Задание 4: Генетический алгоритм

Реализуйте генетический алгоритм. В генетических алгоритмах имитируется естественный отбор в популяции на протяжении многих поколений.

Решите с его помощью задачу, в которой для оценки решения необходимо имитировать смену состояний в клеточном автомате. Состояния клеточных автоматов иногда тоже называются “поколения“, но будем их называть именно состояниями, во избежание путаницы с поколениями ГА.

Исследуйте зависимость стабильности и качества решения, а также вычислительных затрат на выполнение алгоритма, от интенсивности мутации.

Решаемая задача

Имеется поле размером `50x50` квадратных клеток Каждая клетка может быть либо заполненной (значение 1), либо пустой (значение 0). Конкретное сочетание заполненных и пустых клеток поля называется конфигурацией (или состоянием) клеточного автомата.

У каждой конфигурации есть “потомок“. Он определяется по правилам клеточного автомата Дж. Конвея “Жизнь“. Замена конфигурации на её потомка называется шагом клеточного автомата.

Ограниченность поля учитывается следующим образом: для клетки, находящейся на границе поля (в т.ч. в углу), её соседи, находящиеся за границей поля, считаются всегда пустыми. Как если бы поле было окружено рамкой из постоянных “нулей“.

Среди конфигураций, удовлетворяющих заданному ограничению, **требуется** найти ту, на которой минимизируется значение заданного критерия.

Критерий: количество заполненных клеток после 100 шагов клеточного автомата (т.е. в 101-й конфигурации, в нумерации с 1).

Ограничение: конфигурация, возникающая после 100 шагов клеточного автомата, не является стационарной. То есть её потомок (результат следующего шага клеточного автомата) не совпадает с ней.

Требования к реализации генетического алгоритма (ГА)

Необходимо реализовать "обычный" генетический алгоритм согласно описанию из лекций В.А. Костенко. Более детальное описание ГА, в т.ч. операций селекции, скрещивания и мутации, можно найти в этом материале.

Уточнение описанной в лекции схемы: по ходу выполнения алгоритма необходимо запоминать лучшее из найденных решений и обновлять его (если найдено ещё лучшее). При этом НЕ включая его в обязательном порядке в каждую из последующих популяций. Смысл – не потерять лучшее решение, найденное на какой-то из промежуточных итераций ГА.

Алгоритм должен быть реализован в виде:

- Головного класса, реализующего основной цикл ГА
- Абстрактных классов для операций ГА и вычисления функции выживаемости
- Конкретных классов, реализующих операции ГА и вычисление функции выживаемости (организация классов – по аналогии с заданием по имитации отжига)

Решение можно представлять без абстракции, сразу в виде битового вектора. Это соответствует классической схеме ГА, в которой представление решения в виде битовой строки уже является его абстрагированием.

Внимание: использование свободно доступной реализации движка клеточного автомата "Жизнь" – допустимо. Но некорректность работы сдаваемой реализации, связанная с ошибкой в таком движке или в интеграции его с ГА, считается полновесной ошибкой (отсылки к "глючности" заимствованного движка не являются оправданиями).

Конкретизация алгоритма

- Функция выживаемости: значение оптимизируемого критерия. Возможно, с добавлением штрафа (о штрафе написано ниже).
- Размер популяции: $N_{\text{pop}} = 100$.
- Кодирование решения битовым вектором: очевидное, с учётом того что у каждой клетки поля два возможных значения.
- Начальная популяция: полностью случайно генерируется, вероятность того что в клетке 1 равна 0.5.
- Операция мутации: как в лекциях, т.е. каждый бит решения мутирует (инвертируется) с вероятностью P_{mut}. Значение P_{mut} изменяется в ходе исследования (см. ниже). В ходе конкретного выполнения алгоритма P_{mut} должно быть зафиксировано.
- Операция селекции:
  - `S1`: пропорциональная селекция
  - `S2`: рулеточная селекция
  - `S3`: турнирная селекция (пункт 9.3.3).

- Операция скрещивания:
  - `X1`: одноточечное скрещивание
  - `X2`: двухточечное скрещивание
- Вероятность скрещивания: 0.8
- Допускать или не допускать в популяцию решения, не удовлетворяющие ограничению (т.е. становящиеся стационарными через N шагов клеточного автомата) – выбор на усмотрение студента:

  a) Штрафовать решение, не удовлетворяющее ограничению (искусственно повышать, т.е. ухудшать, значение критерия на нём)

  b) Отбрасывать решения, не удовлетворяющие ограничению (и повторно выполнять операцию мутации или скрещивания, породившую отброшенное решение – чтобы в силу рандомизации со следующей попытки эта операция породила решение, удовлетворяющее ограничению)

  Вариант a) рискован тем, что можно "застрять" в области некорректных (не удовлетворяющих ограничению) решений. Некорректные решения нельзя запоминать в качестве "текущих лучших", а также выдавать в качестве итогового результата.

  В варианте b) важно не зациклиться на попытках формирования корректных решений. Против такого зацикливания нужно принимать меры.
- Критерий останова: 50 итераций ГА (т.е. 50 смен популяций) подряд без улучшения значения оптимизируемого критерия **на лучшем из найденных решений**. То есть 50 итераций подряд не удалось найти решение с меньшим значением критерия, чем у ранее запомненного лучшего из ранее найденных решений.

