МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»**
Кафедра математической кибернетики и компьютерных наук

ОПТИМИЗАЦИИ ПОИСКА ПРИБЛИЖЕННЫХ РЕШЕНИЙ В ЭВОЛЮЦИОНИРУЮЩИХ КЛЕТОЧНЫХ АВТОМАТАХ

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студента 4 курса 451 группы
направления 09.03.04 — Программная инженерия
факультета КНиИТ
Григорьева Алексея Александровича

Научный руководитель
доцент _____ М. С. Семенов

Заведующий кафедрой
к. ф.-м. н., доцент _____ А. С. Иванов

Саратов 2020

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Клеточный автомат — дискретная модель, изучаемая теорией автоматов [1]. В общем случае, клеточный автомат состоит из регулярной решетки ячеек, множества состояний, множества входных сигналов и функции перехода. Клеточные автоматы применяются в компьютерном зрении [2], криптографии [3], компьютерной графике [4] и при моделировании различных процессов: физических [5], химических [6], биологических [7]. Одним из наиболее известных является клеточный автомат — игра «Жизнь» [8].

Существует огромное количество конфигураций клеточных автоматов. Меняя число возможных состояний, правила подсчета входных сигналов, возможные начальные состояния, получаются новые множества автоматов, перебор которых невозможен с текущими вычислительными мощностями. Поэтому для поиска клеточных автоматов, удовлетворяющих каким-либо условиям, используют алгоритмы нахождения приближенного результата. В недавних работах на схожую тему применялись генетические алгоритмы [9] и нейронные сети [10]. В выпускной квалификационной работе перебор осуществляется с использованием генетических алгоритмов.

В основе бакалаврской работы лежит выполнение экспериментов по нахождению приближенного решения клеточными автоматами. Цель каждого эксперимента — стабильное выполнение заданных условий ячейками, полученными после работы клеточных автоматов из множества начальных состояний. Начальное состояние задано случайно в каждой ячейке автомата. Для наглядности в качестве цели, которая будет поставлена перед началом экспериментов, будет воспроизведение заданного наперед целевого изображения (паттерна). Благодаря этому можно будет не только полагаться на полученные числовые значения, но и сопоставить результат с визуализацией в приложении. После каждого эксперимента должна сохраняться исчерпывающая информация о поведении клеточного автомата. По ней должно быть возможно определить конфигурацию автомата, генетического алгоритма, был ли удачным эксперимент и сколько было затрачено времени.

Эксперименты по оптимизации были проведены для двумерных клеточных автоматов первого порядка, с возможными состояниями 0 и 1. Следующее состояние ячейки клеточного автомата определяется 9 соседними ячейками на решетке включая данную или, другими словами, окрестностью Мура порядка

2 [11]. Тип автоматов выбран не случайно: количество всевозможных правил равно 2^{512} [12], что гарантирует невозможность перебора. Более того, не существует и общего аналитического решения для воспроизведения автоматами произвольного изображения, при условии что изначальная двумерная сетка задана случайными значениями. У выбранных типов клеточных автоматов также есть преимущества, связанные с возможностью оптимизации алгоритмов с использованием быстрых побитовых операций [13]. Многие из описанных далее результатов возможно перенести на случай клеточных автоматов N-го порядка [14], а также для клеточных автоматов с увеличенным количеством состояний или иными правилами переходов.

Целью бакалаврской работы являлось исследование возможностей по оптимизации поиска приближенного решения на основе паттернов двумерными клеточными автоматами первого порядка при помощи генетического алгоритма. Для достижения этой цели необходимо было:

1. разработать гибкое приложение, моделирующее работу клеточных автоматов с возможностью конфигурации генетического алгоритма;
2. добавить визуализацию клеточных автоматов и сбор статистики по экспериментам;
3. провести множество экспериментов с различными конфигурациями и целевыми изображениями;
4. сделать выводы, определить лучшие параметры генетического алгоритма, предложить возможные оптимизации.

