NONOGRAM

Japanese crossword

grid-based logic puzzle

Presenters: Darbinyan L.
Hovhannisyan H.
Yeranosyan V.

Instructor: Stepanyan M.

American University of Armenia

11:11PM

nda • Agenda • Agenda • Agenda • Agenda • Agenda • Agenda • Agenda • Agenda • Agenda • Agenda • Agenda • Agenda • Agen

Tip: In order to have fun with this presentation, you can tap on anything that is underlined or is a mouse pointer, it will take you to the exact pages that you need

How: Magic ("մաջիք")

Agenda

Topics Covered

Start

Problem

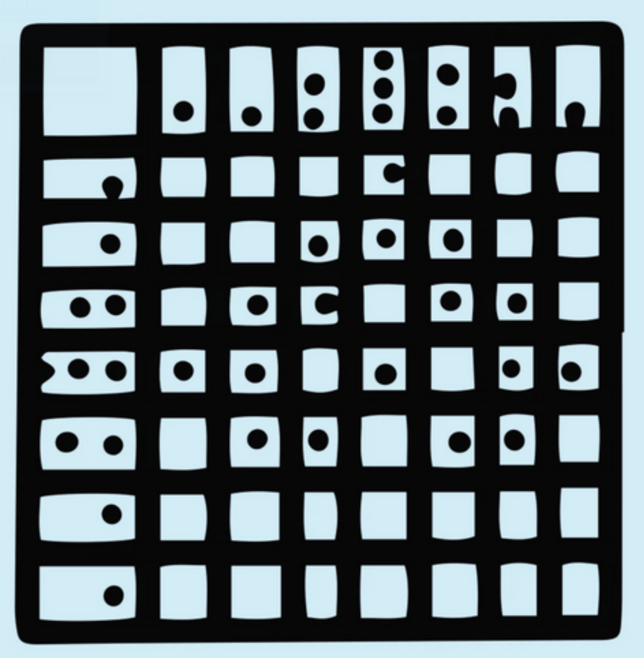
Literature

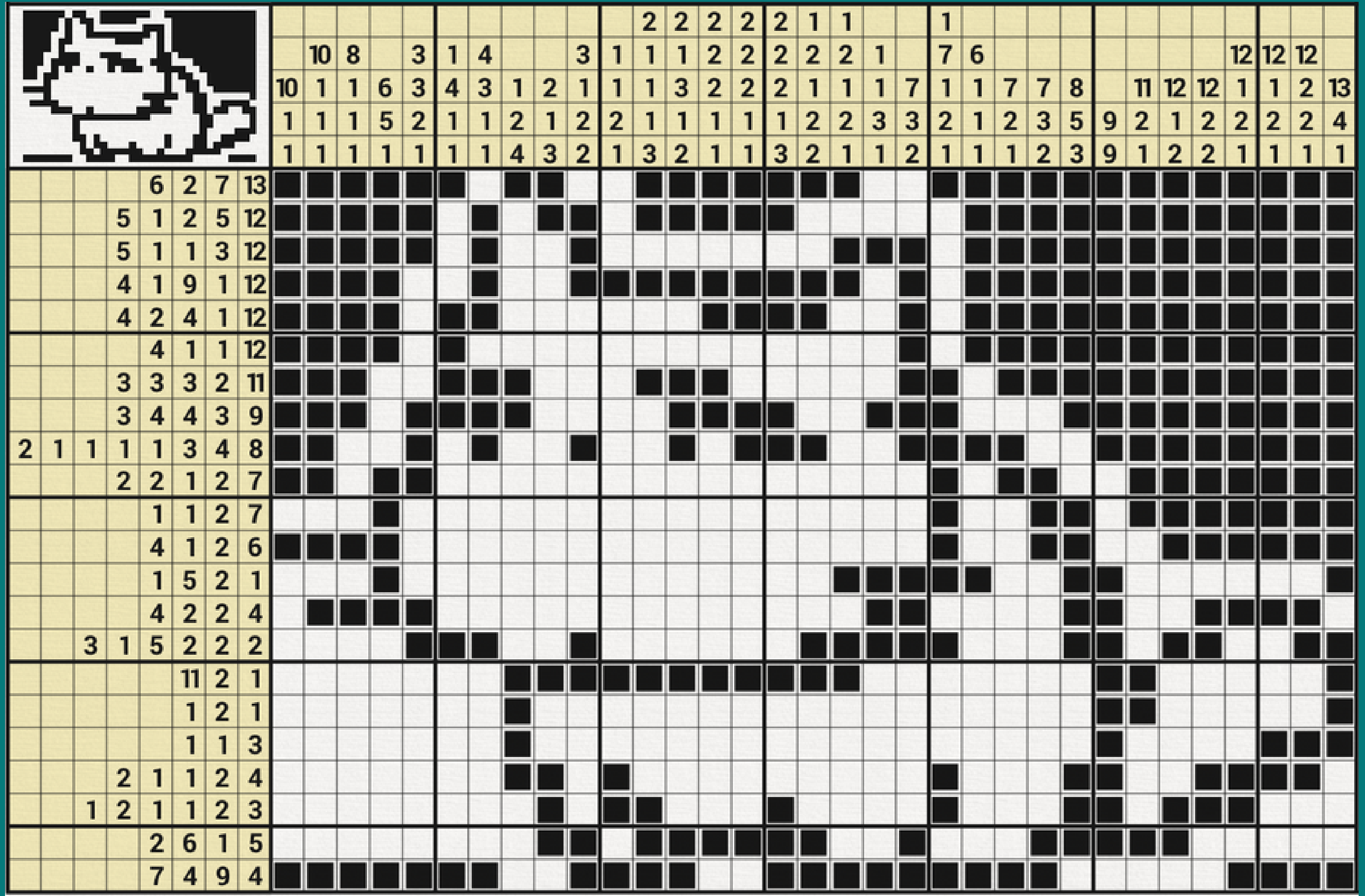
Methods

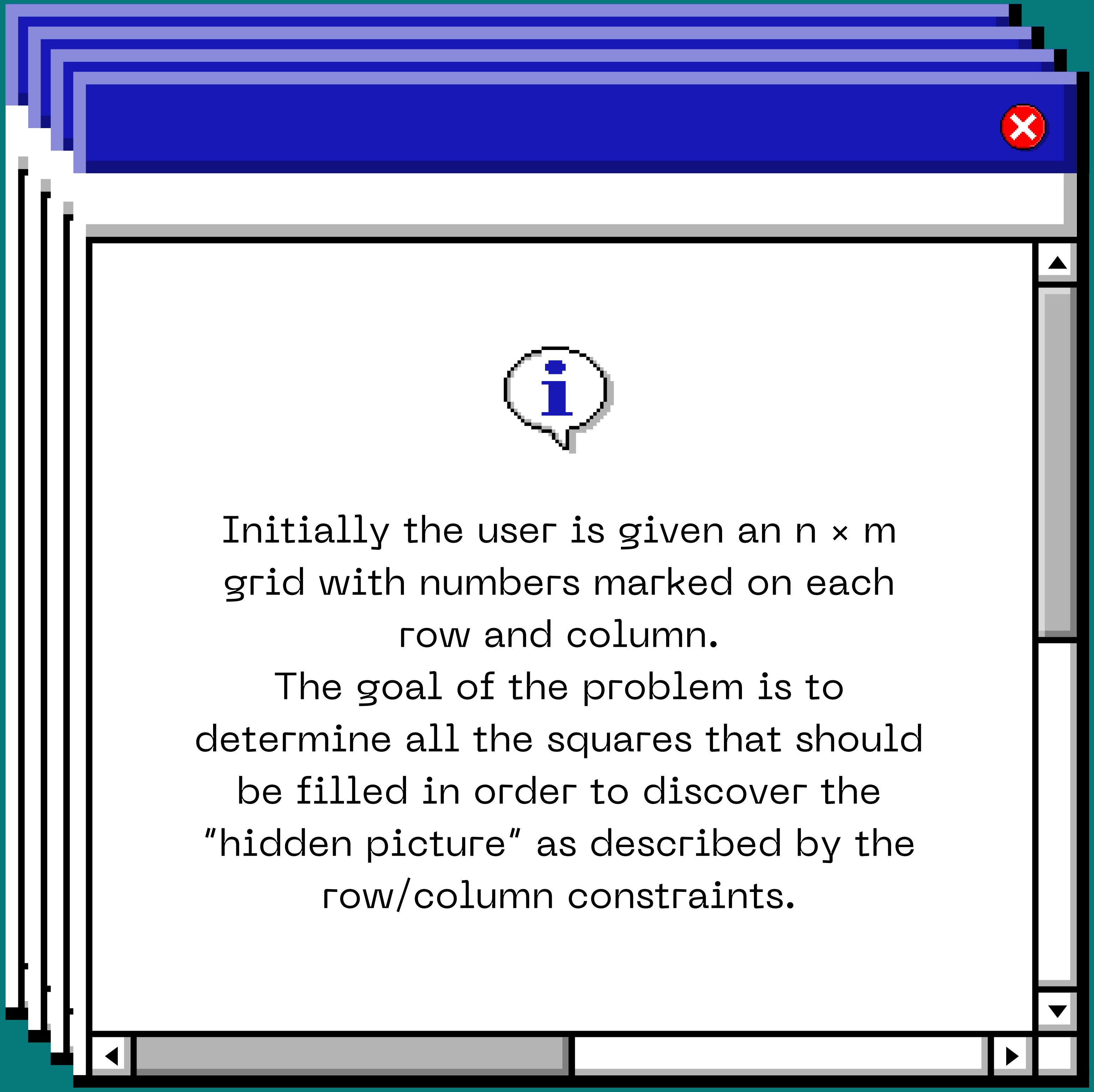
Results

PROBLEM

Nonogram is a grid-based logic puzzle.







