, National Pingtung University of Education, Taiwan. From 1990 to 2004, he was a Researcher and the Chief of the Automation Control Section of Metal Industries Research and Development Center, Taiwan. From 2004 to 2006, he was an Assistant Professor with the Medical Information Management Department, Kaohsiung Medical University, Taiwan. His research interests include evolutionary computation, intelligent control and systems, neural networks, and quality engineering.

Ping-Yi Chou received the B.S. degree in computer science from the National Pingtung University of Education, Taiwan, in 2010, and is currently pursuing the M.S. degree in computer science at the same university.

His research interests are evolutionary algorithms and cloud computing.

Jia-Cen Fang received the B.S. degree in bioinformatics from Asia University, Taiwan, in 2010, and is currently pursuing the M.S. degree in computer science at the National Pingtung University of Education, Taiwan.

Her research interests are evolutionary algorithms and cloud computing.