Game of Life
Presentation

20023064
Byung-guk Kim
Introduction

• Conway’s Game of Life
  – Simplest possible universe capable of computation
  – Rule
    • Dies if # of alive neighbor cells <= 2 (loneliness)
    • Dies if # of alive neighbor cells >= 5 (overcrowding)
    • Lives if # of alive neighbor cells = 3 (procreation)
    • Remains if # of alive neighbor cells = 4

• Possible rules to program the Game of Life
  – $3^9 = 19683$
Interestingness of Life Model

My choice to interesting life model

Convergence limit

\[ \text{# of cells of first stage} \times 0.2 < \text{# of cells of each stage} < \text{# of cells of first stage} \times 5 \]

Variety of cell position

\( \text{cell position of current stage} \neq \text{cell position of past stage} \)

The most interestingness

\( \text{maximum difference of cell position of each stage} \)
Approach to find interesting life model

First : Find interesting rules among every rules
Set initial state ( Set cell position of first stage )

- 20 generations by current rule
- Interestingness found ?
  - yes: Store current rule
  - no: Change rule sequentially

Change rule sequentially:
- Every rule check ?
  - no: no
  - yes: Every rule check

Second : Search the most interesting rule among stored interesting rules

Check the difference of cell position of each stage during 50 generations

Print out interesting rule and difference result

The most interestingness => maximum difference (variety of cell position of each generation)
Finding the most interesting rule

First search: Find interesting rules among every rules,
             Run 20 generations every rule

Second search: Search the most interesting rule among stored interesting rules,
              Run 50 generations every rule that is found in the first search

If interesting rule found in first search gets out of my function to interesting life
model in second search, the program prints out “Not Found” in the textbox.
The form of rule

Consists of 9 ternary variables

The number of alive neighbor
012345678
Example: 000210000
‘0’ means that cell dies
‘1’ means that cell remains
‘2’ means that cell lives (borns)

Lives if # of alive neighbor cells = 3

Remains if # of alive neighbor cells = 4
The extent of difference of cell position

Same cell position on several generations: *Not interesting*

Simple difference of cell position on several generations: *Not interesting*

Complex difference of cell position on several generations: *Interesting*
Modification for DEMO version

It takes long time to check all case of rules !!!

So, DEMO version only checks following rules.

# of cases of “cell remains” is one & # of cases of “cell lives” is one
Conclusion

I choose the sequential search to find interesting rule because in case of finite rules, sequential search requires less execution time than genetic algorithm.

The number of interesting life model depends on initial state.

Interesting events found by my function & certain initial state

=> Bomb, fireworks display