Initially the user is given an n × m grid with numbers marked on each row and column.
The goal of the problem is to determine all the squares that should be filled in order to discover the "hidden picture" as described by the row/column constraints.

Back to Agenda Page

LITERATURE

NP-Complete problem tried to be solved by:

Converting Nonogram problem to Integer Linear Programming (ILP) problem

Definition of puzzle can be used by CPLEX

Can be used only for black and white nonograms

Simple Depth First Search (Brute Force) algorithm filling and emptying cells

Recursive brute force trying every conceivable configuration of cells

The recursive algorithm requires exponential time to implement

Developing a heuristic using Simulated Annealing (SA) to explore various search spaces

Starting with high temperature, error determined by adding up constraint violations

Not complete, can get stuck at local minimum when temperature is not high enough

Back to Agenda Page

METHODS

Constraint Satisfaction Problem (CSP)

Builds valid configurations of each row and tries to combine them to meet the column restrictions. Pre-processes the data to remove incorrect subtrees determined from previous iterations.

Genetic Algorithm (GA)

Generates a fitness function, starts running the algorithm on a random image, and optimizes it through the fitess function, takes into consideration the mutations and the cross over

Constraint Satisfaction Problem

Step 1:

Analyzing the layout of the board to identify the accuracy of the filling. The conjunction of the leftmost and rightmost complete run . From leftmost filling with the length of the first constraint till the last. For rightmost, the same from right to left. Taking the intersection of two runs to fill the squares.

Step 2:

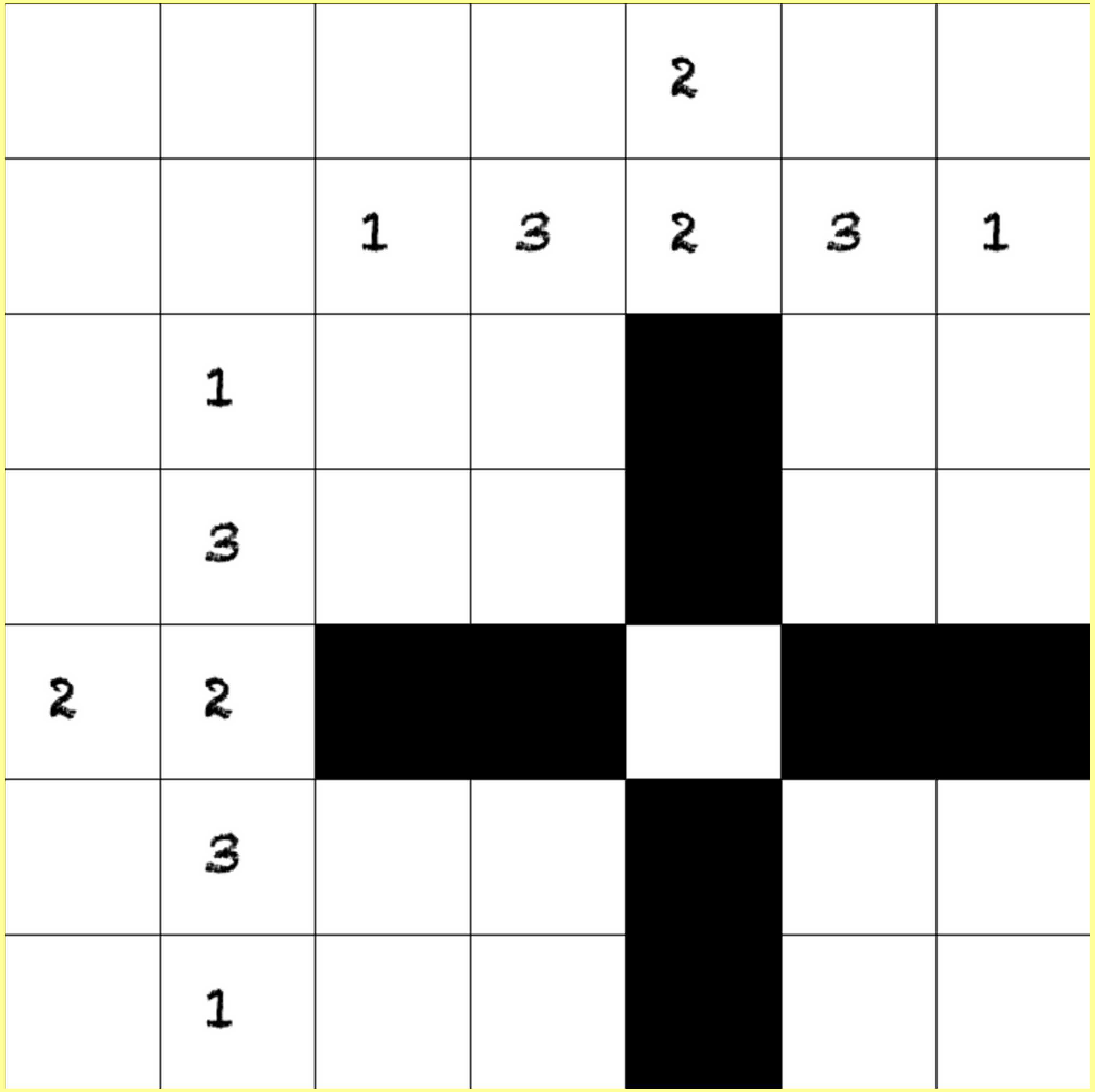
Creating all eligible row configurations by gradually increasing # of white spaces between runs till the last cell, calculating all combinations. The state space is built by the permutations. Check for node violation; if depicted, that node and its subtree are removed from the state space

