Assembly of Puzzles Using a Genetic Algorithm

Fubito Toyama, Yukihiro Fujiki, Kenji Shoji, and Juichi Miyamichi
Faculty of Engineering, Utsunomiya University
7-1-2, Yoto, Utsunomiya-shi, 321–8585 JAPAN
fubito@is.utsunomiya-u.ac.jp

Abstract

In this paper, we proposed a method for solving the rectangle piece jigsaw puzzle assembly problem. A shape of a piece is a rectangle, and a picture of a puzzle is only painted in black and white, i.e., puzzles are processed as binary images. The assembly of the puzzle is performed only using information of the pixel value on the border line of the pieces. This problem cannot be solved by the simple local piece matching because there are many similar pieces. Global matching is required. The proposed method utilizes a genetic algorithm (GA) to search the optimum piece arrangement because GA has the ability to find the global solution in the large optimization space. The proposed method correctly assembled all pieces in the 8 × 8-piece puzzle.

1. Introduction

Solving puzzles by computer is typical pattern recognition problem. When a shape of a piece is a rectangle, this problem is especially difficult and contains a number of problems endemic to applications such as the optimum location problem and image mosaicking. A shape of a piece is a rectangle and a picture of a puzzle is only painted in black and white, i.e., puzzles are processed as binary images. We assume that a piece does not rotate. Thus, the piece directions are fixed. But the puzzle problem is very difficult on this condition. We proposed a method for solving this rectangle piece jigsaw puzzle assembly problem. Figure 1 shows an example of puzzles used in this paper.

The previous works in jigsaw puzzle assembly can be found in [1][2][3][4]. Only shape information is utilized in these methods. Pictorial information is not a factor.

In this method, the assembly of the puzzle is performed only using information of the pixel values (1 or 0) on the border line of the pieces which are processed as binary images. This puzzle problem can be considered the problem for searching piece arrangement such that the total difference of the pixel values on the border line of all neighboring pieces becomes smallest. This problem cannot be solved by the simple local piece matching. Global matching is required. The proposed method utilizes a genetic algorithm (GA) to search the optimum piece arrangement because GA has the ability to find a global solution in a large optimization space. The proposed method correctly assembled all pieces in the 8 × 8-piece puzzle.

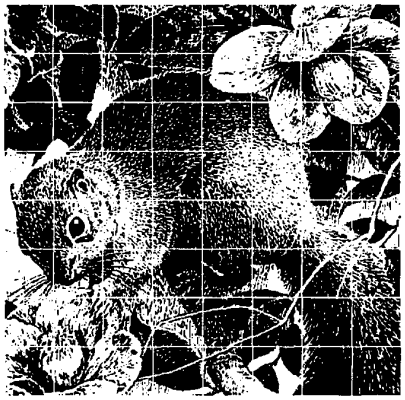

Figure 1. Example of the puzzle used in this paper (8 × 8-piece puzzle).

2. Assembly of puzzles using a GA

A rectangle puzzle consists of $N \times M$ pieces. We call $n \times m$ puzzle piece arrangement "partial piece arrangement", $(1 \leq n \leq N)$ and $(1 \leq m \leq M)$.

The puzzles used in this paper have a unique solution in which fitness function F (this function is described in Section 2.2) has the maximum value. The assembly of the puzzle is performed only using the pixel values on the border line of the pieces. Thus, only information which is shown as figure 2 is actually used in the assembly of the puzzle. Figure 2 shows only information on the border line of the pieces of figure 1. We treat the piece arrangement problem in which global matching is required. Therefore, only pixel values on the border line of the pieces are used in this paper. This approach cannot handle extra and missing pieces.

In the puzzle problem, there is the constraint that every

1051-4651/02 $17.00 © 2002 IEEE

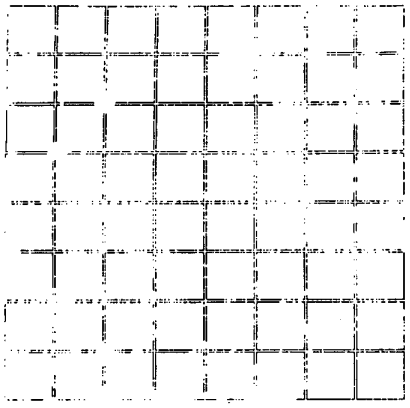

Figure 2. Information on the border line of the pieces of figure 1.

piece must be used exactly once. Thus, the definition of chromosomes and genetic operators which satisfy this constraint are required. In this method, partial piece arrangement and $N \times M$ piece arrangement are defined as an individual and a population, respectively. Not only the individual but the population evolves under some conditions. Optimum piece arrangement is searched without destroying the constraint of the puzzle.

