Проектирование эволюционного алгоритма для задачи расстановки ферзей

Цель работы

Целью данной работы является освоение всего цикла разработки эволюционных алгоритмов, начиная с анализа проблемы и проектирования, заканчивая настройкой параметров и анализом эффективности.

Выполнение работы

Для выполнения работы был использован фреймворк watchmaker.

Представление решений

Представление решения должно однозначно определять расположение `n` ферзей на квадратной доске `n x n`. При этом известно, что все решения, в которых ферзи стоят в одном столбце(строке) являются неверными, поэтому сама структура решений может предполагать данные ограничения. В связи с этим в качестве представления решения была выбрана случайная перестановка чисел `[0...n]`. Таким образом решение может быть неверным только в случае, если ферзи бьют друг друга по диагонали.

Fitness function

Для оценки качества сгенерированного решения было решено использовать количество пар ферзей, стоящих на одной иагонали. Это стало возможным благодаря нашей кодировке решения, предполагающей непересекаемость по вертикали/горизонтали.

Мутация

Оператор мутации представляет из себя `k` случайных попарных перестановок элементов решения. После такой операции решение останется перестановкой.

Кроссовер

Кроссовер, применяемый к двум особям сохраняет некоторую подпоследовательность одного из родителей на своем месте, а остальные элементы перестановки сортирует в соответствии с порядком во втором родителе и заполняет пропуски.

Нюансы

В качестве условия терминации было выбрано достижение `fitness_function = 0`. Для слишком больших размерностей задачи заменили условие остановки на стагнацию на протяжении 1000 итераций.

Оператор селекции = `RankSelection`.

Для поиска решения используется генетический алгоритм.

Параметрами алгоритма являются:

- размер популяции
- количество мутаций

Результаты

N	population size	mutations	fitness evaluations	generations	min fitness
4	10	2	14,90	0,50	0
8	30	2	335,20	10,30	0
16	50	2	63100,20	268,80	0
32	100	2	181685,40	825,60	6,40
64	500	10	1368309.50	1249,40	31,20
128	1000	10	2576804,60	1583,80	63,60

Для достаточно больших размерностей алгоритм не сошелся за вменяемое время.

Вопросы

1. Задача не является оптимизационной, т.к. мы стараемся не столько улучшить fitness, сколько использовать его для нахождения решения задачи. При этом проверить, является ли решение таковым, можно с точностью.
2. Сложность вычисления fitness-function зависит от размерности как O(n^2). Однако при увеличении размерности для получения решения необходимо также увиличивать размер популяции, поэтому и количество вычислений fitness-function будет увеличиваться.