Project Title

Genetic Algorithm-Based Solver for Very Large Jigsaw Puzzles

Team members:
Gunjan Gupta (2019111035)
Shivaan Sehgal (2018111026)
Vedant Mundheda (2018112006)
Akash Verma (2018101011)

TA Mentor:
Anushree Korturti

Team 20:
DIP CHICK

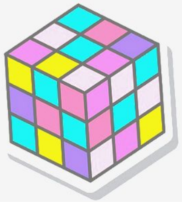
Repo URL:
https://github.com/Digital-Image-Processing-IIITH/dip-project-dip-chick

Introduction...

Jigsaw puzzles have been prevalent across the world for centuries. They are introduced to children to develop their improve cognition and visual-spatial reasoning. Solving such puzzles would have a greater impact in biology, literature, recovery of ancient/ shredded documents etc.

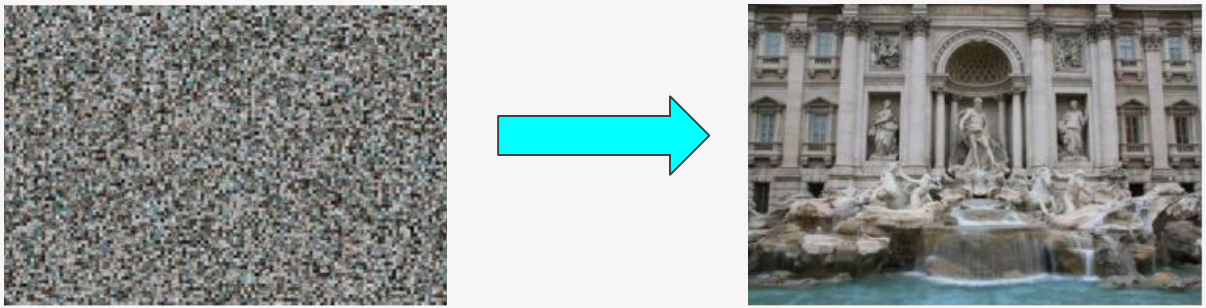
The aim of the paper is to effective automated, genetic algorithm (GA)-based jigsaw puzzle solver. Given n different non-overlapping pieces of an image, the goal is to reconstruct the original image, taking advantage of both the shape and chromatic information of each piece.

Problem Statement

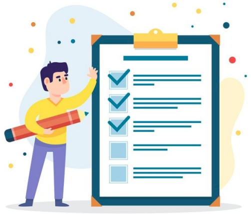


- To Solve the NP-complete problem of Jigsaw puzzle using Genetic Algorithms.
- We create a random arrangement of image pieces (square pieces of the image) and try to create the original image to check it’s accuracy.

We generate a jigsaw puzzle with any image and try to solve it using genetic algorithm, let us see how ...

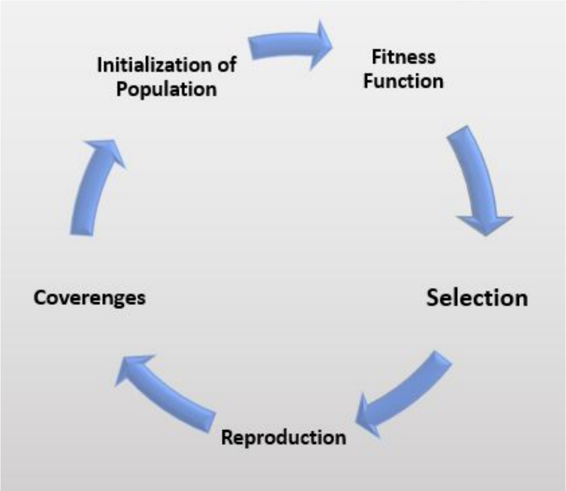


Procedure



The paper divides Genetic Algorithm based Jigsaw solving into different sections which are summarized in the following slides as follows

- Initialization of Population
- Fitness Function
- Selection
- Reproduction
- Converenges

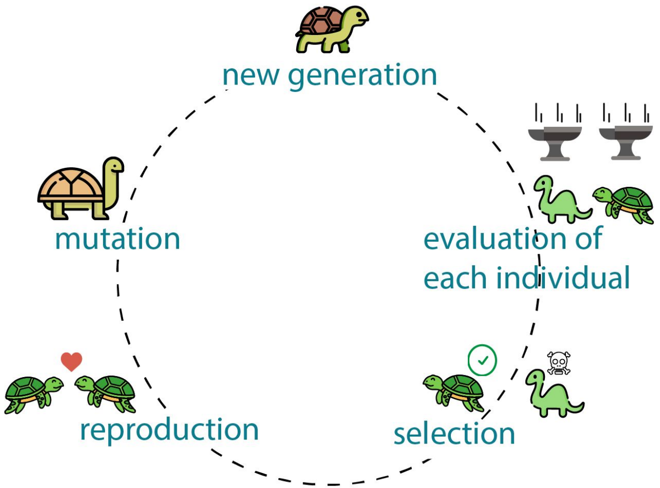


1. Genetic Algorithms

Genetic Algorithms

The genetic algorithm is a method for solving both constrained and unconstrained optimization problems that is based on natural selection, the process that drives biological evolution. The genetic algorithm repeatedly modifies a population of individual solutions.

