Программа для синхронизации углов с децентрализованным управлением

С. А. Кузьмин
СПбГЭТУ «ЛЭТИ»
KSA84@yandex.ru

Д. И. Шохалевич
СПбГЭТУ «ЛЭТИ»
dashashokhh@gmail.com

И. В. Герасимов
СПбГЭТУ «ЛЭТИ»
IVGerasimov-45@yandex.ru

Аннотация. **В работе рассматривается практическая реализация задачи Майхилла о синхронизации цепи стрелков без наличия центрального командира. В качестве примера была разработана программа для синхронизации двумерного массива объектов – углов поворота (имитирующих направление поворота стрелков с ружьями). В процессе синхронизации каждый объект "видит" только соседние с ним объекты и "договаривается" с ними о едином угле поворота. В программе учтена проблема возможного появления областей "зацикливания". Также она позволяет выбрать вариант визуализации процесса синхронизации и задать его скорость.**

Ключевые слова: задача Майхилла; синхронизация цепи стрелков; децентрализованное управление

I. Теоретическая часть

A. Задача Майхилла о синхронизации цепи стрелков

Задача Майхилла – классическая задача теории клеточных автоматов, сформулированная в 1957 году [1]. В ее основе лежит проблема согласования действий множества одинаковых элементов (на пример стрелков), которые могут обмениваться информацией только с ближайшими соседями. Необходимо разработать набор локальных правил, обеспечивающих одновременный переход всех элементов (стрелков) линейной цепи в специальное состояние ("выстрел"), причем алгоритм должен быть универсальным и не зависеть от длины цепи.

Каждый элемент в задаче представляет собой конечный автомат, который различает только свои соседние состояния. В начальный момент состояние всех элементов одинаково, за исключением крайнего слева, которому подается сигнал от командира. Требуется организовать локальное взаимодействие так, чтобы ни один элемент не перешел в конечное состояние преждевременно, а все сделали это строго одновременно.

Данная задача иллюстрирует фундаментальный принцип: глобально согласованное действие может быть достигнуто за счет исключительно локальных взаимодействий, без центрального управления. Этот принцип имеет прямую аналогию с моделируемым процессом синхронизации направлений на двумерной решетке, где элементы также обладают только локальной информацией, но постепенно приходят к единому состоянию.

B. Принцип децентрализованного управления

Принцип децентрализованного управления состоит в том, что система способна достигать согласованного поведения без единого управляющего центра. Каждый элемент опирается только на локально доступную информацию – состояния ближайших соседей, – и этого оказывается достаточно для формирования упорядоченной коллективной динамики.

Классический пример такого подхода приводится Д. А. Поспеловым в его статье «Оркестр играет без дирижера» [2], где согласованность возникает не за счет внешнего контроля, а благодаря взаимной координации участников. Подобные системы характерны для природы и техники: их аналоги встречаются в движении стаи птиц, работе муравейника, синхронизации светлячков, распределенных вычислениях и самоорганизующихся сетях.

Ключевая идея заключается в том, что глобальный порядок является следствием простых локальных правил. Именно этот принцип лежит в основе рассматриваемой модели: элементы двумерной решетки, взаимодействуя только с соседями, постепенно согласуют свои направления и приходят к устойчивому упорядоченному состоянию.

C. Основная идея программы

Основная идея работы – реализация задачи о синхронизации цепи стрелков на двумерной решетке на основе принципа децентрализованного управления [3]. То есть, сама по себе синхронизация происходит по соседним стрелкам без центрального командира.

II. Алгоритм

A. Предствление объектов

Рассматривается прямоугольное поле фиксированного размера, в каждой ячейке которого находится элемент ("стрелок"), ориентированный в некоторую сторону. Для упрощения визуализации, стрелок в программе будет обозначен кругом, а направление его ружья – отрезком (или стрелкой), исходящем из центра круга. То есть, фактически стрелок и его ружье будут условно представлены как "циферблат часов".

Предполагается, что каждый элемент может взаимодействовать только с ближайшими соседями по вертикали и горизонтали (рис. 1), не имея информации о состоянии далеких участков поля.

