Южное окружное управление образования государственное бюджетное образовательное учреждение «Школа №1553 имени В.И. Вернадского»

Сравнение генетических алгоритмов  
 с мутацией и без мутации

Курсовая работа

Заблудовского Ильи

9 класс

Научный руководитель:

Петров Илья Михайлович

# Введение

При создании программ для роботов, к примеру для езды по линии, требуется выполнять достаточно долгую и трудоемкую настройку всей программы путем подбора подходящих коэффициентов. Существуют алгоритмы, позволяющие подбирать эти коэффициенты в автоматическом или полуавтоматическом режиме. Объектом нашего исследования стал один из таких алгоритмов, а именно - Генетический алгоритм. Его полезное свойство - подбор коэффициентов - тем не менее нуждается в оценке эффективности, что и стало целью нашей работы.

# Цели задачи и методы

Цель: Сравнить два генетических алгоритма: с мутацией и без.

Задачи:

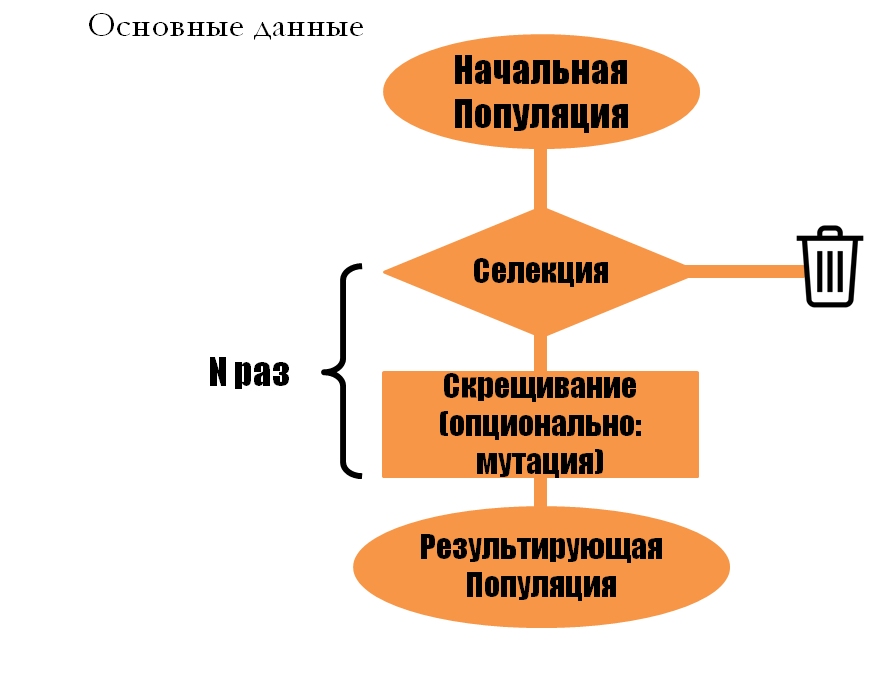
* Изучить особенности генетического алгоритма
* Собрать робота на основе микроконтроллера LEGO MINDSTORMS NXT 2.0
* Написать программу
* Провести эксперименты
* Сравнить результаты экспериментов

Методы:

* Сравнительный анализ

# Генетический алгоритм

Генетический алгоритм по сути своей есть алгоритм искусственной селекции, который мы применяем к программе для поиска оптимальных коэффициентов. Коэффициенты в данном случае являются хромосомой программы, а один набор коэффициентов - особью.



