Ход работы:

Для решения задачи о нахождении расстановки N ферзей на доске NxN так, чтобы ни один ферзь не бил другого, был выбран фреймворк Watchmaker для реализации генетического алгоритма.

Язык программирования: Kotlin.

В задаче требуется найти одно из оптимальных решений.

Очевидно, что ферзи должны стоять в разных столбцах и строках, т.е. в каждом столбце и строке должен быть ровно один ферзь. При этом мы не можем одновременно учитывать еще и бои по диагонали, их количество в будущем нам нужно минимизировать.

Будем хранить одно решение как номера клеток в столбцах, на которые необходимо поставить ферзей. Нумерацию строк и столбцов будем вести с 0. Например, для задачи N=5, одно из решений будет [2, 0, 3, 1, 4], т.е. первый ферзь стоит на пересечении 0 столбца и 2 строки, второй ферзь на пересечении 1 столбца и 0 строки и т.п.

Фитнесс функция будет подсчитывать количество боев ферзей друг друга в текущем решении, это значение необходимо минимизировать.

Генерация особей: случайная перестановка от 0 до N не включительно, которая не создает конфликтов (боев) по столбцам и строкам, т.к. у такой особи все столбцы различны и все строки различны. При этом у таких особей могут быть конфликты по диагонали. От этих конфликтов мы и будем избавляться.

Мутация: лучше всего себя показала SWAP мутация, когда два случайных гена меняются местами.

Кроссовер: лучшим оказался упорядоченный кроссовер, при этом я также реализовал кроссовер, у которого потомок содержит только повторяющиеся у родителей значения (одинаковые столбец + строка), остальные позиции заполняются случайно оставшимися значениями.

Условие терминации алгоритма: значение фитнесс-функции 0 или превышение заданного количества итераций (если фитнесс-функция не 0, то решение не найдено).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Размер | Популяция; макс. итераций | Вероятность кроссовера | Вероятность мутации | Итерация с  решением |
| 8 | 10; 100 | 0.1 | 0.7 | 23 |
| 16 | 50; 500 | 0.2 | 0.5 | 260 |
| 32 | 100; 3000 | 0.3 | 0.5 | 1374 |
| 64 | 100; 10000 | 0.3 | 0.5 | 3519 |
| 128 | 200; 1000 | 0.1 | 0.5 | 591 |
| 200 | 300; 5000 | 0.1 | 0.5 | 705 |
| 400 | 600; 3000 | 0.1 | 0.5 | 2109 |

Дополнительные параметры, подобранные эмпирически:   
1. Elite count = размер\_задачи / 6  
2. Selection operator: RankSelection

Общие наблюдения: иногда очень влияет рандом, а именно начальная популяция, кроссовер должен происходить с маленькой вероятность, а вероятность мутации не так сильно важна и может быть в пределах 0.5-0.9, но лучше ближе к 0.5.

Вопросы:

1. Задача является оптимизационной, так как решается методами оптимизации.
2. Судя по результатам этой https://arxiv.org/pdf/2107.13460.pdf статьи, общее количество решений сопоставимо с n^n. Но в текущей задаче, когда требуется найти только одно решение, всё зависит от используемого метода. Конкретно для ГА: сложность фитнесс-функции квадратичная.