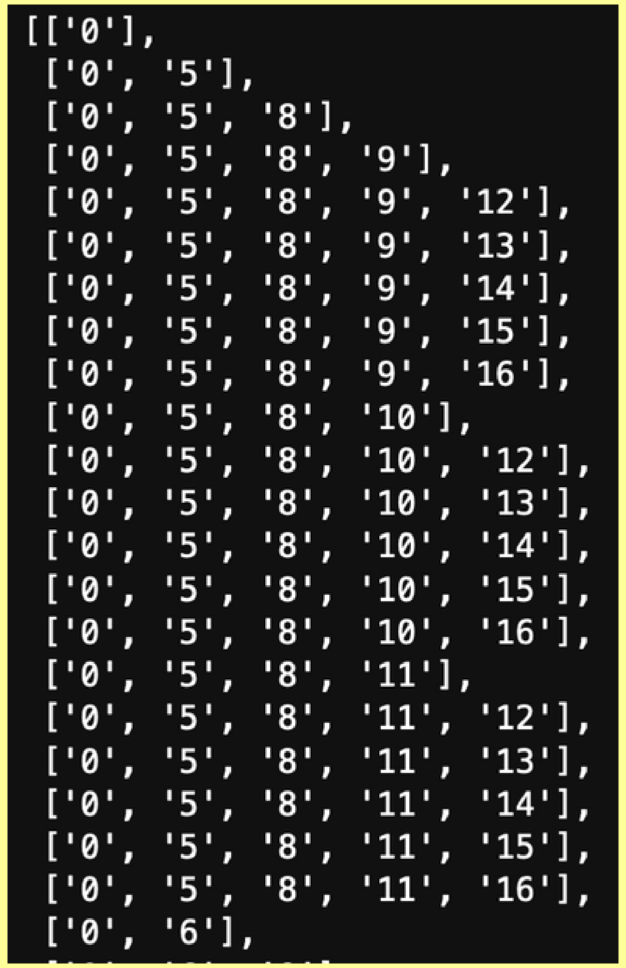
Step 3:

The solution from created tree is identified by a DFS-style backtracking search algorithm. The leaf nodes contain the goal state (define the configurations for last row). The algorithm keeps # of column constraints, and evaluates combinations of current, next rows. Prunes any subtree violating col constraint.

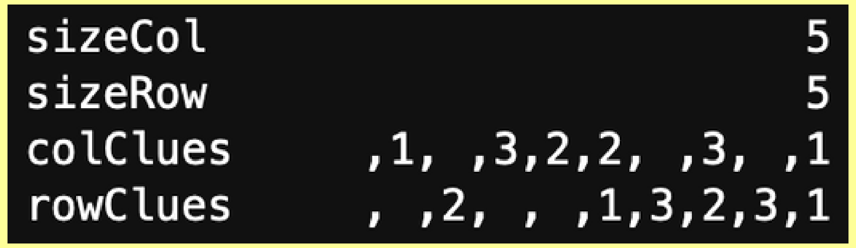
Back to Methods Page

Constraint Satisfaction Problem





```
[['0'],
 ['0', '5'],
 ['0', '5', '8'],
 ['0', '5', '8', '9'],
 ['0', '5', '8', '9', '12'],
 ['0', '5', '8', '9', '13'],
 ['0', '5', '8', '9', '14'],
 ['0', '5', '8', '9', '15'],
 ['0', '5', '8', '9', '16'],
 ['0', '5', '8', '10'],
 ['0', '5', '8', '10', '12'],
 ['0', '5', '8', '10', '13'],
 ['0', '5', '8', '10', '14'],
 ['0', '5', '8', '10', '15'],
 ['0', '5', '8', '10', '16'],
 ['0', '5', '8', '11'],
 ['0', '5', '8', '11', '12'],
 ['0', '5', '8', '11', '13'],
 ['0', '5', '8', '11', '14'],
 ['0', '5', '8', '11', '15'],
 ['0', '5', '8', '11', '16'],
 ['0', '6'],
```