Figure 3 shows the flow of GA. First, initial populations are generated by randomly assembling $N \times M$ pieces. Optimum piece arrangement is searched efficiently by evolving multiple populations. Second, offsprings are generated by exchanging two partial piece arrangements. Multiple offsprings are generated from a population. Third, fitness values for each generated population ($N \times M$-piece puzzle) are calculated, and populations for next generation are selected. Offsprings are generated again from those surviving populations. This process is repeated for a given number of generations.

2.1. Definition of individual and population

We define the population as $N \times M$ piece arrangement. Multiple populations are used in a GA. In a population, $n \times M$ and $N \times m$ partial piece arrangements are defined as row individual and column individual, respectively. Individuals include row individual and column individual. Figure 4 shows these definitions.

2.2. Fitness function

Fitness functions of two different types are calculated from one population in this method. One is the fitness function which is defined as difference values of the pixel values (binary data) on the border line of all neighboring pieces. The other is the fitness function which is calculated from ranking between neighboring pieces. Two fitness functions are calculated using pixel values of the border line of pieces. Two measures are used to keep variety of populations.

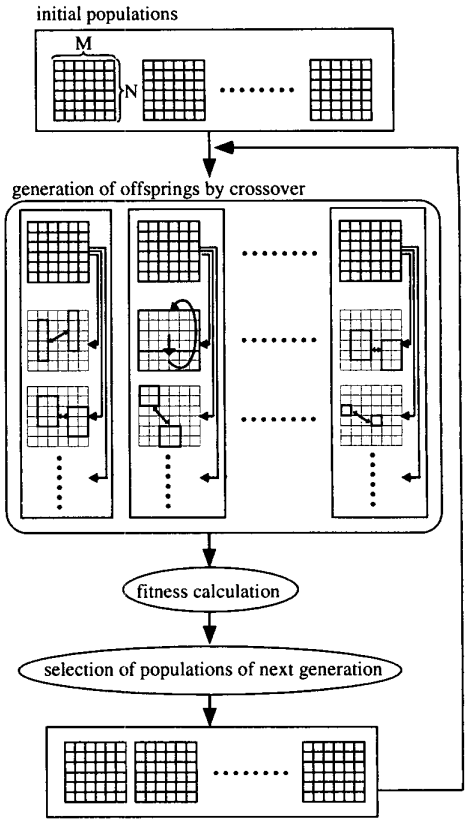

Figure 3. Optimization procedure using a GA.

2.2.1 Fitness function by using difference values of pixels

Distance between two pieces is defined as average absolute difference between every adjacent pixel pair on the touching border line of two pieces. Let $d_1(p, q)$ be the right direction distance between two pieces, where p is left piece, q is right piece. Let $d_2(p, q)$ be the bottom direction distance between two pieces, where p is top piece, q is bottom piece. Similarly, let $d_3(p, q)$, $d_4(p, q)$ be left direction distance and top direction distance, respectively. These distances do not satisfy distance axiom.

Fitness function F by using difference values of pixels is defined as

$$F = 1 - \frac{1}{2NM - N - M}\left(\sum_{i=1}^{N}\sum_{j=1}^{M-1} d_1(p_{i,j}, p_{i,j+1}) + \sum_{i=1}^{N-1}\sum_{j=1}^{M} d_2(p_{i,j}, p_{i+1,j})\right) \quad (1)$$

where $p_{i,j}$ is a piece of *i-th* row and *j-th* column.

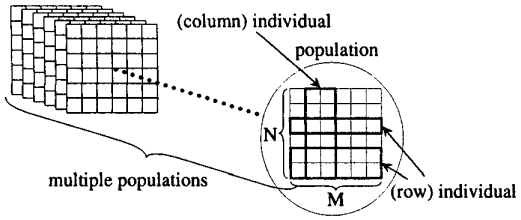

Figure 4. Definition of individual and population.

