Лабораторная работа №5. Проектирование эволюционного алгоритма для задачи расстановки ферзей.

Цель работы

Освоение всего цикла разработки эволюционных алгоритмов, начиная с анализа проблемы и проектирования, заканчивая настройкой параметров и анализом эффективности.

Оборудование и программное обеспечение

- Использована Java JDK 11
- Watchmaker framework версии 0.7.1 (требует JDK 1.8 и выше)
- Для тестов был использован JUnit 5 (требует JDK 8.0 и выше)

Задача

Разработать генетический алгоритм, решающий задачу расстаовки N ферзей на доске NxN, чтобы они не били друг друга.

Для ршения задачи будет использован генетический алгоритм.

Представление решений (QueensSolution)

Решение (объект класса QueensSolution) имеет поле rowIndexes - целочисленный массив, хранящее гены. Гены представлены в виде одномерного массива длиной N, содержащего перестановку чисел от 0 до N-1. Элемент i данного массива содержит нимер строки ферзя, стоящего на i-ом столбце. Такое представление гарантирует, что в любом решении ни один ферзь не будет стоять с другим на одной строке или столбце.

QueensSolution имеет вспомогательные методы для реализации генетических операторов. Генетические опреаторы не взаимодействуют с полем rowIndexes напрямую: только с помощью этих методов. Таким образом достигается абстракиця и инкапсуляция. Если потребуется хранить гены в другом виде, нам придется поменять реализацию интерфейса QueensSolution, в дргуих классах код останется прежним.

Фитнес-функция (QueensFitnessFunction)

Фитнес функция подсчитывает число ферзей, находящихся под ударом. Благодаря представлению рещений нам достаточно проверять, находятся ли ферзи на одной диагонали.

Терминация

Условией терминации является нулевое значение фитнесс-функции.

Мутации (QueensMutation, QueensScrambleMutation, QueensInversionMutation, QueensInsertMutation, QueensSwapMutation)

В пайплайн внедрены сразу 4 мутации: вставкой, перемешиванием, инверсией и перестановкой. Все эти мутации имеют один параметр (передается в конструктор) - вероятность для каждого элемента популяции, что к нему будет применена данная мутация.

Кроссовер (QueensCrossover)

Реализует упорядоченный кроссовер.

Результаты экспериментов

В таблице представлены результаты работы алгоритма при разных N. Результаты усреднены по 10 запускам, кроме N = 32. Для N = 32 был произведен 1 запуск.

Пояснение названий столбцов:

- C - вероятность кроссовера
- SwM - вероятность мутации перестановкой
- InsM - вероятность мутации вставкой
- InvM - вероятность мутации инверсией
- ScM - вероятность мутации перемешиванием

N	Размер популяции	C	SwM	InsM	InvM	ScM	Число итераций до решения	Минимум итераций до решения
4	10	1.0	0.5	0.5	0.5	0.5	1	1
8	100	1.0	0.5	0.5	0.5	0.5	6.3	1
8	100	1.0	0.3	0.3	0.3	0.3	5.3	1
8	100	1.0	0.4	0	0	0	4.8	1
8	100	1.0	0	0	0	0.4	7.3	1
16	2000	1.0	0.4	0.3	0.1	0.1	488.4	1
16	20000	1.0	0.4	0.3	0.1	0.1	38.4	4
16	200000	1.0	0.4	0.3	0.1	0.1	7.2	2
16	1000000	1.0	0.4	0.3	0.1	0.1	1.6	2
32	2000000	1.0	0.4	0.3	0.1	0.1	517	-

Ответы на вопросы

1. Является ли задача оптимизационной или ограниченной?

Из методических материалов я сделал выввод, что ограниченная задача - задача, в которой при определенных условиях невозможно вычислить функцию качетсва. Если так определять ограниченную задачу, то данная задача не является ограниченной.

Данную задачу можно назвать оптимизационной, потому что мы ищем решение, соответствующее оптимальному значению фитнес-функции. Оптимум фитнес-функции известен заранее.

Еще нужно сказать, что мы выбрав представление рещения ограничили пространство поиска.

2. Как растет сложность задачи при увеличении размерности?

Сложность задачи быстро растет с увеличением размерности. Дело в том, что для размерности N число возможных решений при описанном выше представлении равно N!. Знаменитая последовательность числа расстановок ферзей растет гораздо медленне чем последовательность факториалов. Поэтому мало того, что пространство поиска растет факториально, но вероятность того, что случайно выбранная перестановка окажется верным решением падает факториально.

При N = 27!

27! = 10 888 869 450 418 352 160 768 000 000

nSolutions_27 = 234 907 967 154 122 528