Each generation consist of a population of individuals which represent a point in search space and possible solution. Each individual is represented as a string bits. This string is analogous to the Chromosome.



Genetic Algorithms Puzzle solver

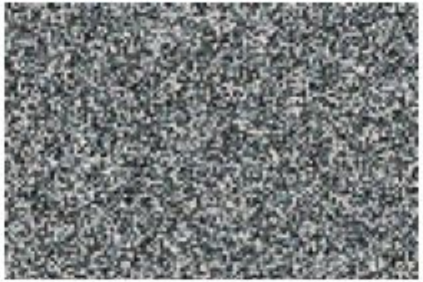
*It is the optimization algorithm inspired from the **theory of natural selection.***
The initial population will be randomly generated (random arrangement of puzzle pieces), then
we'll select some chromosomes to produce offsprings using fitness function. (Explained ahead)

Then the selected chromosomes will perform crossover to have offsprings with the traits of both the parents. Then,the children are mutated and then this become new population and this process is followed until the expected results(correct image) is not obtained.

Genetic Algorithms Puzzle solver

This is certain that it will give expected output because the fitness function will correctly detect chromosomes containing promising solution parts to be passed on to the next generations.

We’ll start with 1000 random chromosomes in the initial population.




2. Fitness Function

Fitness Function

The fitness function is evaluated for all chromosomes for the purpose of **selection.** In our GA, each chromosome represents a complete solution to the jigsaw puzzle problem i.e. a suggested placement of all pieces.

In this, we tend to predict the probability of 2 pieces to be adjacent. To achieve this, we'll find the dissimilarity measure between 2 pixels in the particular placement.

$$D(x_i, x_j, r) = \sqrt[2]{\sum_{k=1}^{K}\sum_{b=1}^{3}(x_i(k, K, b) - x_j(k, 1, b))^2}$$

x_j is considered to be at the right of x_i

The pieces are compatible if the dissimilarity is minimum.

Fitness Function

Continued

The fitness function is chosen to be of **run-time cost** and be computationally -inexpensive for one chromosome.
The fitness function of a given chromosome is the **sum of pairwise dissimilarities over all neighboring pieces** (whose configuration is represented by the chromosome)

$$\sum_{i=1}^{N}\sum_{j=1}^{M-1}(D(x_{i,j},x_{i,j+1},r))+\sum_{i=1}^{N-1}\sum_{j=1}^{M}(D(x_{i,j},x_{i+1,j},d))$$

Final Fitness function

A Chromosome is represented by an (N × M) matrix, where a matrix entry xi,j (i = 1 ..N, j = 1 ..M) corresponds to a single puzzle piece.

$D(x_i, x_j, r)$ Dissimilarity

r Right

d Down

3. Representation and Crossover

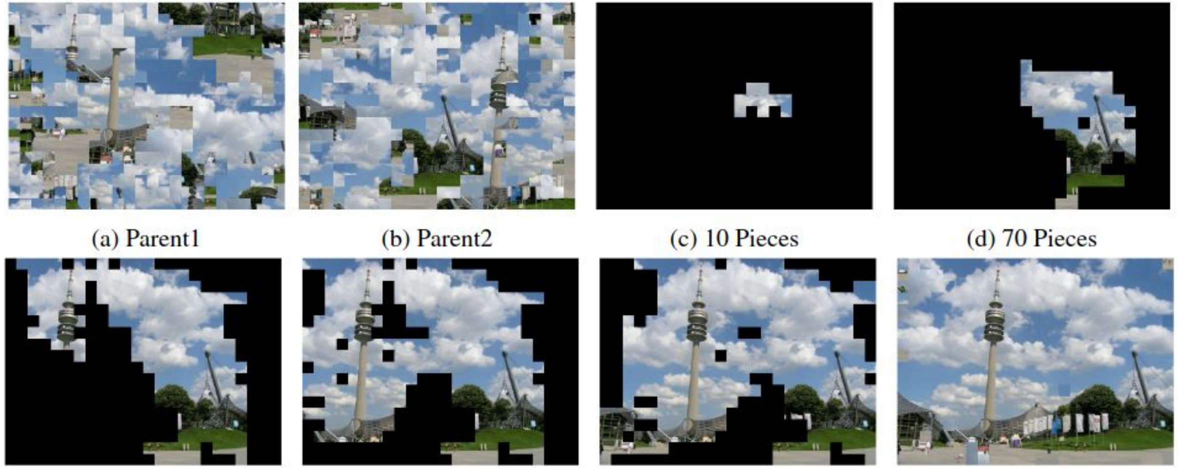
Redefining the Problem

Given a puzzle (image) of (N × M) pieces, a chromosome may be represented by an (N × M) matrix, each entry of which corresponds to a piece number.
The aim is to find a good crossover which transfers “good traits” to the children.