Угол представляется косинусно-синусным выражением вместо градусного для предотвращения некорректного усреднения. Значение по координате x представляется через cos(), а по координате y – через sin(). Такой подход помогает избежать ситуаций усреднений, к примеру, углов $350°$ и $10°$. В градусном

представлении усредненное значение будет 180°, в то время как в косинусно-синусном значение близкое к 0°.

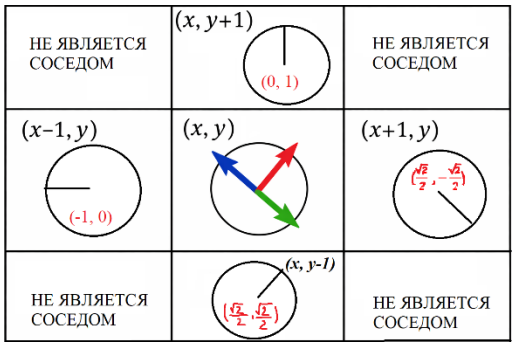


Рис. 1. Представление текущей клетки и ее соседей через координаты и значение угла (в косинусно-синусном представлении)

B. *Описание алгоритма*

В начале работы программы ориентации всех элементов задаются случайно – то есть поле полностью хаотично и не содержит никакой упорядоченной структуры.

Алгоритм моделирования представляет собой итерацию по всем элементам двумерного массива grid по горизонтали и вертикали. Находясь в текущей ячейке массива, он проверяет значения углов соседних с ней ячеек.

Синхронизация в программе происходит через механизм децентрализованного взаимодействия, где каждый элемент анализирует направления своих ближайших соседей и вычисляет среднее значение направлений векторов. После чего производится нормализация, которая приводит полученный вектор к единичной длине, сохраняя его направление. Это необходимо для устранения искажений, которые могут возникнуть при сложении векторов и корректировке направления.

Работа алгоритма строится на повторении обновления состояния для всех клеток до тех пор, пока разница значений всех соседних клеток не будет превышать установленной максимальной погрешности.

Для устранения проблемы возникновения "вихрей" (зацикленности) используется взаимная коррекция углов. Каждый элемент не только усредняет значения своих параметров, но и сохраняет часть своего предыдущего состояния:

```
for nx_, ny_ in neighbors:
    vx, vy = grid[ny_][nx_]
    tx = (1 - NEIGHBOR_PULL) * vx + NEIGHBOR_PULL * nx
    ty = (1 - NEIGHBOR_PULL) * vy + NEIGHBOR_PULL * ny
```

где NEGHBOR_PULL – коэффициент влияния соседей, который определяет, насколько сильно соседний элемент будет ориентироваться на текущее направление.

Критериями остановки цикла являются: параметр изменения, который отражает величину изменения системы за итерацию; параметр порядка, который определяется как длина результирующего вектора всех элементов и отражает степень глобальной согласованности системы.

III. Реализация программы

Разработка программы велась на основе языка программирования Python [4] в среде Visual Studio Code. Для визуализации окна приложения использовалась библиотека pygame [5].

A. *Интерфейс программы*

При запуске программы задаются начальные параметры окна 10x15 (рис. 2).

Пользователю предоставляются на выбор три режима работы программы:

- **NO COLOR** – отображение стрелков (в виде кружков) и углов их поворота (в виде стрелок), без отображения информации цветом.
- **ANGLE** – режим, в котором за каждым значением угла закреплен определенный цвет. То есть угол поворота отображается не направлением стрелки, а цветом.
- **ACTIVITY** – режим, в котором угол поворота отображается направлением стрелки, а цвет заливки круга меняется только в случае, если было изменено направление стрелка. Цвет не закреплен за направлением – он выбирается в зависимости от сравнения нового и старого значений угла поворота. Если угол уменьшился, то цвет становится синим, а если увеличился – красным.