1 Окружение для проведения экспериментов

В данном разделе приведено описание проекта и вспомогательных функций для моделирования и визуализации двумерных клеточных автоматов первого порядка.

1.1 Среда разработки

Программа для моделирования и визуализации клеточных автоматов выполнена в среде разработки `Unity 2019.2.21f`.

Программный код написан на языке `C#`. Основная логика экспериментов реализована в сцене MainScene. Весь проект подключен к системе контроля версий Git [15] на базе популярного ресурса для хранения версий Github.

1.2 Реализация клеточного автомата

Каждый клеточный автомат представляется ячейками на двумерной сетке. Для прозрачности процесса и наглядности результатов, эксперимент проводится полностью в реальном времени: в каждую единицу времени все ячейки автоматов на всех сетках переходят в новое состояние, определенное входными сигналами и правилами перехода.

Правила перехода описаны для двумерного клеточного автомата первого порядка. При подсчете входного сигнала в ячейке автомата учитываются состояния девяти ячеек в окрестности Мура порядка 2, для подсчета берутся значения с последней итерации. Взят самый простой случай: автомат может принимать значения 0 и 1. Таким образом, возможны $2^9 = 512$ вариантов входных сигналов.

В качестве визуализации клеточных автоматов в `Unity` использованы шейдеры. Плюс этого подхода заключается в том, что компьютер не будет тратить ресурсы оперативной памяти для визуализации автоматов, и вся нагрузка на это отдается видеокарте, находящейся большую часть времени в простое. Шейдеры позволяют создавать крупные визуализации огромной размерности благодаря эффективным вычислениям на видеокарте [16].

1.2.1 Правило перехода

Была создана простейшая реализация подсчета входных сигналов для ячеек в клеточном автомате. Для избежания лишних затрат памяти, в основной

части программы заранее созданы массивы для временного хранения следующих состояний ячеек автомата. Это необходимо, чтобы полученный результат не влиял на соседние ячейки в данный момент времени.

1.2.2 Оптимизированное вычисление входных сигналов

В вычислении следующего состояния каждого автомата используется 9 обращений к сетке. Более того, при каждом последующем обращении к сетке после первого, повторно считываются 6 состояний. Квадрат 3×3, окрестность Мура порядка 2, может быть представлен в виде окна, которое можно сдвигать вправо по сетке. Разбивая окно на 3 горизонтальные буфер-линии и переходя в новое состояние (аналогично сдвигу окна вправо), будут считываться лишь 3 новых бита справа. Добавление каждого бита в строку-буфер сопровождается побитовым сдвигом строки на 1 влево с отсечением крайнего левого бита. Сигналом для текущего автомата будет сумма верхней строки, средней строки с побитовым сдвигом влево на 3 и нижней строки с побитовым сдвигом на 6.

Результат аналогичен тому, что было получено в предыдущем разделе 1.2.1, но будет требоваться примерно в 3 раза меньше операций доступа.

1.2.3 Правило перехода на языке шейдеров (HLSL)

Обновление клеточных автоматов было полностью перенесено в шейдеры. Каждая сетка с клеточным автоматом использовует свою текстуру и шейдер (его параметры). В шейдере определена uniform-переменная: таблица переходов размерности 512 (все возможные состояния рассматриваемого клеточного автомата).

Была описана функция, возвращающая значение пикселя с заданным отступом в сетке относительно текущего клеточного автомата. Аналогично построению двоичного числа, последовательно, с левого верхнего угла, биты входных сигналов клеточного автомата в окрестности Мура порядка 2 определяют следующее состояние из таблицы переходов. Данный код выполняется в фрагментном шейдере [17].

2 Генетический алгоритм для поиска клеточных автоматов

Для получения приближенных результатов, был использован генетический алгоритм [18]. **Особи** — сетки с клеточными автоматами, **гены** — правила перехода для клеточных автоматов, **приспособленность** особи — суммарная по двумерной сетке степень совпадения каждой окрестности ячейки клеточного автомата некоторому целевому изображению (паттерну), воспроизведение которого и будет являться целью каждого эксперимента.