The fitness function is incapable of rewarding a correct position, hence the segments might appear most likely in incorrect absolute positions. This is one “good trait” that should be passed on to the child chromosome.

```
population ← generate 1000 random chromosomes
for generation_number = 1 → 100 do
    evaluate all chromosomes using the fitness function
    new_population ← NULL
    copy 4 best chromosomes to new_population
    while size(new_population) ≤ 1000 do
        parent1 ← select chromosome
        parent2 ← select chromosome
        child ← crossover(parent1, parent2)
        add child to new_population
    end while
    population ← new_population
end for
```

Crossover Operation



(a) Parent1 (b) Parent2 (c) 10 Pieces (d) 70 Pieces

(e) 180 Pieces (f) 258 Pieces (g) 304 Pieces (h) Child

Given (a) Parent1 and (b) Parent2, (c) – (g) depict how a kernel of pieces is gradually grown until (h) a complete child.

Note the detection of parts of the tower in both parents, which are then shifted and merged to the complete tower; shifting of images during kernel growing is due to piece position independence.

Proposed Solution

- Given 2 parent Chromosomes, the operator starts with a single piece and gradually joins other pieces at available boundaries

- New pieces may be joined only adjacently to existing pieces, so that the emerging image is always contiguous. The operator keeps adding pieces from a bank of available pieces until there are no more pieces left.

- We have to select pieces from the pieces bank and where to locate it in a child. Given a kernel, i.e. a partial image, we can mark all the boundaries where a new piece might be placed. A piece boundary is denoted by a pair (xi, R), consisting of the piece number and a spatial relation.

Three Phase Procedure

- Given all existing boundaries, the operator checks whether there exists a piece boundary for which both parents agree on a piece xj (meaning, both contain this piece in the spatial direction R of xi). If such a piece exists, then it is placed in the correct location. If the parents agree on two or more boundaries, one of them is chosen at random and the respective piece is assigned.

- If there is no agreement between the parents on any piece at any boundary, the second phase begins. To understand this phase, we briefly review the concept of a best-buddy piece, two pieces are said to be best-buddies if each piece considers the other as its most compatible piece.

- Finally, if no best-buddy piece exists, the operator randomly selects a boundary and assigns it the most compatible piece available. To introduce mutation – in the first and last phase the operator places, with low probability, an available piece at random, instead of the most compatible relevant piece available.

Best Buddies

Two pieces are said to be best-buddies if each piece considers the other as its most compatible piece. The pieces x_i and x_j are called buddy if

$$\forall x_k \in Pieces,\ C(x_i, x_j, R_1) \geq C(x_i, x_k, R_1)$$

and

$$\forall x_p \in Pieces,\ C(x_j, x_i, R_2) \geq C(x_j, x_p, R_2)$$

where Pieces is the set of all given image pieces and R_1 and R_2 are ”complementary” spatial relations (e.g. if R_1 = right, then R_2 = left and vice versa).

Crossover Operator

Algorithm 2 Crossover operator simplified

1: If any available boundary meets the criterion of Phase 1 (both parents agree on a piece), place the piece there and goto (1); otherwise continue.
2: If any available boundary meets the criterion of Phase 2 (one parent contains a best-buddy piece), place the piece there and goto (1); otherwise continue.
3: Randomly choose a boundary, place the most compatible available piece there and goto (1).

Rationale

- In a GA framework, good traits should be passed on to the child. Here, since position independence of pieces is encouraged, the trait of interest is captured by a piece's set of neighbors. Correct puzzle segments correspond to a correct placement of pieces next to each other.

- As each chromosome is a complete placement of all pieces, the good traits of parents should be passed on to the child. If both parents agree on a relation, we regard it as true with high probability.

Note that not all agreed relations are copied immediately to the child. Since a kernel-growing algorithm is used, some agreed pieces might "prematurely" serve as most compatible pieces at another boundary and be subsequently disqualified for later use. Thus, random agreements in early generations are likely to be nullified.