2.2.2 Fitness function by using ranking

Let a peace be p, let a set of the remaining pieces except piece p be Q. Right direction distance $d_1(p,q)$ are calculated, where $q \in Q$. $q \in Q$ is given ranking points in increasing order of distance values $d_1(p,q)$. Ranking point $r_1(p,q)$ is given in the following manner: first place, 10 point; second place, 9 point; third place 8 point; ... and 10th place, 1 point. Similarly, let $r_2(p,q)$, $r_3(p,q)$ and $r_4(p,q)$ be ranking points using bottom, left and top direction distance, respectively.

Fitness function R by using ranking is defined as

$$R = \frac{1}{2(2NM - N - M)} \times \left(\sum_{i=1}^{N} \sum_{j=1}^{M-1} (r_1(p_{i,j}, p_{i,j+1}) + r_3(p_{i,j+1}, p_{i,j})) + \sum_{i=1}^{N-1} \sum_{j=1}^{M} (r_2(p_{i,j}, p_{i+1,j}) + r_4(p_{i+1,j}, p_{i,j})) \right) \quad (2)$$

where $p_{i,j}$ is a piece of *i-th* row and *j-th* column.

2.3. Crossover

There are two types of crossover methods in this method. One is the 2-point crossover which is typically used in GAs. The other is self-crossover in which an offspring is generated from one parent. A type of crossover is selected randomly. Figure 5 shows examples of exchange of pieces by crossover.

2.3.1 2-point crossover

First, two row individuals or two column individuals are selected randomly. Then, the same piece does not have to be selected in two individuals. Selected two row (or column) individuals are of the same size, $m \times N (or M \times n)$. Next, the two crossover points are selected randomly, and chromosomes between the crossover points are exchanged. Then, the number of piece between the crossover points is same. Figure 5 (a) and (b) show examples of 2-point crossover.

2.3.2 Self-crossover

First, a row individual or a column individual is selected randomly. Next, two crossover points are selected randomly. As shown in figure 5(c),(d),(e) and (f), pieces are exchanged by the right (top) or left (bottom) direction piece rotation. The directions are decided randomly.

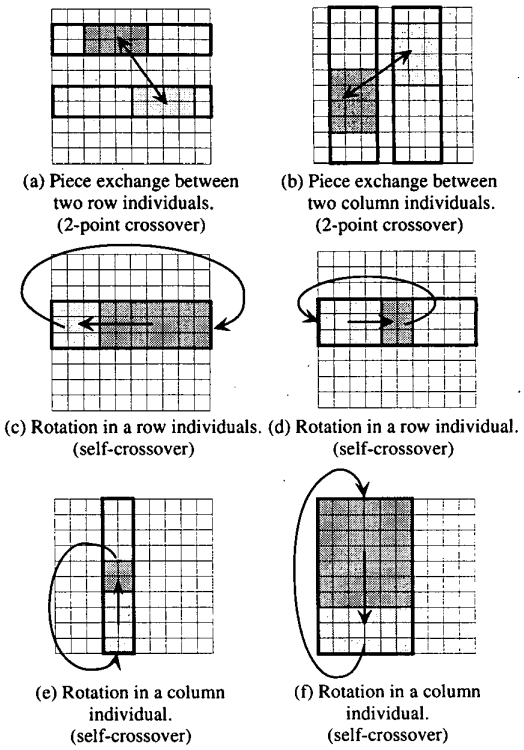

Figure 5. Examples of exchange of pieces by crossover.

2.4. Selection

Child populations are generated from one parent population. The number of children which are generated by crossover is 100 in the experiments. Populations of next generation are selected from these child and parent populations. The half populations of the number of next generation populations are selected in decreasing order of the value F of equation 1. The remaining half populations are selected in decreasing order of the value R of equation 2. The duplication of populations between the two is avoided.

The number of offsprings generated from same parent before k generation is limited to z. In the experiments, let k and z be 5 and 10, respectively. All individuals cannot survive over 5 generation. The above limitations are utilized to keep variety of populations.

3. Experiments

We made $N \times M$-piece puzzles by dividing binary images which were captured by the scanner. 8×8-piece puz-

(a) dormouse (b) squirrel (c) coffee

Figure 6. Input images (input puzzles).

zles were made by dividing input images (496×496 pixels). Let the number of populations be 200. The number of children which are generated from one parent is 100. Thus, a total number of children is 200×100. The highest population of the fitness function F is the optimum solution for 1,500 generations. We have done simulations using three types of 8×8-piece puzzles as shown in figure 6(a),(b) and (c). Correct piece arrangements was found in all the puzzles.

