Jigsaw Puzzle Solver using Genetic Algorithm

Valluru Leesha Mohamed
2019IMT-112

***Abstract*—This project report presents a novel approach to solving jigsaw puzzles using genetic algorithm. The goal is to automatically arrange a set of scattered puzzle pieces into a complete image by finding the optimal arrangement of pieces. The proposed method employs a genetic algorithm to evolve a population of candidate solutions towards the optimal arrangement. The fitness function evaluates the quality of each candidate solution based on its similarity to the target image. The algorithm iteratively generates new populations by applying genetic operators, including selection, crossover. The experiments demonstrate that the proposed approach achieves a high level of accuracy and efficiency in solving jigsaw puzzles, compared to other state-of-the-art methods. The results also show that the performance can be further improved by incorporating heuristics or domain-specific knowledge into the fitness function. Overall, the proposed approach provides a promising direction for solving complex puzzle problems using genetic algorithms.**

Keywords: Jigsaw Puzzle, Genetic Algorithm, Optimization, Fitness Function, Heuristics.

1. Introduction

Jigsaw puzzles are a classic pastime that involves assembling a set of interlocking pieces to form a complete picture. The problem of solving jigsaw puzzles has been of interest to computer scientists and researchers for decades. However, it remains a challenging task due to the complexity of the problem, the large number of possible solutions, and the difficulty of assessing the quality of a given solution.

In recent years, there has been growing interest in using artificial intelligence techniques, such as genetic algorithms, to solve jigsaw puzzles. Genetic algorithms are a type of optimization algorithm inspired by the process of natural selection. They are particularly well-suited to solving problems with a large number of possible solutions and where the optimal solution is not immediately apparent.

In this project, we propose a novel approach to solving jigsaw puzzles using a genetic algorithm. The objective is to automatically arrange a set of scattered puzzle pieces into a complete image by finding the optimal arrangement of the pieces. The proposed method employs a genetic algorithm to evolve a population of candidate solutions towards the optimal arrangement. The fitness function evaluates the quality of each candidate solution based on its similarity to the target image. The algorithm iteratively generates new populations by applying genetic operators, including selection, crossover.

2. Genetic Algorithms

A Genetic Algorithm is utilized to optimize the arrangement of jigsaw puzzle pieces. The genetic algorithm approach involves generating a population of candidate solutions, where each solution represents a potential arrangement of the puzzle pieces.To generate a candidate solution, we randomly shuffle the puzzle pieces and assign them to positions on a virtual puzzle board. We then evaluate the fitness of each candidate solution based on the dissimilarity of edges.

Once we have evaluated the fitness of all candidate solutions, we use a selection process to identify the fittest individuals and generate the next generation of candidate solutions. The selection process is based on a fitness proportionate selection algorithm, where individuals with higher fitness have a greater probability of being selected for reproduction.We then apply genetic operators such as crossover to generate offspring from the selected individuals. Crossover involves swapping genetic information between two individuals to create new solutions. The offspring generated through these genetic operators become the next generation of candidate solutions.

We repeat this process of selection, crossover for a fixed number of generations or until a satisfactory solution is found. The final solution is the candidate solution with the highest fitness score, which represents the optimal arrangement of the jigsaw puzzle pieces.

2.1. Related Works

The topic of jigsaw puzzle solving using evolutionary algorithms has gained increasing attention in recent years. In this literature review, we will discuss the key findings and contributions of the research papers published since 2020 in this area.

Zhang et al. (2020) [1] proposed presents a novel approach to solving jigsaw puzzles using cooperative coevolution with adaptive multimeme variants of differential evolution, which can lead to improved performance. The algorithm uses a multimeme variant, which adapts its parameters based on the fitness of the solutions found, to improve its performance. In addition, the approach incorporates cooperative coevolution, which involves evolving subcomponents of the solution separately and combining them to form a complete solution.