*рис. 1*

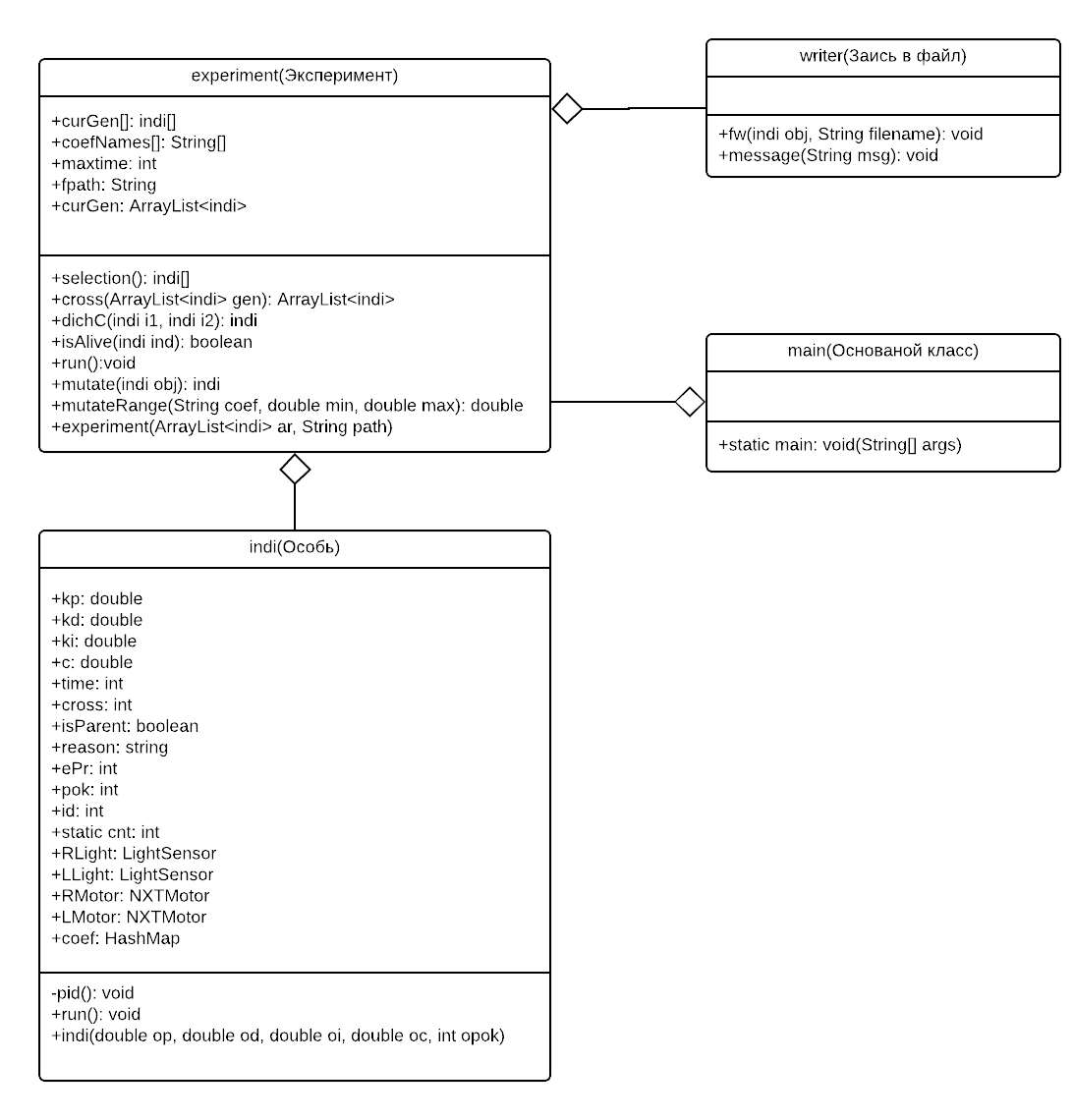
Первым шагом, мы генерируем начальную популяцию по определенному принципу либо случайно, затем эта популяция “Живет” в течении “Жизни” Особь может скреститься с другой особью, формируя потомство со скрещенной и опционально мутировать. После чего происходит селекция, или же отбор. Этот отбор происходит по принципу выжил – не выжил, а в моем случае - проехал робот по линии или слетел с нее. После чего выжившие особи создают новую популяцию, размножаясь между собой.

Так это повторяется много раз до того момента, как особь будет удовлетворять так называемую

“функцию останова” – это функция, которая должна определять удовлетворяет нас эта особь или нет. После того как все итерации выполнены, мы получаем результирующую популяцию, из которой выбираем наилучшую особь по какому-либо признаку, в нашем случае по скорости прохождения трассы. Эта особь и будет являться результатом работы генетического алгоритма.

# Реализация

Я решил реализовать генетический алгоритм на объектно-ориентированном языке программирования Java. Моей целью было написать алгоритм, который может выполнять эксперимент с мутацией и без нее в зависимости от передаваемых в нее данных. Я составил диаграмму классов (см рис. 2), описывающую классы, функции,сгруппированные по классам, и их отношение внутри программы.