sizeCol	5
sizeRow	5
colClues	,1, ,3,2,2, ,3, ,1
rowClues	, ,2, , ,1,3,2,3,1

Back to Methods Page

• Genetic Algorithm • Genetic Algorithm • Genetic Algorithm • Genetic Algorithm • Genetic Algorithm • Genetic Algorithm • Gene

METHODS

Constraint Satisfaction Problem (CSP)

Builds valid configurations of each row and tries to combine them to meet the column restrictions. Pre-processes the data to remove incorrect subtrees determined from previous iterations.

Genetic Algorithm (GA)

Generates a fitness function, starts running the algorithm on a random image, and optimizes it through the fitess function, takes into consideration the mutations and the cross over

A

Genetic Algorithm (GA)

Fitness Function

The fitness function or heuristic function determines how fit an individual is (the ability of an individual to compete with other individuals). It gives a fitness score to each individual. The probability that an individual will be selected for reproduction is based on its fitness score.

Crossover

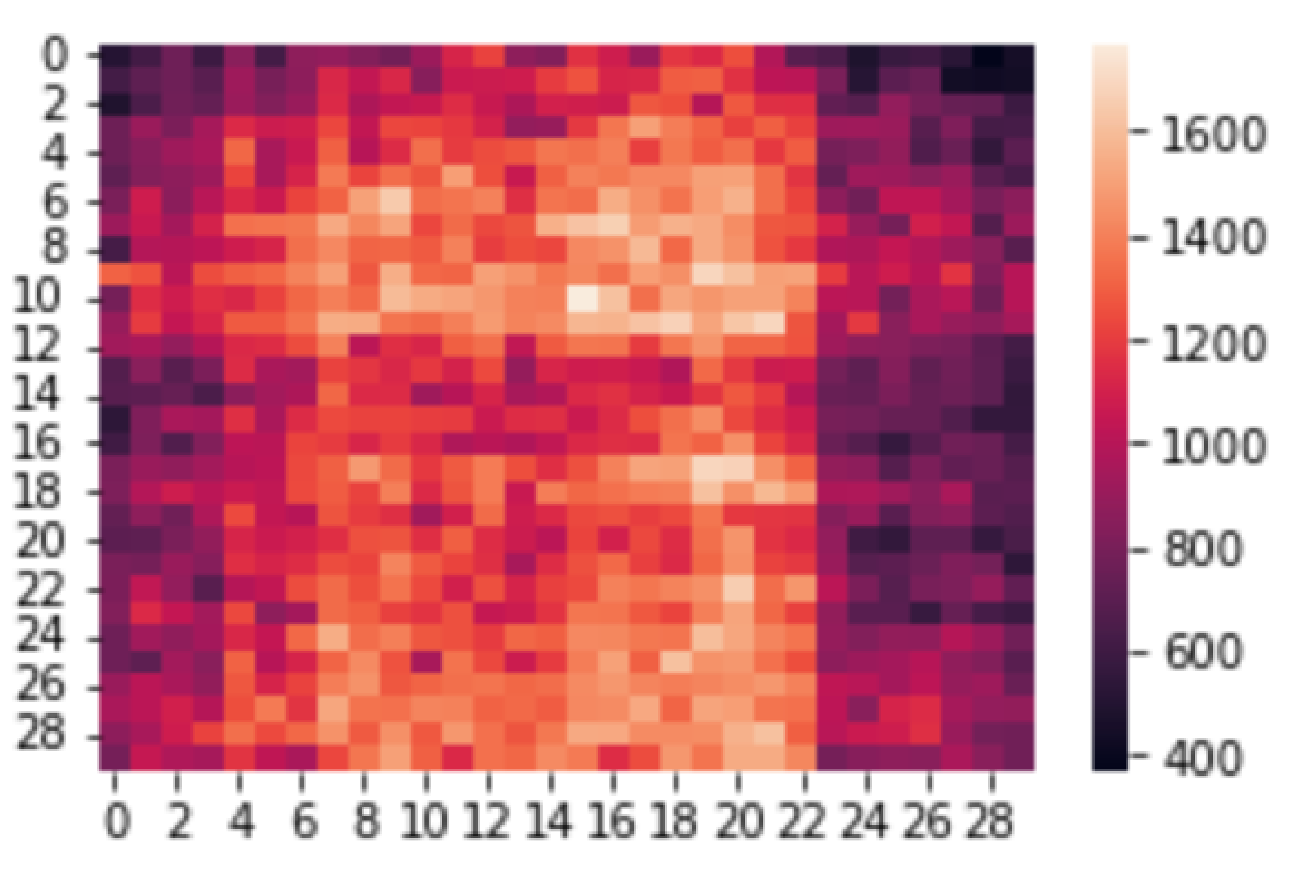
For each pair of parents to be mated, a crossover point is chosen at random from within the genes. Offsprings are created by exchanging the genes of parents among themselves until the crossover point is reached. The new offspring are added to the population.