2.1 Подсчет приспособленности

Для правильной работы генетического алгоритма необходимо различать «полезные» особи от «ненужных». Для этого определим приспособленность особи. Приспособленностью особи считается как процент совпадения окрестностей клеток на сетке заданному перед началом эксперимента паттерну.

При подсчете приспособленности учитывается количество допустимых ошибок при сравнении с паттерном. Также учтено, что совпадение может быть найдено не с единственным изображением, а с целым набором.

2.2 Подсчет приспособленности на основе данных с GPU

Была предпринята попытка снизить нагрузку во время подсчета приспособленности клеточных автоматов на сетке за счет переноса обновления сетки с процессора на видеокарту. Так как визуализация клеточных автоматов с помощью шейдеров уже готова, достаточно считывать изображение перед каждым подсчетом приспособленности, а после считывания перевести пиксели цвета изображения в числа 0 и 1 и воспользоваться готовой функцией подсчета приспособленности.

Вопреки ожиданиям, это не ускорило программу. Это можно объяснить тем, что скорость передачи данных с видеопамяти в оперативную память крайне медленная [19].

2.3 Оптимизация подсчета приспособленности

Оптимизация, описанная в разделе 1.2.2, была использована и для подсчета приспособленности. Главное отличие заключается в том, что паттерн может быть произвольного размера. Для начала нужно перевести двоичные числа, образованные последовательным считыванием слева направа каждой

строки паттерна в десятичное число и сохранить эту информацию во вспомогательном массиве. Данные числа останутся неизменными. Далее, нужно провести сравнение битов сетки клеточного автомата и паттерна.

Имея два двоичных числа в десятичном представлении, можно получить новое двоичное число, которое будет состоять исключительно из различающихся битов в двух исходных числах. Для быстрого подсчета единиц был использован параллельный SWAR-алгоритм, с которым можно ознакомиться в источнике [20]. После подсчета разницы, для каждого числа-буфера, полученного для сетки, необходимо сделать побитовый сдвиг влево с выбрасыванием крайнего левого бита (используя бинарное «И») и добавить новое число справа.

Даже с учетом подобных и ранее описанных оптимизаций в разделе 1.2.2, данный этап остается самым высоко нагруженным. Поэтому для пользователя программы сделана возможность определить количество подсчетов приспособленности на каждом цикле эволюции перед этапом скрещивания.

2.4 Эволюция

Этап эволюции начинается после завершения подсчетов приспособленности клеточных автоматов на последних кадрах. На основании полученных данных будет отбираться половина особей с наивысшей приспособленностью, из них случайным образом будут созданы пары, каждая из которых создаст два потомка. Правила переходов для потомков формируются из генов родителей, которые также сохраняются до следующего этапа эволюции. В работе рассмотрены два вида скрещивания.

После скрещивания случайным образом будут мутируются гены автоматов, полученных после этапа эволюции. В случае клеточных автоматов с возможными состояниями 0 и 1, 0 меняется на 1 и наоборот. Возможны до X мутаций битов в одном автомате, а вероятность того что выбранный клеточный автомат мутирует — Y. Оба параметра настраиваются пользователем перед экспериментом.

2.5 Сбор данных и визуализация эксперимента

Перед началом эксперимента по поиску клеточного автомата, воспроизводящего паттерн, случайным образом будет создан идентификационный номер (ID) эксперимента. Название сопутствующих файлов со статистикой будет

содержать в суффиксе ID эксперимента. Во время исполнения эксперимента, можно наблюдать за графиком приспособленности и генофондом популяции.

После каждого эксперимента статистика записывается в файлы. Она содержит полную информацию о настройках и результатах (средняя приспособленность, время, количество итераций эволюции) эксперимента, максимальные и средние значения приспособленности на каждой итерации, опорные биты (см. раздел 3.1). Данные файлы расположены в папке:

`{Расположение_проекта}/Assets/SimulationData/`

После завершения эксперимента, в файл также записываются и правила переходов в виде генов. Файл расположенный в пути:

`{Расположение_проекта}/Assets/SimulationData/Genes/`

Каждый файл имеет название `G-{Имя_Паттерна}-{IDэксперимента}.txt`.