  
*рис. 2*

Класс indi является “шаблоном” особи. Рассмотрим его методы. После создании особи необходимо вызвать метод run(), который будет в цикле запускать метод pid(). Выход из этого цикла происходит только в двух случаях: либо робот проехал трассу успешно, либо была нажата кнопка остановки программы, что говорит о том, что робот слетел с трассы.



*рис. 3*

Успешным проходом трассы является тот случай, когда робот проехал стартовый перекресток и также проехал конечный перекресток.

Метод pid() считывает данные с сенсоров и запускает моторы с определенной мощностью. Эта мощность считается при помощи PID алгоритма.

PID алгоритм - это способ расчета некоторой управляющей величины на основе ошибки, исходящей из данных с сенсоров. Эта величина считается по формуле

Где - средняя скорость, - ошибка, а именно разница значений датчиков, - интегральная величина, а именно накапливаемая ошибка(), которая сбрасывается при пересечении линии датчиком, - дифференциальная величина, которая считается по формуле где - предыдущая ошибка. это коэффициенты, созданные для баланса их соответствующих величин, их-то и надо подобрать.

Перейдем к классу эксперимента. Он контролирует все манипуляции с особями: их запуск, селекцию, скрещивание и мутацию.

Рассмотрим его методы. При создании объекта класса “эксперимент” мы передаем в него: первое поколение, имя файла, в который будут сохраняться данные об особях, количество поколений, после создания которых программа остановится, и переменную, которая определяет, будет ли в эксперименте запускаться мутация.

Метод run() описывает последовательность действий, применяемых к особям: он повторяет алгоритм, описанный выше (см рис. 1)

Начинает он с запуска метода run() у особи, после того как метод выполнился, запускается метод selection() который фильтрует особей по двум признакам, первый признак - время менее 100 секунд, а второй признак - успешное прохождение трассы. После фильтрации создается следующее поколение методом cross(), который скрещивает 2 случайно взятые особи и создает из них одну новую со скрещенной хромосомой. Если в данном эксперименте используется мутация, то далее запускается метод mutate(), который по очереди запускает все коэффициенты в метод mutateRange(), который в свою очередь изменяет каждый коэффициент в своем интервале(см табл. 1).



*табл. 1*

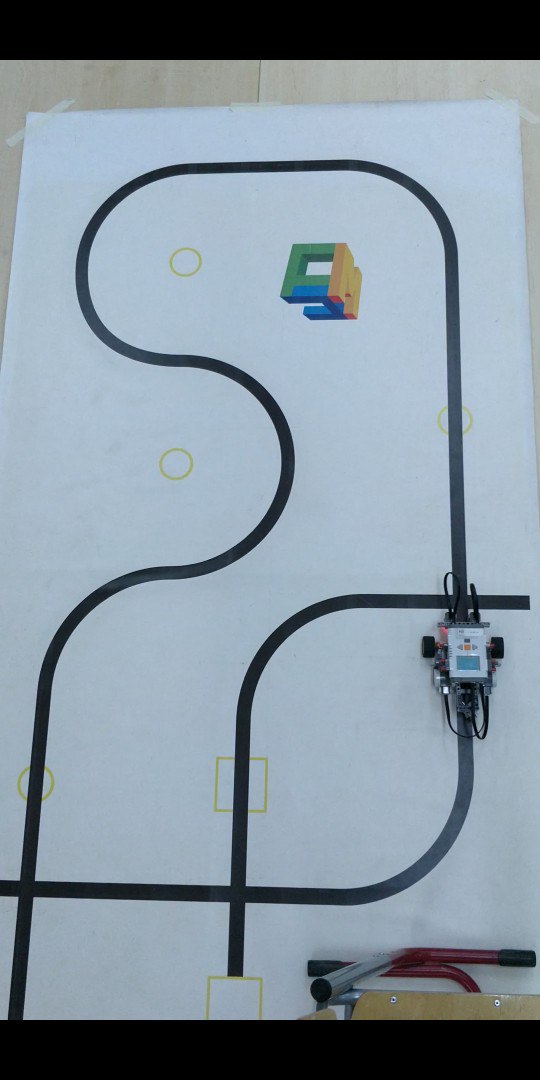
После прохождения n-ого количества поколений программа выключается.

Также есть вспомогательный класс fw, то есть fileWriter, который осуществляет запись данных об особях в файл.

