Федеральное государственное бюджетное учреждение  
высшего профессионального образования и науки  
Санкт-Петербургский Академический университет — научно-образовательный центр нанотехнологий Российской академии наук  
Лицей «Физико-техническая школа»

**Практическая работа**

**Разработка и оптимизация генетического алгоритма**

Работу выполнил:

Корнев И. В. (11В)

Научный руководитель:

Дворкин М. Э.

Место выполнения:

Лицей ФТШ

Содержание

[Введение 1](#_Toc386524189)

[Описание алгоритма 2](#_Toc386524190)

[Главный цикл 2](#_Toc386524191)

[Размножение 2](#_Toc386524192)

[Мутация 2](#_Toc386524193)

[Селекция 3](#_Toc386524194)

[Параметры 3](#_Toc386524195)

[Самоадаптация 3](#_Toc386524196)

[Реализация алгоритма 4](#_Toc386524197)

[Структура модуля Genetic 4](#_Toc386524198)

[Модули Knights и Queens 5](#_Toc386524199)

[Модуль Tests 5](#_Toc386524200)

[Полученные результаты 6](#_Toc386524201)

[Knights 6](#_Toc386524202)

[Queens 6](#_Toc386524203)

[Поклеточные скрещивание и мутация 6](#_Toc386524204)

[Построчные скрещивание и мутация 7](#_Toc386524205)

[Список литературы 8](#_Toc386524206)

# Введение

Генетический алгоритм – это алгоритм, основанный на эволюционных методах. Он является эвристической оптимизацией полного перебора, то есть генетическим алгоритмом можно решить любую задачу, в которой нужно подобрать некоторый набор параметров (генов) для обеспечения наилучшего результата (максимальная выживаемость), однако не гарантировано то, что конечное решение будет идеально.

Целью научной практики было написание алгоритма и проверка целесообразности его использования и применения к самому себе для саморазвития с целью увеличения получения идеального решения.

# Описание алгоритма

## Главный цикл

Работу алгоритма в общем виде можно описать блок-схемой, представленной справа. Каждая особь начальной популяции получает случайные гены. Здесь и дальше  
*цикл* – это одно последовательное выполнение процедур размножения, мутации и селекции. *Размер популяции* в начале и конце каждого цикла одинаков, однако в течение цикла может меняться.

## Размножение

Особи не имеют гендерных различий, и на этапе размножения любая из них может стать отцом или матерью. В каждом цикле рождается одинаковое *количество детей*, каждый из которых имеет комбинацию генов всех родителей (*количество родителей* одного ребенка не обязательно равно двум). Рождение одного ребенка – это один запуск алгоритма, изображенного на блок-схеме слева. Вероятность для особи быть выбранной в качестве родителя пропорциональна ее приспособленности.

## Мутация

Действия на блок-схеме справа выполняются поочередно для каждой особи популяции, при этом *максимальное количество изменений (max)* у особей прошлого поколения и у новорожденных может быть задано отдельно.

## Селекция

Селекция обеспечивает отсеивание худших особей и возвращение размера популяции к изначально установленному.

## Параметры

У данной модели генетического алгоритма есть несколько основных параметров, сильно влияющих на ход эволюции, которые были отмечены курсивом: размер популяции, количество детей, рождающихся в каждом цикле, количество родителей одного ребенка, максимальное количество изменений генов особей прошлого и нового поколений. Эффективность алгоритма также сильно зависит от функций приспособленности, мутации и скрещивания особи. Для каждой задачи есть свой оптимальный набор параметров, при котором ее решение находится алгоритмом с наибольшей вероятностью, следовательно, применяя генетический алгоритм сам к себе, можно автоматически подобрать такой набор.

## Самоадаптация

При рассмотрении применения алгоритма к самому себе было бы удобно называть главный алгоритм, который занимается эволюцией подвластного ему алгоритма, верхним, а подвластный ему – нижним. Особью верхнего алгоритма является набор нижних алгоритмов с одинаковыми параметрами, а генами – эти параметры. Каждой особи верхнего алгоритма нужно запускать несколько нижних эволюций и доводить их до конца, после чего считать приспособленность лучшей особи каждого нижнего алгоритма и усреднять полученные значения. Это усредненное значение и будет приспособленностью особи верхнего алгоритма. Таким образом мы понимаем, насколько хорошие в среднем получаются особи нижних алгоритмов в зависимости от параметров этих алгоритмов.