В нем ген каждого клеточного автомата записывается в новой строке, и, соответственно, количество строк в файле определяется числом клеточных автоматов в проведенном эксперименте.

Для просмотра клеточных автоматов после эксперимента создана дополнительная сцена в Unity с названием GeneVisualisation. В основном, она отличается от главной сцены тем, что в ней имеется одна большая сетка размерности 1024×1024 и отключена возможность эволюции автоматов. Для считывания файла с генами создан дополнительный скрипт. В редакторе необходимо выбрать автоматически созданный файл с данными генов. Запустив сцену, можно увидеть, как сетка меняет свое изначальное изображение, используя правила перехода из файла.

3 Эксперименты

В данном разделе были рассмотрены процессы проведения экспериментов для сбора статистики, с помощью которой в дальнейшем можно выявить зависимость между параметрами эксперимента и скоростью нахождения клеточных автоматов, удовлетворяющих заданным условиям. Перед выполнением экспериментов были поставлены следующие вопросы. Возможно ли найти такие параметры модели, которые для любого паттерна будут приводить к более быстрому нахождению клеточного автомата, моделирующего его?

Для задания целевых изображений был создан формат текстового файла, позволяющий попиксельно задать паттерны. После каждого эксперимента остается блок сообщений, описывающий полный информацию о результатах эксперимента: время работы, итоговый средний коэффициент приспособленности и другие.

3.1 Поиск закономерностей в таблице переходов

Было сделано предположение, что для любого правила клеточного автомата существуют «опорные» биты, являющиеся наиболее важными при формировании определенного паттерна. Было введено вспомогательное определение ценности бита. **Опорными** считаются те биты, значение ценности по модулю которых превосходит среднее значение (по модулю) ценности среди всех битов правила. Для того чтобы узнать, сохраняются ли опорные биты для повторных попыток найти правила, воспроизводящих заданный паттерн, необходимо найти «глобальные опорные биты». Все биты, глобальное значение ценности которых превышает половину числа подсчетов опорных битов, будут считаться глобальными опорными битами и отображаются на генофонде отдельным (бирюзовым) цветом.

Полученные результаты противоречат предположению. Не найдены зависимости и аналитические способы предугадать диапазон важных для паттерна битов. Все дальнейшие оптимизации нахождения паттерна были проведены за счет выбора «правильной» конфигурации эксперимента.

3.2 Описание проведенных экспериментов

В настоящей работе было проведено 1306 экспериментов над различными паттернами. Каждый паттерн дополняется похожими версиями себя, которые строятся отражением по вертикали или горизонтали исходного целе-

вого изображения или сменой цветов на противоположные (черный на белый и наоборот).

Большая часть паттернов имеет размер 5×5. Такой выбор объясняется тем, что клеточные автоматы для паттернов такого размера зачастую ищутся примерно за 300 этапов эволюции, что позволяет провести значительное количество экспериментов за относительно короткий промежуток времени и одновременно заметить разницу между различными параметрами генетического алгоритма. Так как в дальнейшем выводы о наилучшей конфигурации будут браться из статистических данных, необходимо провести большое количество запусков программы.

4 Анализ экспериментов

Данный раздел посвящен анализу данных, полученных после проведения экспериментов. Для каждого целевого изображения проводились эксперименты с различными настройками генетического алгоритма: количество клеточных автоматов, вероятность мутации, число мутирующих битов.

4.1 Обработка и визуализация данных

Для обработки и визуализации полученных данных создана программа, написанная на языке `Python` с использованием библиотеки `Matplotlib`. Программный код с комментариями написан в среде `Jupyter Notebook`, позволяющей создать наглядный и задокументированный программный код с сохраненным результатом выполнения. Файл в формате `.ipynb` хранится в репозитории. Он расположен в корне папки `Statistics`.