# Описание эксперимента

В эксперименте использовался робот, собранный из ЛЕГО, управляемый микроконтроллером LEGO MINDSTORMS NXT 2.0. Так как мной был выбран объектно-ориентированный язык программирования Java, на микроконтроллер NXT я установил прошивку LejOS. Также робот снабжен двумя датчиками освещения, двумя сервомоторами.

В качестве полигона для проведения эксперимента была использована черная линия на белом фоне с двумя перекрестками для обозначения начала и конца трассы(см рис. 4).

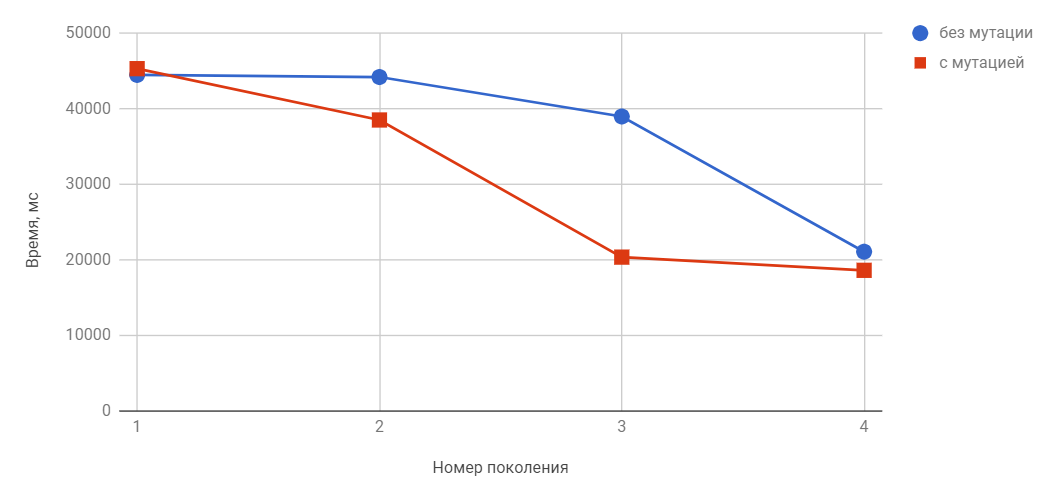


*рис. 4*

Робот выставляется перед стартовым перекрестком, включается программа. Робот проезжает первый перекресток, после чего включает отсчет времени и выключает его по достижению второго перекрестка. Если во время выполнения программы робот слетает с трассы, то ему нажимают кнопку остановки, что означает, что робот не прошел трассу, то есть умер. После этого робота возвращают на стартовый перекресток и запускают следующую особь. Когда все представители первого поколения завершили выполнение своей программы, нажимается кнопка, которая вызывает последовательно методы селекции (selection, cross, опционально: mutation). Затем робот издает сигнал о успешном создании следующего поколения. Этот алгоритм повторяется n раз, затем программа выключается.

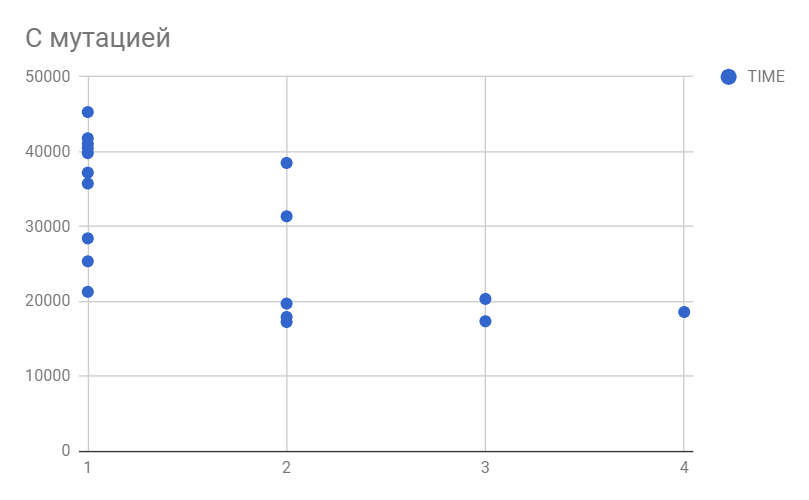
# Результаты

Было проведено два типа эксперимента: один с мутацией, а второй без мутации соответственно. На графике (см рис. 5) можно сравнить время самой медленной особи каждого поколения из обоих экспериментов. Можно заметить, что алгоритм с мутацией приводит к наиболее быстрой особи, также можно заметить, что алгоритм с мутацией более эффективно справляется с поставленной задачей.