Выбор операций селекции и скрещивания

- Выбор операции селекции: на основании остатка от деления на 3 контрольной суммы `CRC32` от фамилии и инициалов студента (в формате `ИвановАБ`).
  - остаток 1: вариант `S1`
  - остаток 2: вариант `S2`
  - остаток 0: вариант `S3`

  Онлайн-расчёт `CRC32` или питоном:
  ```
  python3 -c "print(__import__('zlib').crc32(input().encode()) % 3)" <<< "ИвановАБ"
  ```
- Выбор операции скрещивания: на основании чётности контрольной суммы `CRC32` от фамилии и инициалов студента (в формате `ИвановАБ`). Если контрольная сумма нечётная, то вариант `X1`. Если чётная, то вариант `X2`.

Требования к исследованию

Все входные данные описаны выше и являются, в основном, настройками алгоритма.

Необходимо исследовать зависимость характеристик работы алгоритма от интенсивности мутации, т.е. от значения P_{mut}. Начальное значение P_{mut}: $P_{\text{mut-init}} = 1/(50 \cdot 50) = 0.0004$, т.е. в среднем в каждом решении мутирует 1 бит.

Изменять P_{mut} в ходе исследования следует по формуле: $P_{\text{mut}}(i) = P_{\text{mut-init}} \cdot 1.5^i$, $i = \overline{0,9}$; i – номер серии экспериментов, т.е. нужно провести 10 серий экспериментов, каждая со своим фиксированным значением P_{mut}. Например, в серии 3 $P_{\text{mut}} = P_{\text{mut}}(3) = 0.0004 \cdot 1.5^3 = 0.00135$.

Для каждого значения i необходимо провести серию из 10 запусков ГА с соответствующим значением $P_{\text{mut}} = P_{\text{mut}}(i)$ и определить:

- Стабильность алгоритма (разброс значений критерия на решении-результате, т.е. разность между значениями критерия на худшем и на лучшем прогоне)
- качество работы алгоритма (значение критерия на лучшем прогоне)
- вычислительные затраты на выполнение алгоритма (количество процессорного времени, затраченного на прогон; брать максимум по 10 прогонам)

В отчёт по исследованию должны входить:

- график зависимости стабильности алгоритма от значения P_{mut}
- график зависимости качества работы алгоритма от значения P_{mut}
- график зависимости времени прогона от значения P_{mut}

На горизонтальной оси графика откладывать значение i.

Внимание: не забудьте перезапускать датчик случайных чисел, чтобы он каждый раз выдавал новую последовательность. Если не перезапускать датчик, алгоритм станет работать детерминированно.

Визуализация:

- Для каждого прогона алгоритма, в файл должно быть сохранено наилучшее найденное решение, а также результат работы клеточного автомата через 100 шагов. Сохранять нужно в виде визуально читающейся матрицы `50x50`, где "-" - незаполненная клетка, "X" – заполненная.

  Пример матрицы `3x3` ("планер"):

  ```
  -X-
  X--
  XXX
  ```

- Имя файла с решением должно иметь вид: `series_*_run_*_sol.txt`, где вместо звездочек указаны значение i и номер прогона в рамках серии для этого i. Пример: `series_10_run_1.txt`, `series_100_run_10.txt`
- Имя файла с результатом работы клеточного автомата через 100 шагов должно иметь вид: `series_*_run_*_sol_after100.txt`

Визуализация (опционально)

Реализуйте пошаговую визуализацию работы клеточного автомата на наилучшем найденном решении (взятом как начальное состояние) в течение 100 шагов.

Например, в текстовом терминале. Формат вывода матрицы – как для файла (см. выше). Частота смены состояний – два состояния в секунду. Перерисовываться поле должно "красиво и на том же месте", а не прокруткой экрана. Вдохновиться можно веб-имитатором; он, кстати, правильно отрабатывает граничные эффекты.

Для текстового вывода можно использовать (например) библиотеку `ncurses`.

Рекомендации по отладке движка клеточного автомата

Проверьте, как ведут себя такие известные конфигурации, как "планер" и другие нестационарные конфигурации со страницы тут.

Отдельно проверьте правильность обработки границ поля. Поставьте на границе (в т.ч. в углу поля) какую-нибудь стационарную конфигурацию. Запустите на границу "планер".

В отладке сильно поможет визуализатор – тот самый, за который даётся бонус.

Проверка движка настоятельно рекомендуется в т.ч. в случае использования заимствованного движка.

Рекомендации по автоматизации исследования

Рекомендуется полностью автоматизировать запуск серии экспериментов, хотя бы для заданного значения i, с помощью командного сценария (i - параметр сценария). Командный сценарий передаёт значение i и номер эксперимента в рамках серии в реализацию алгоритма при запуске её исполняемого файла.

Также командный сценарий замеряет время выполнения каждого запуска, получает от алгоритма значение целевой функции на итоговом решении (например, через временный файл), и сохраняет эти два числа в файл протокола запуска (например, в формате `CSV`; в каждой строчке – длительность запуска и значение целевой функции).

Объединять выведенные разными запусками строки формата `CSV` в общий файл можно при помощи до-записи в этот файл (перенаправление вывода по `>>` под Linux).

Отлаживать сценарий такого "пакетного" выполнения запусков рекомендуется на задаче с урезанными параметрами, например при значительно сокращенном размере поля и количестве шагов клеточного автомата. Чтобы пакет экспериментов выполнялся быстрее. Полноценные эксперименты запускать с помощью уже отлаженного командного сценария.

Оценка составных частей задания в баллах

- (обязательно) Реализация и исследование алгоритма: 7 баллов
- (опционально) Реализация визуализатора: 3 балла
- Реализация без исследования оценке не подлежит (0 баллов)