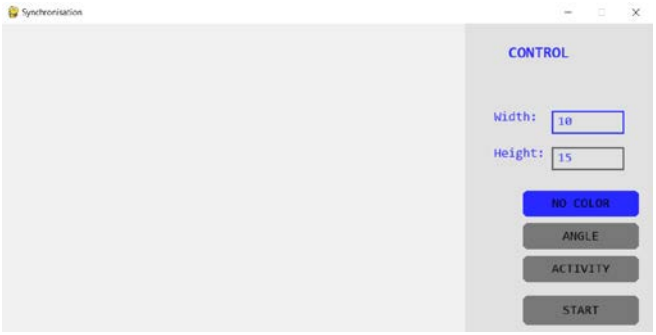


Рис. 2. Окно программы с заданными начальными параметрами поля

При запуске программы в одном из режимов в окне приложения появляются: счетчик итераций; критерий сходимости (Order R); статус, который отражает состояние программы в данный момент (RUNNING – система продолжает синхронизацию, SYNCED – система синхронизировалась). Параметр FPS позволяет выбрать скорость смены кадров (итераций) при отображении в окне приложения (регулируется от 1 до 30). Кнопки PAUSE и BACK отвечают за управление визуализацией. При нажатии на кнопку PAUSE программа приостанавливает работу с возможностью ее продолжения при нажатии на появившуюся кнопку RESUME. А при нажатии на кнопку BACK программа возвращается к начальному экрану приложения.

B. *Пример работы программы в режиме NO COLOR*

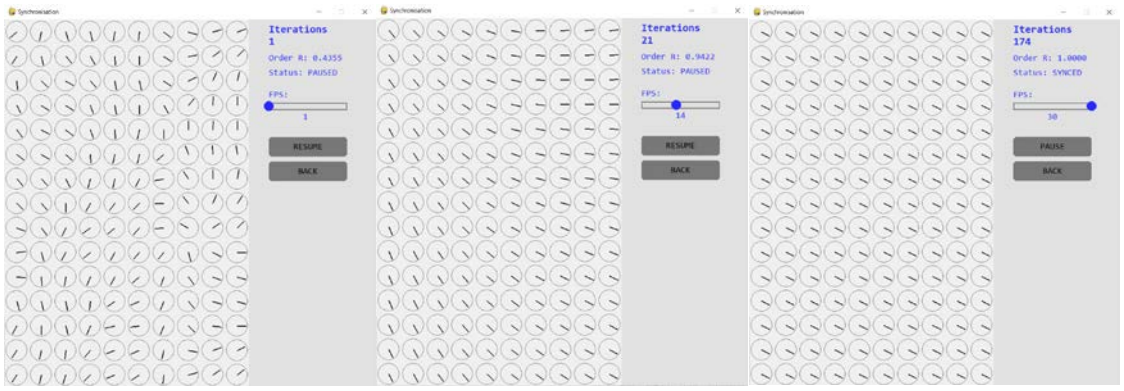
Начальное состояние углов поворота видно на рис. 3 а. Процесс синхронизации показан на рис. 3 б. Как видно на рис 3 в, в данном примере система стрелков достигла синхронизации через 174 итераций.

C. *Пример работы программы в режиме ANGLE*

В данном режиме направления стрелков отображаются не только направлением стрелки, но и цветом (рис. 4 а). При работе программы, можно наблюдать в каких областях массива уже произошли изменения поворота стрелков. На рис. 4 б представлена динамика изменения направлений спустя 21 итерацию. На рис. 4 в программа завершила работу через 205 итераций.

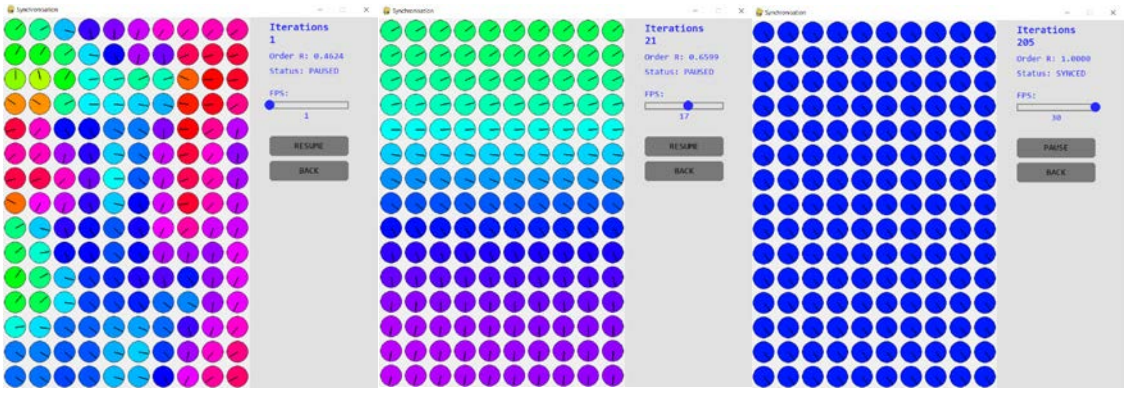
D. *Пример работы программы в режиме ACTIVITY*