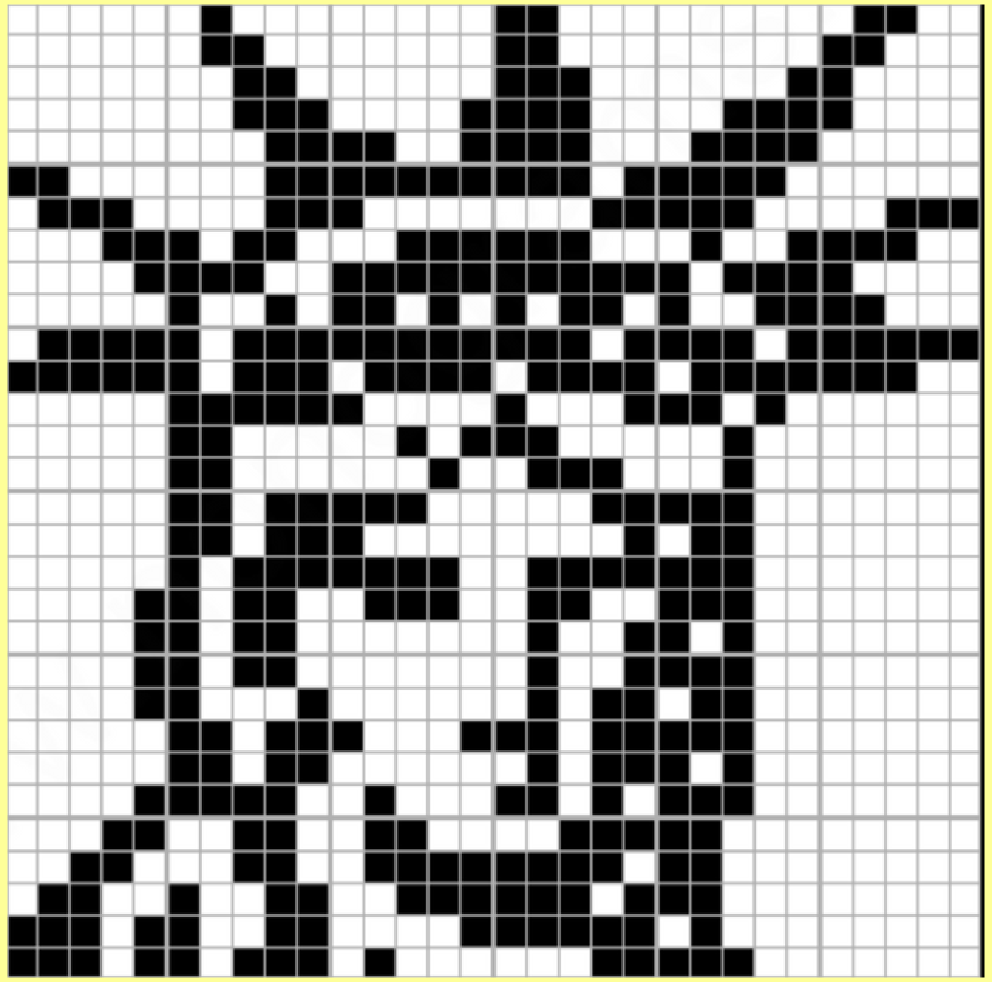
Mutation

In certain new offspring formed, some of their genes can be subjected to a mutation with a low random probability. This implies that some of the bits in the bit string can be flipped. Mutation occurs to maintain diversity within the population and prevent premature convergence.

