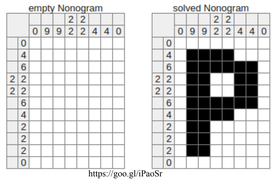
Automated creation of Nonogram puzzle game with Constraint Programming

University of St Andrews

Aigerim Yessenbayeva (ay63@st-andrews.ac.uk)
Supervisor: Dr. Chris Jefferson

INTRODUCTION

People solve a lot of decision-making problems in everyday life using the main logic of Constraint Programming without awareness of it, such as timetabling, scheduling and planning, traffic unloading. The most popular representation of the decision-making problem in academia is a puzzle game. Nonogram was chosen as one of the representation of the puzzle game in Constraint Programming. Nonogram or another name is Picross is one of the popular newspaper puzzle game.



PROBLEMS

1. There are so many problems in different areas, such as business, Internet traffic, scheduling, unloading concurrent processes on machines, which cannot be solved easily or represented mathematically. In those cases Constraint Programming can be helpful.
2. Recently it becomes more popular of measuring the difficulty of levels for the puzzle games, such as Chess and Sudoku. This project will try to extend it to new puzzle game (Nonogram)

GOALS

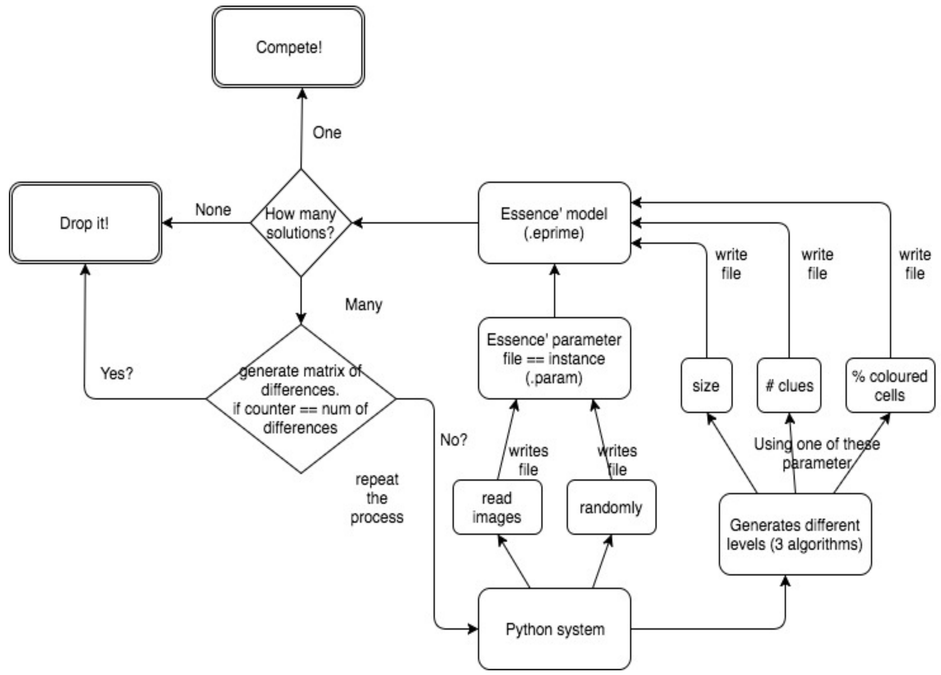
1. Implement solver for any level of Nonogram using Constraint Programming in Essence' language
2. Implement system which will provide instances for the solver in text file (.param extension)
3. Check whether the instance has one solution or many solutions or no solutions
4. Enhance the instance with many solutions to get one
5. Find and implement algorithms which can generate harder levels of the Nonogram
6. Produce the game itself
7. Compare the rates of difficulty between human and machine

DESIGN CRITERIA

•Implement the Constraint model in Essence' using Savile Row and Minion
• Based on the constraint model implement the rest of the system using Python
•For analysis we can use Python or R.

DESIGN & ARCHITECTURE DECISION

I. Whole process of the system

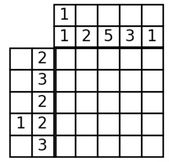


- Essence':
  - Essence' model implemented in .eprime file (using Savile Row and Minion).
  - Instances provided in text file with .param files.

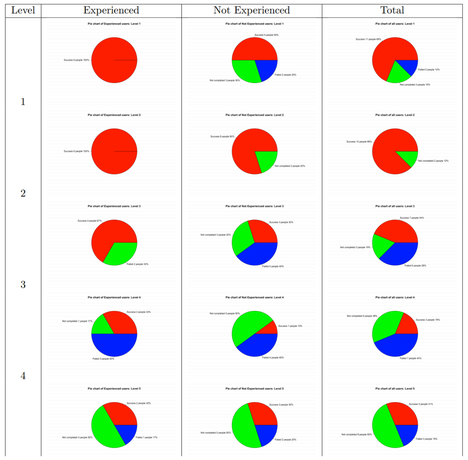
- Python:
  - Instances generation way: (1) reading images (2) random generation (3) create particular level of difficulty.
  - Instance might have three outputs:
    - (1) one solution – is ideal
    - (2) no solutions – too long or has too many solutions
    - (3) many solutions – can be improved to get one solution or dropped
      a. Compare 2 solutions only
      b. Compare all solutions
  - Three parameters for creating new levels of difficulty (hypothesis):
    - Size – the bigger size the more difficult level
    - Number of clues – more clues - more difficult
    - Percentage of coloured cells – more couloured cells the easier it is.

OUTPUTS & ANALYSIS

Possible outputs:



User Study results:



CONCLUSIONS

All objectives of the project were met. However, there could be done more research in trying to implement different other algorithms for generation of different levels of the Nonogram. As well to conduct more experiments with participants and get more precise analysis.