В данном режиме изменение цвета стрелка происходит в момент смены им направления. Изначально все стрелки окрашены в желтый цвет (рис. 5 а). Если в процессе очередной синхронизации значение направления стрелка уменьшилось, он окрашивается в синий цвет. А если оно увеличилось – то в красный цвет (рис. 5 б). Тем самым, после выполнения 253 итераций видно, в каких областях больше всего менялось направление стрелка в ту или другую сторону (рис. 5 в).



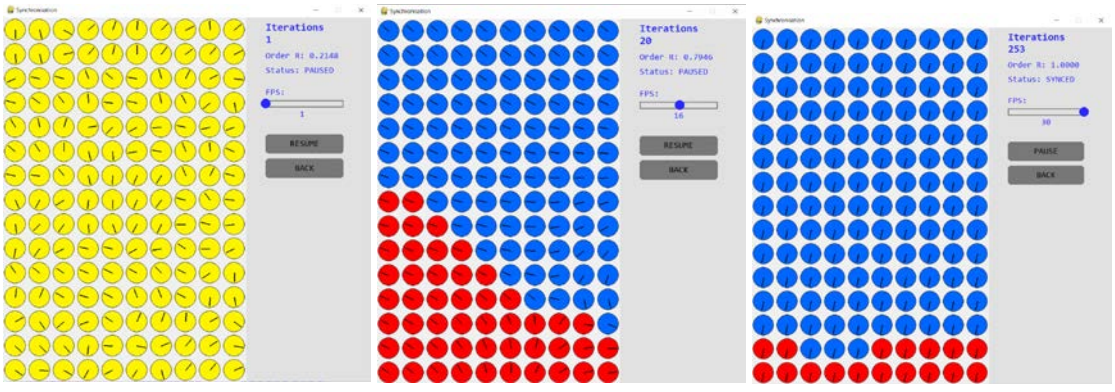
а) б) в)

Рис. 3. Работа программы в режиме NO COLOR (а – начальное состояние, б – промежуточное состояние, в – конечное состояние)



а) б) в)

Рис. 4. Работа программы в режиме ANGLE (а – начальное состояние, б – промежуточное состояние, в – конечное состояние)



а) б) в)

Рис. 5. Работа программы в режиме ACTIVITY (а – начальное состояние, б – промежуточное состояние, в – конечное состояние)

IV. Заключение

Использование векторного представления направления вместо углов позволило избежать неоднозначностей при усреднении и обеспечило корректность вычислений на всех этапах. Введенный механизм взаимной коррекции соседей существенно ускорил процесс выравнивания и стабилизировал динамику системы. Параметры останова, основанные на величине изменения направления и степени глобальной согласованности, позволили точно определить момент завершения синхронизации.

Проведенное моделирование подтвердило, что локальные итеративные правила способны порождать глобальный порядок, а скорость синхронизации во многом зависит от случайной начальной конфигурации. Таким образом, был создан алгоритм, демонстрирующий формирование согласованного направления во всей двумерной решетке за счет исключительно локальных взаимодействий и самоорганизации системы.

Список литературы

[1] Турбин М. Д. Задача Майхилла о синхронизации ряда стрелков // Математическое просвещение. 2016. Вып. 20. С. 238–241.

[2] Варшавский В. И., Поспелов Д. А. Оркестр играет без дирижера: размышления об эволюции некоторых технических систем и управлении ими // Наука. М., 1984.

[3] Герасимов И. В., Кузьмин С. А., Сафьянников Н. М. Репликационные процессы в толерантных системах // Управление и информационные технологии (УИТ-2008): Доклады 5-й научной конференции, Санкт-Петербург, 14-16 октября 2008 г. / СПбГЭТУ «ЛЭТИ», СПб., 2008. в 2-х т. Т.1. 228 с.

[4] Python documentation. URL: https://www.python.org/doc/ (дата обращения: 20.03.2026 г.).

[5] Pygame documentation. URL: https://www.pygame.org/docs/ (дата обращения: 20.03.2026 г.).