Back to Methods Page

Genetic Algorithm (GA)





Back to Methods Page

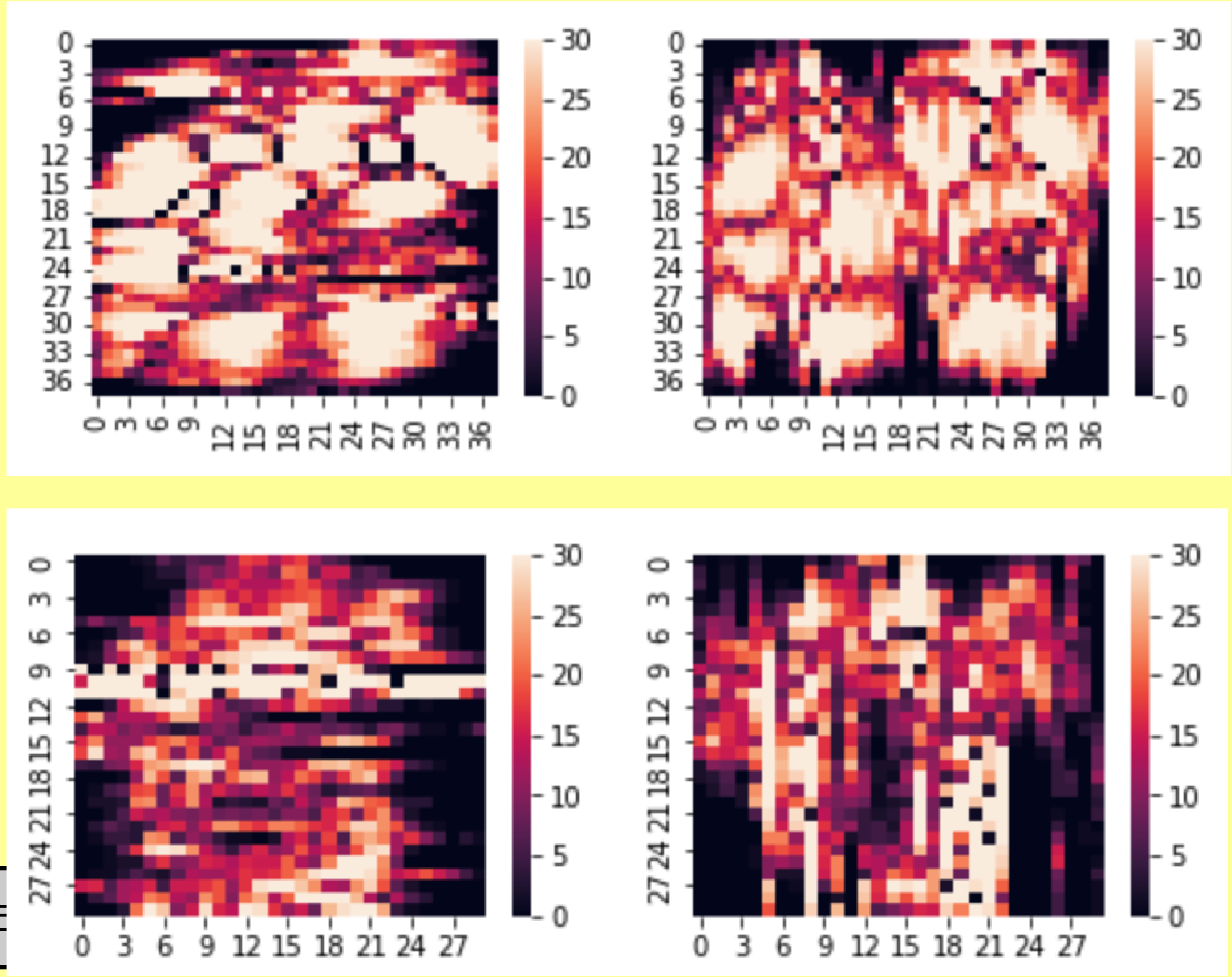
• Genetic Algorithm • Genetic Algorithm • Genetic Algorithm • Genetic Algorithm • Genetic Algorithm • Genetic Algorithm • Gene