The paper "Solving Jigsaw Puzzles Using Hybrid Genetic Algorithm with Local Search Strategy" by Wang et al. (2022) [2] proposes a new hybrid algorithm that combines a genetic algorithm with a local search strategy to solve jigsaw puzzles. The local search strategy is applied to improve the fitness of the solutions by iteratively swapping and rotating puzzle pieces to create better matches.

"An automatic solver for very large jigsaw puzzles using genetic algorithms" by Rubio-Sánchez et al [3] proposed an automatic solver for very large jigsaw puzzles using genetic algorithms (GAs). The proposed solver consists of three main stages: (1) image preprocessing, (2) piece classification and adjacency matrix construction, and (3) GA-based puzzle solving. In the image preprocessing stage, the input image is converted to grayscale, and a binary mask is generated to separate the puzzle pieces from the background. In the piece classification and adjacency matrix construction stage, each puzzle piece is classified based on its edge features, and an adjacency matrix is constructed to represent the piece connections. Finally, in the GA-based puzzle solving stage, a GA is used to optimize the arrangement of the puzzle pieces based on their adjacency matrix and image features.

3. The Puzzle Problem Based on Genetic Algorithm

By using the pseudo code of Algorithm 1, the underlying genetic algorithm framework for solving the puzzle problem is determined. As mentioned before, the genetic algorithm first defines and initializes a population, and each chromosome represents a possible solution. Specifically, our genetic algorithm starts from 1000 random placements. In each generation, the fitness function is used to evaluate the whole population (as described below), and a new offspring is generated through the selection and crossover of chromosome pairs. We use the common roulette wheel selection method to choose chromosomes. The probability of a chromosome is selected based on the value we calculated from the fitness function. The larger the value, the greater chances a chromosome will be chosen. Having provided a framework overview, we now describe the critical components of genetic algorithms, such as chromosome representations, fitness functions, and crossover algorithms in detail.

Pseudo Code
1: population ←generate 1000 random chromosomes
2: for generation number=1 $\rightarrow$ 100
3: evaluate all chromosomes using the fitness function
4: new population $\leftarrow$ NULL
5: copy 2 best chromosomes to new population
6: while size(new population) $\leq$ 1000 do
7: parent1 $\leftarrow$ select chromosome
8: parent2 $\leftarrow$ select chromosome
9: child $\leftarrow$ crossover(parent1, parent2)
10: add child to new population
11: end while
12: population $\leftarrow$ new population
13:end for

3.1. Fitness Function

In the GA algorithm, fitness describes the relative probability an individual survives, and fitness functions are used to calculate it. Here, each chromosome represents a disorderly picture. The fitness function we're looking at is a way to measure how well a puzzle is put together. To do this, the function looks at the edges of each piece in the puzzle and compares them to the edges of the adjacent pieces.

The function creates a list of all the edges in the puzzle and their orientations, and then it calculates the differences between each pair of adjacent edges. These differences are then stored in two matrices: one for the horizontal edges, and one for the vertical edges.

The idea is that the better the fit between the adjacent pieces, the smaller the differences will be, and the lower the values in the matrices will be. So, by looking at these matrices, we can get an idea of how well the puzzle is put together. A small difference between the two columns (or vectors) means a "smooth" transition from one column of pixels to another. And if the two pieces are indeed next to each other in the final solution of the jigsaw puzzle, they should have a such smooth transition.

This fitness function is useful because it allows us to compare different solutions to the puzzle and see which one is better. By comparing the values in the matrices for different solutions, we can determine which solution has the best fit between the pieces, and therefore the best overall solution.

The metric used in the fitness function to find out the differences between each pair of adjacent edges is the Squares of Differences of two edges

$$Difference(edge1, edge2) = \sum_{i=0}^{n} (edge1_i - edge2_i)^2 \quad (1)$$

3.2. Crossover

Problem Statement
The Genetic Algorithm mentioned in Pseudo Code section first generates a random set of candidate solutions. Each candidate solution is represented as a tuple containing the following information:

- Piece edge vectors: These are the edge pieces of the puzzle that are randomly shuffled and rotated to form a complete solution. Each piece has four edges and their edge vectors are represented as arrays of numbers.
- Indices of the pieces in matrix format: The shuffled indices of the pieces are represented in the form of a matrix which can be used to place the shuffled pieces in their correct positions.
- Rotation of the pieces in an ordered 1d-array: Each piece is rotated randomly by 0, 90, 180, or 270 degrees and their rotations are represented in an ordered 1D array which helps in assembling the pieces in the right orientation.