Данный «ноутбук» собирает и обрабатывает все файлы со статистикой по выбранному набору файлов.

4.2 Паттерн-зависимые результаты эксперимента

Было замечено, что чем больше ошибок допускается при сравнении с целевыми изображениями, тем больше диапазон возможных клеточных автоматов, воспроизводящих заданные паттерны. Тем самым, время поиска любой подходящей комбинации битов в генах снижается. Интересными оказались наблюдения, связанные с паттернами одинакового размера. Было замечено, что среднее время работы таких паттернов может сильно различаться. По полученным данным можно сделать предположение, что на скорость нахождения определенного паттерна влияет форма изображения.

В связи с высокой долей непредсказуемости результатов по разным паттернам, в работе сравниваются друг с другом лишь эксперименты, проведенные над одинаковыми целевыми изображениями. Для минимизации вероятности того, что с определенным паттерном «повезло», были проведены эксперименты с разными параметрами генетического алгоритма над несколькими целевыми изображениями.

4.3 Оптимальные параметры эксперимента

Важной составляющей для достижения наиболее быстрого нахождения клеточного автомата, воспроизводящего заданные целевые изображения, яв-

ляется подбор параметров для генетического алгоритма. В настоящей работе рассматриваются следующие основные параметры генетического алгоритма: процент мутации и максимальное количество бит, которые могут быть мутированы (выбирается случайное число от 1 до максимального).

Для проведения экспериментов были выбраны паттерны, приближенный результат для которых находится меньше чем за 500 итераций (полчаса реального времени). Это позволило провести большое количество экспериментов, тем самым проверив множество комбинаций процента мутации и количества бит. Получены результаты для четырех целевых изображений: «квадрат 3×3 с рамкой 1 пиксель», «елка 5×5», «крест 5×5» и «треугольник 7×5 в двух поворотах».

По результатам эксперимента были сделаны выводы о параметрах генетического алгоритма, дающих большую приспособленность. Вполне ожидаемо, что низкий процент мутации и мутация одного бита не дает достаточного разнообразия особей. Рассматривая высокие вероятности мутации и количество бит, можно заметить, что конфигурация генетического алгоритма, при которой процент мутации равен 25, а количество бит может достигать 8 или 16, дает наилучшую приспособленность среди всех других рассмотренных конфигураций генетических алгоритмов. Дальнейшее увеличение процента мутаций и количества бит не даст увеличения коэффициента приспособленности. Снижение процента мутации, наоборот, не создает достаточного разнообразия особей.

Данные закономерности прослеживаются для различных паттернов: разной размерности, сложности фигуры, количества допустимых ошибок. Можно быть уверенным, что данные результаты применимы к любому паттерну. Также были приведены результаты для некоторых других паттернов. Они подтверждают сделанные выводы.

В бакалаврской работе приведены дальнейшие возможные исследования.

ЗАКЛЮЧЕНИЕ

В результате дипломной работы было создано полноценное приложение для нахождения и визуализации двумерных клеточных автоматов первого порядка, воспроизводящих заданное целевое изображение. Данное приложение применимо для проведения экспериментов по подбору наилучших параметров для быстрого и эффективного (по значению приспособленности) поиска заданных клеточных автоматов. С помощью приложения были проведены эксперименты для различных параметров генетического алгоритма. На основе обработанных данных были определены оптимальные параметры генетического алгоритма. Оптимизации также проведены в коде программы, что привело к сокращению времени исполнения алгоритмов в несколько раз. Полученные результаты можно использовать и для других клеточных автоматов: высшего порядка, с большим числом состояний, с другим множеством начальных состояний.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 [Клеточные автоматы - реализация и эксперименты]. — URL: https://www.osp.ru/pcworld/2003/08/166226/ (Дата обращения 27.05.2020). Загл. с экр. Яз. рус.

2 *Rosin, P.* Edge Detection Using Cellular Automata / P. Rosin, X. Sun. — 2014. — Pp. 85–103.