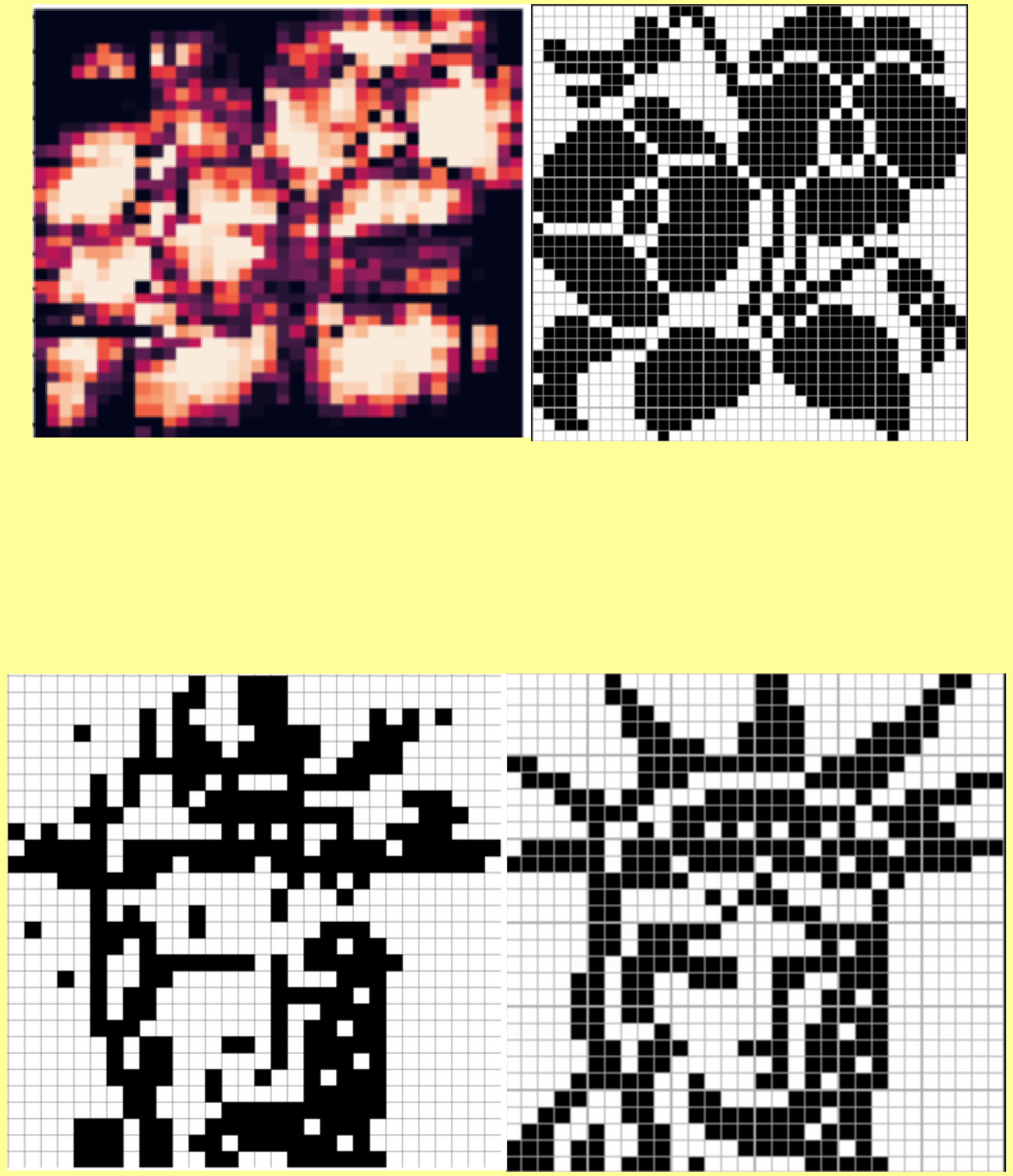
Creative Genetic Algorithm (CGA)

Row Agent

Column Agent

Solution





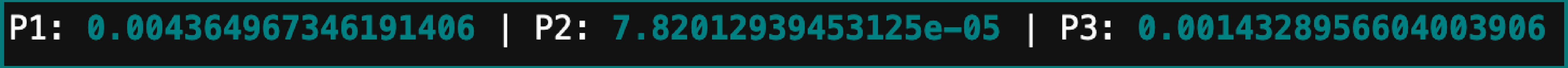
Back to GA F

Creative Genetic Algorithm • Creative Genetic Algorithm • Creative Genetic Algorithm • Creative Genetic Algorithm • Creativ

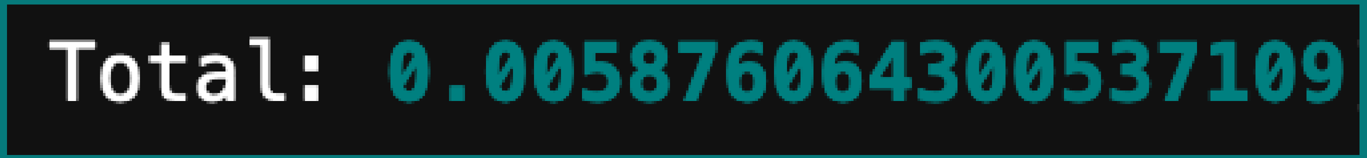
RESULTS

Runtime Test

CSP



```
P1: 0.004364967346191406 | P2: 7.82012939453125e-05 | P3: 0.0014328956604003906
```



```
Total: 0.005876064300537109
```

GA

Total: 136.0588545693465925

Back to Agenda Page

Thank you!

Get in the Game!