# Реализация алгоритма

Алгоритм был написан на языке Python 3. Главным модулем и интерфейсом является модуль Genetic.

## Структура модуля Genetic

Класс Population:

* \_\_init\_\_ – конструктор, задает начальную популяцию
* kind – указывает на класс особи популяции
* cycle - цикл
* breed\_all – этап размножения
* mutate\_all – этап мутации
* select – этап селекции
* choose\_parent – вероятностный выбор одного родителя

Все эти методы, кроме kind, определены.

Класс Species:

* \_\_init\_\_ – конструктор
* breed(mates) – особь, данный метод которой будет вызван, родит ребенка, полученного смешением генов данной особи и всех особей из списка mates
* mutate – метод, производяций случайное изменение одного гена, вызывается из класса Population по несколько раз
* fitness – функция приспособленности
* \_fitness – оптимизация, немного сокращающая количество вызовов fitness
* draw – используется в GUI
* clone – создание копии особи без сохранения ссылок на нее

Все методы, кроме \_fitness, не определены, их задает пользователь.

Класс GUI не представляет особого интереса в контексте работы.

## Модули Knights и Queens

Работа алгоритма рассматривалась на примере задачи про коней и задачи про ферзей, суть которых весьма похожа: надо расставить максимальное количество фигур по шахматному полю так, чтобы они не могли бить друг друга. Эти модули определяют классы, наследующие Population и Species и доопределяющие их.

Особями являются шахматные поля, генами – расположения фигур. Одна мутация – это выбор случайной клетки и, если там стояла фигура, убирание ее, иначе – наоборот. Скрещивание – это заимствование для каждой клетки поля ребенка наличия или отсутствия в ней фигуры у одного случайного родителя из выбранных. Функция приспособленности – это количество фигур минус количество атакуемых фигур.

## Модуль Tests

Этот модуль занимается эволюцией эволюций, и классы этого модуля тоже наследуют классам модуля Genetic. В нем реализован многопоточный параллельный запуск эволюций коней или ферзей для увеличения скорости работы.

# Полученные результаты

## Knights

Зависимость среднего процента верных (идеальных) решений для задачи про коней по всем нижним алгоритмам от количества завершенных циклов верхнего алгоритма представлена на графике.

Интересен тот факт, что верхний алгоритм решил, что особям нижнего алгоритма лучше не скрещиваться, то есть только мутировать и «почковаться», создавать клонов. Изменение функции скрещивания не дало явного результата.

## Queens

### Поклеточные скрещивание и мутация

Более того, оказалось, заметные улучшения в нижних эволюциях для ферзей происходят только на первых трех циклах верхней эволюции. Скорее всего, это происходит из-за того, что хорошее поле с ферзями очень неустойчиво, и любая малейшая неудачная мутация вызывает сильное падение приспособленности, поэтому было решено немного оптимизировать функции скрещивания и мутации.

### Построчные скрещивание и мутация

Вполне очевидно, что два ферзя не могут находиться в одной строке, следовательно можно учесть это в функциях. Если сделать, чтобы при скрещивании заимствовались от родителей не одиночные клетки, а строки, а при мутации две случайные строки шахматного поля менялись местами, то результат резко улучшается: за те же три цикла верхней эволюции нижние эволюции начинают находить правильный ответ в ста процентах случаев. При этом видно, что верхний алгоритм тоже «заметил» улучшение качества функций: до улучшения он приходил к выводу, что скрещиваться особям не надо (как в задаче с конями), а после решил, что лучший вариант – это наличие сразу четырех родителей.

# Список литературы

Электронные ресурсы:

1. G. Zhang, S. Wang, and Y. Li, A Self-adaptive Genetic Algorithm Based on the Principle of Searching for Things, Baoding, China
2. S. Meyer-Nieberg and H. Beyer, Self-Adaptation in Evolutionary Algorithms, Springer-Verlag Berlin Heidelberg 2007
3. А. Еремеев, Генетические алгоритмы и оптимизация, РФФИ
4. L. Budin, M. Golub, D. Jakobovic, Parallel Adaptive Genetic Algorithm, Zagreb, Croatia
5. А. Сергиенко, П. Галушин, В. Бухтояров, Р. Сергиенко, Е. Сопов,  
   С. Сопов, Описание стандартного генетического алгоритма, Сибирский государственный аэрокосмический университет имени академика  
   М.Ф. Решетнева