3 *Zhu, B.-P.* Public-key cryptosystem based on cellular automata / B.-P. Zhu, L. Zhou, F.-Y. Liu. — 10 2007. — Vol. 31. — Pp. 612–616.

4 [Conway's Game of Life]. — URL: https://www.conwaylife.com/ (Дата обращения 27.05.2020). Загл. с экр. Яз. англ.

5 *Ladd, A.* An application of lattice-gas cellular automata to the study of brownian motion / A. Ladd, D. Frenkel, M. Colvin. — 01 1988. — Vol. 60. — Pp. 975–978.

6 *Oono, Y.* Discrete model of chemical turbulence / Y. Oono, M. Kohmoto // *Phys. Rev. Lett.* — Dec 1985. — Vol. 55. — Pp. 2927–2931. https://link.aps.org/doi/10.1103/PhysRevLett.55.2927.

7 *Sander, L.* Fractal growth processes / L. Sander // *Nature*. — 08 1986. — Vol. 322. — Pp. 789–793.

8 [Game of Life]. — URL: https://mathworld.wolfram.com/GameofLife.html (Дата обращения 27.05.2020). Загл. с экр. Яз. англ.

9 *Chavoya, A.* Using a genetic algorithm to evolve cellular automata for 2d/3d computational development. — Vol. 1. — 01 2006. — Pp. 231–232.

10 *Mordvintsev, A.* Growing neural cellular automata / A. Mordvintsev, E. Randazzo, E. Niklasson, M. Levin // *Distill*. — 2020. — https://distill.pub/2020/growing-ca.

11 [Moore Neighborhood]. — URL: https://mathworld.wolfram.com/MooreNeighborhood.html (Дата обращения 27.05.2020). Загл. с экр. Яз. англ.

12 *Packard N.H., W. S.* Two-dimensional cellular automata / W. S. Packard, N.H. // *Journal of Statistical Physics*. — 1985. — Vol. 38. — Pp. 901–946.

13 [Fundamentals of Computer Programming]. — Pp. 150–151. — URL: https://books.google.ru/books?id=xYgCAQAAQBAJ&pg=PA150&

lpg=PA150&dq=bit+operations+performance+c%23&source=bl&ots=FPdhyQlS5h&sig=ACfU3U3dz1jBOUOWD_0553muTAbXAM2jFw&hl=ru&sa=X&ved=2ahUKEwiH8M_WntnpAhXMyKYKHbohC3QQ6AEwDXoECAoQAQ (Дата обращения 27.05.2020). Загл. с экр. Яз. англ.

14 *Dennunzio, A.* On the dynamical behaviour of linear higher-order cellular automata and its decidability / A. Dennunzio, E. Formenti, L. Manzoni, L. Margara, A. Porreca // *Information Sciences*. — 02 2019. — Vol. 486.

15 [Git - What is Git?]. — URL: https://git-scm.com/book/en/v2/Getting-Started-What-is-Git%3F (Дата обращения 27.05.2020). Загл. с экр. Яз. англ.

16 [FLOPs per Cycle for CPUs, GPUs and Xeon Phis]. — URL: https://www.karlrupp.net/2016/08/flops-per-cycle-for-cpus-gpus-and-xeon-phis/ (Дата обращения 27.05.2020). Загл. с экр. Яз. англ.

17 [Rendering Pipeline Overview]. — URL: https://www.khronos.org/opengl/wiki/Rendering_Pipeline_Overview (Дата обращения 27.05.2020). Загл. с экр. Яз. англ.

18 [Global Optimization Algorithms]. — URL: http://www.it-weise.de/projects/book.pdf (Дата обращения 27.05.2020). Загл. с экр. Яз. англ.

19 *v. Werkhoven, B.* Performance models for cpu-gpu data transfers // 2014 14th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing. — 2014. — Pp. 11–20.

20 [Bit Twiddling Hacks]. — URL: https://graphics.stanford.edu/~seander/bithacks.html#CountBitsSetParallel (Дата обращения 27.05.2020). Загл. с экр. Яз. англ.