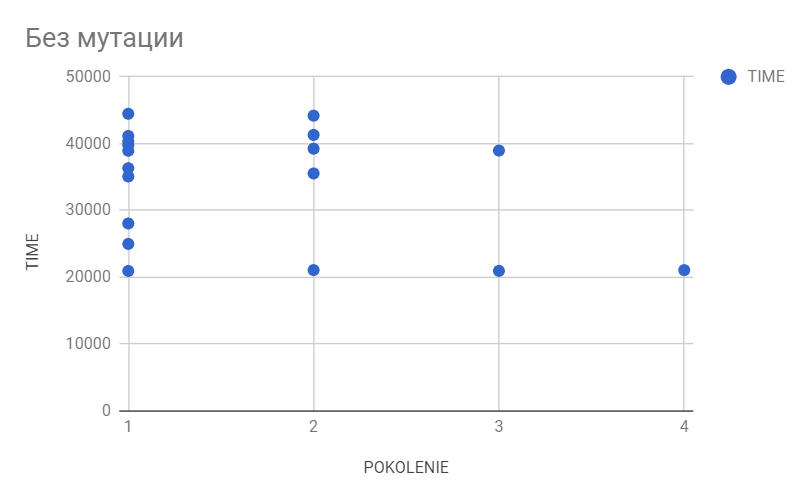


*рис. 5*

С более подробными результатами экспериментов (разброс скоростей в поколениях) можно ознакомиться на рисунках 6 и 7.



*рис. 6*

**

*рис. 7*

# Выводы

1. Собран робот
2. Написана программа на языке Java
3. Проведены эксперименты
4. Произведено сравнение результатов экспериментов. Которые показывают, что генетический алгоритм с мутацией работает лучше и эффективнее генетического алгоритма без мутации.

# Список литературы

1. Т.Г. Каплунов / Вестник ТИ им. А.П Чехова № 1; ИСПОЛЬЗОВАНИЕ ГЕНЕТИЧЕСКИХ АЛГОРИТМОВ ДЛЯ АВТОМАТИЧЕСКОГО ФОРМИРОВАНИЯ ПРОГРАММ УПРАВЛЕНИЯ ДВИЖЕНИЕМ
2. Курейчик В.М. / Генетические Алгоритмы
3. Генетический алгоритм. Просто о сложном / mrk-andreev [Habrahabr](https://habrahabr.ru/post/128704/) - https://habrahabr.ru/post/128704/ / Дата обращения: 01.05.2018
4. Д.И. Батищев, Е.А. Неймарк, Н.В. Старостин./ Применение генетических алгоритмов к решению задач дискретной оптимизации / Новгород 2007г

# 

# Приложения

GitHub repo - https://github.com/MrLalatg/Genetic-Algorythm-NXT-2.0

import java.io.DataOutputStream;

import java.io.File;

import java.io.FileOutputStream;

import java.io.IOException;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.Map;

import java.util.Random;

import lejos.nxt.Button;

import lejos.nxt.ButtonListener;

import lejos.nxt.LightSensor;

import lejos.nxt.Motor;

import lejos.nxt.MotorPort;

import lejos.nxt.NXTMotor;

import lejos.nxt.SensorPort;

import lejos.nxt.Sound;

import lejos.util.Delay;

import lejos.util.Stopwatch;

class writer {

void fw(indi obj, String filename) {

FileOutputStream fos = null;

File inform = new File(filename + ".txt");

try {

fos = new FileOutputStream(inform, true);

} catch(IOException ex) {

System.err.println("Failed to create output stream");

Button.waitForAnyPress();

System.exit(1);

}

DataOutputStream dOut = new DataOutputStream(fos);

try {

dOut.writeChars("KP:" + obj.coef.get("kp").toString() + "\n");

dOut.writeChars("KD:" + obj.coef.get("kd").toString() + "\n");

dOut.writeChars("KI:" + obj.coef.get("ki").toString() + "\n");

dOut.writeChars("C:" + obj.coef.get("c").toString() + "\n");

dOut.writeChars("REASON:" + obj.reason + "\n");

dOut.writeChars("ID:" +obj.id + "\n");

dOut.writeChars("POKOLENIE:" + obj.pok + "\n");

dOut.writeChars("TIME:" + obj.time + "\n");

fos.close();