Rationale

- Since piece placement in parents might be random and since even best-buddy pieces might not capture the correct match, we combine the two. The fact that two pieces are both best-buddies and are adjacent at a parent is a good indication for the validity of this match.
- If each parent contains a correct segment and these segments partially overlap, the overlapping (agreed upon) part will be copied to the child in the first phase and be completed from both parents at the second stage, thus combining the segments into a larger correct segment, creating a better child solution, and advancing the pursuit of the entire correct image.

Concluding - As for the more greedy third step, the GA concurrently tries many different greedy placements, and only those that seem correct propagate through the generations. This exemplifies the principle of propagation of good traits in the spirit of the theory of natural selection.

4. Results Obtained

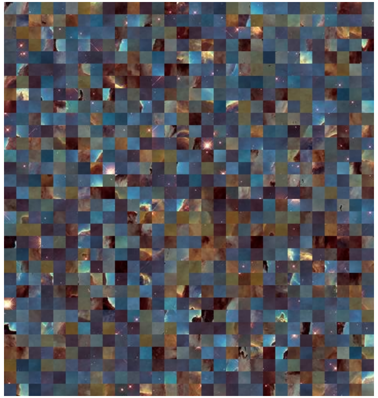
Results

The results obtained are with almost perfect. The genetic algorithm converged after few generations only. The image obtained are very similar to the original. The accuracy is calculated by the neighbour similarity rather than pixel similarity as the images could have been shifted but could have better accuracy than the pixel one.

The algorithm will work well with the images that don't have monotonous background. Having a monotonous background will confuse the algorithm and reduces it's accuracy.

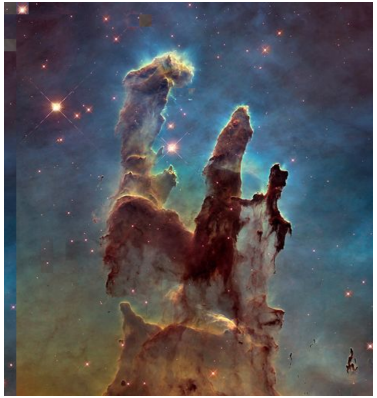
Sample Final Outputs

Generation 0

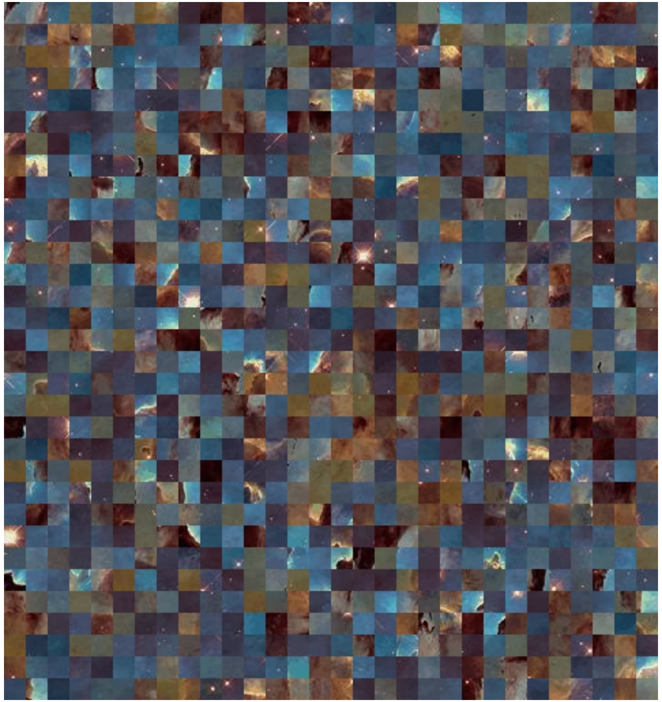


Generation 1

Generation 4



Final Generation



Original Image

Final Image

WORK DIVISION:

Theory :

Section 1.1 : Problem Formulation & Analysis

Lead : Gunjan

Section 1.2 :

1.2.1 : Lead : Gunjan

1.2.2 : Lead : Shivaan

1.2.3 : Lead : Vedant

Section 3.3 : : Shivaan

Implementation :

Section - 3.2:

3.2.1 : Main Algo : Shivaan Vedant Gunjan

3.2.2 : Lead : Shivaan

3.2.3 : Lead : Akash

Section 3.3 :

Lead : Akash

Presentation :

Lead 1 : Vedant

Lead 2 : Akash

Documentation :

Lead 1 : Gunjan

Lead 2 : Shivaan

Dataset :

Lead : Shivaan

References

[1] D. Sholomon, O. David and N. S. Netanyahu, "A Genetic Algorithm-Based Solver for Very Large Jigsaw Puzzles," *2013 IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 1767-1774, doi: 10.1109/CVPR.2013.231.


A *Picture* is worth a
Thousand words