Table 1 shows the correct rate for 7×7, 8×8 and 9×9 piece puzzles which were made by dividing input images (figure 6(a),(b) and (c)). The number of successful trials and all ones is shown in parentheses of Table 1. The number of trials was 50 in each case. Input image sizes of 7×7, 9×9 piece puzzles are 497×497 and 495×495 pixels, respectively. Generation sizes are 1,000, 1,500 and 3,000 for 7×7, 8×8 and 9×9 piece puzzles, respectively. Thus, the total numbers of calculations of fitness function F are 20, 30 and 60 millions for 1,000, 1,500 and 3,000 generations, respectively. In case of 9×9 piece puzzle of figure 6(b), the correct piece arrangement was not found, but the correct piece arrangements was found in other cases. The number of piece exchanges is shown in figure 7. This number is average values of ten trials. When the correct piece arrangement was found, the process was finished. If the optimum solution could not be found in a given generation, the solution is searched from the beginning once more. The number of piece exchanges increases exponentially, see figure 7.

We have compared this method with the local search method. In this paper, local search method is that (1) $N \times M$ pieces are assembled randomly, (2) the pieces are exchanged when the fitness values F have increased at time of two pieces being exchanged, (3) the pieces are repeatedly exchanged until the fitness values F do not increase whatever piece may be changed, (4) and then the local solution is found. The above process is done repeatedly, and the optimum solution can be searched. The total numbers of piece exchanges was 5 billions in the local search method. In cases of all puzzles (7×7, 8×8 and 9×9 piece puzzles), the correct piece arrangements was not found.

As is known from the above, the propose method shows the capability of searching the optimum solution.

Table 1. Correct rates

No. of divisions	correct (%) (dormouse)	correct (%) (squirrel)	correct (%) (coffee)
7×7	100 (50/50)	100 (50/50)	100 (50/50)
8×8	94 (47/50)	72 (36/50)	92 (46/50)
9×9	62 (31/50)	0 (0/50)	44 (22/50)

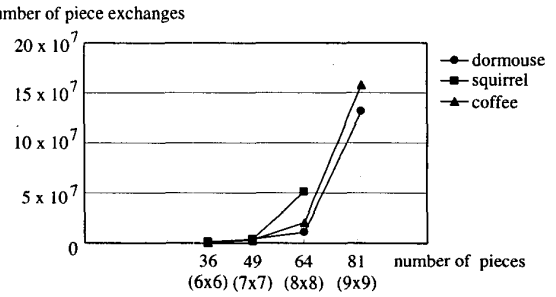

Figure 7. Computational complexity.

4. Conclusion

We have proposed the method for solving the rectangle piece puzzle assembly problem using a GA. Only binary data on the border line of the pieces were used in the puzzle assembly. We have defined the individual and the population as partial piece arrangement and total piece arrangement. By these definitions, the optimum piece arrangement is efficiently searched without destroying the constraint of puzzle. In this method, we have been able to assemble 8×8-piece puzzles, and have shown how our method is an efficient one. This approach can be applied to image mosaicking by modifying this method which can handle lacking pieces.

References

[1] R. W. Webster, P. S. LaFollette, and R. L. Stafford. Isthmus critical points for solving jigsaw puzzles in computer vision. *IEEE Trans. Syst., Man, Cybern.*, vol. 21, no. 5, pp. 1271-1278, 1991.

[2] K. Nagura, K. Sato, H. Maekawa, T. morita, and K. Fujii. Partial contour processing using curvature function – Assembly of jigsaw puzzle and recognition of moving figures. *Syst. Computing.*, vol.2, pp. 30-39, 1986.

[3] G. C. Burdea, and H. J. Wolfson. Solving jigsaw puzzles by a robot. *IEEE Trans. on Robotics and Automation*, vol. 5, no. 6, pp. 752-764, 1989.

[4] H. Freeman and L. Garder. Apictorial jigsaw puzzles: The computer solution of a problem in pattern recognition. *IEEE Trans. Electronic Comp.*, vol. EC-13, pp. 118-127, 1964.