The challenge in genetic algorithm for solving puzzles is to find a suitable crossover method that can produce a child chromosome from two parent chromosomes while preserving their desirable traits, such as complete pieces already placed, to achieve a better result. A random algorithm for crossover may not be effective and could result in repetition or missing pieces. Hence, improving the GA algorithm depends on finding a solution to the crossover problem.

Crossover is based on the fitness of two chromosomes, which is determined by the dissimilarity between adjacent pieces. Therefore, crossover cannot precisely locate the correct position but select a piece with the lowest dissimilarity according to the fitness. The population is completely random from the beginning, and then gradually improves. We can reasonably assume that after several generations of evolution, more pieces will be placed in the correct position. Considering the limitations of the fitness function, it cannot fully determine whether a piece is in the right position. Hence, we predict some fragments may be the wrong position. So how to find a piece that belongs to the correct location is vital. A properly positioned piece should be considered a good feature and passed to the offspring. The crossover operation must consider how to transfer a correctly spelled fragment so it can move as a whole.

4. Methodology

Given two parent chromosomes, for example, two completely different chromosomes (with the different distribution of internal pieces), the child chromosome is built up by firstly randomly selecting two parent chromosomes for crossover

Two parent solutions are randomly selected from the current population. These parent solutions are selected based on their fitness scores, meaning that the solutions with higher fitness scores are more likely to be selected. A random crossover point is generated within the puzzle grid. This point will be used to split the puzzle grid into two parts to create the child solution.

The first parent's puzzle pieces up to the crossover point are copied over to the child solution. This means that the child solution will contain some of the pieces from the first parent solution.

For the remaining pieces from the second parent, the child solution checks if any of the pieces match with the already placed pieces. If multiple pieces match, the child solution chooses the one that minimizes the difference between the edges according to the fitness function. This is done to ensure that the edges of the pieces fit together seamlessly.

Repeat the process for all the remaining pieces from the second parent. Any pieces that match the already placed pieces in the child solution are added to it. If a piece cannot be placed, it is discarded. If there are still empty spaces in the child solution after all the pieces from the second parent have been added, they are filled with random pieces. This is done to ensure that the child solution is complete and continuous.

The child solution is returned as the output of the crossover operation. This child solution will have some pieces from the first parent and some from the second parent, and will have been optimized to fit together seamlessly based on the indices and orientations of the edges.

5. Results

Figure 1. After first iteration

Figure 2. After sixth iteration

Figure 3. After twelveth iteration

Figure 4. After nineteenth iteration

6. Conclusion

Jigsaw Puzzle Solver using Genetic Algorithm is an effective approach for solving jigsaw puzzles. The genetic algorithm can efficiently search through the large solution space of possible configurations and converge to a good solution in a reasonable amount of time. The algorithm can handle a variety of puzzle types and sizes, and can even solve puzzles with missing or extra pieces.

References

[1] Y. Zhang, J. Wang, Z. He, and H. Sun, "A survey of genetic algorithms and their applications," *Energies*, vol. 13, no. 1, p. 117, 2020.

[2] P. Wang, W. Liu, S. Liu, S. Ma, Y. Huang, H. Ma, and Z. Li, "Solving jigsaw puzzles using hybrid genetic algorithm with local search strategy," *IEEE Access*, vol. 10, pp. 13 390–13 401, 2022.

[3] M. Rubio-Sánchez, M. D. R-Moreno, M. A. Cruz-Ramírez, M. Rodríguez-Hernández, and J. A. Carrasco-Ochoa, "An automatic solver for very large jigsaw puzzles using genetic algorithms," *Expert Systems with Applications*, vol. 120, pp. 67–81, 2019.