} catch(IOException ex) {

System.err.println("Failed to write line into the file");

Button.waitForAnyPress();

System.exit(1);

}

}

public void message(String msg) {

FileOutputStream fs = null;

File log = new File("mylog.txt");

try {

fs = new FileOutputStream(log, true);

DataOutputStream out = new DataOutputStream(fs);

out.writeChars(msg + "\n");

out.close();

} catch(IOException ex) {

}

}

}

class indi{

LightSensor RLight = new LightSensor(SensorPort.S1);

LightSensor LLight = new LightSensor(SensorPort.S4);

NXTMotor RMotor = new NXTMotor(MotorPort.A);

NXTMotor LMotor = new NXTMotor(MotorPort.B);

@SuppressWarnings("deprecation")

Map<String, Double> coef = new HashMap<String, Double>();

int time;

boolean isParent = false;

boolean reason;

int cross = 0;

int ePr = 0;

public boolean flag = false;

int pok;

int id = 0;

static int cnt = 0;

private void pid() {

double i = 0;

double d = 0;

double pwrR = 0;

double pwrL = 0;

double err = RLight.readValue() - LLight.readValue();

i += err;

d = ePr-err;

pwrL = coef.get("c") + coef.get("kp")\*err - coef.get("kd")\*d + coef.get("ki")\*i;

pwrR = coef.get("c") - coef.get("kp")\*err + coef.get("kd")\*d - coef.get("ki")\*i;

Motor.A.suspendRegulation();

Motor.B.suspendRegulation();

RMotor.setPower((int) pwrR);

LMotor.setPower((int) pwrL);

if(RLight.readValue()<40 && LLight.readValue()<40) {

cross+=1;

Sound.systemSound(true, 0);

switch(cross) {

case 1:

RMotor.setPower(50);

LMotor.setPower(50);

RMotor.forward();

LMotor.forward();

Delay.msDelay(500);

}

}

}

public void run() {

writer w = new writer();

w.message(id + " " + pok + " " + cross + " " + "run start");

Delay.msDelay(1000);

Stopwatch sw = new Stopwatch();

reason = true;

while(cross<2) {

pid();

if(Button.ENTER.isPressed()) {

w.message(id + " " + pok + " " + cross + " " + "passed into buttonPress");

reason = false;

break;

}

}

time = sw.elapsed();

RMotor.stop();

LMotor.stop();

w.message(id + " " + pok + " " + cross + " " + "exitting run()");

}

public indi(double op, double od, double oi, double oc, int opok) {

id = cnt;

coef.put("kp", op);

coef.put("kd", od);

coef.put("ki", oi);

coef.put("c", oc);

pok = opok;

cnt++;

}

}

class experiment{

String[] coefNames = {"kp", "kd", "ki", "c"};

int maxtime = 100000;

int n;

boolean mut;

String fPath;

ArrayList<indi> curGen = new ArrayList<>();

experiment(ArrayList<indi> ar, String path, int numgen, boolean mutation) {

curGen = ar;

fPath = path;

n = numgen;

mut = mutation;

}

ArrayList<indi> selection() {

Sound.systemSound(true, 1);

writer w = new writer();

ArrayList<indi> tempGen = new ArrayList<>();

for(int i = 0; i < curGen.size(); i++) {

if(isAlive(curGen.get(i))) {

tempGen.add(curGen.get(i));

w.message("IsAlive = TRUE");

} else {

w.message("IsAlive = FALSE");

}

}

w.message("LengthAfterSelection:" + tempGen.size());

return tempGen;

}

indi mutate(indi obj) {

obj.coef.put("kp", mutateRange(obj.coef.get("kp"), obj.coef.get("kp")-0.5, obj.coef.get("kp")+0.5));

obj.coef.put("kd", mutateRange(obj.coef.get("kd"), obj.coef.get("kd")-1.5, obj.coef.get("kd")+1.5));

obj.coef.put("ki", mutateRange(obj.coef.get("ki"), obj.coef.get("ki")-0.005, obj.coef.get("ki")+0.005));

obj.coef.put("c", mutateRange(obj.coef.get("c"), obj.coef.get("c")-5, 100));

return obj;

}

double mutateRange(double coef, double min, double max) {

Random rnd = new Random();

double result = min + (max - min) \* rnd.nextDouble();

return result;

}

ArrayList<indi> cross(ArrayList<indi> gen) {

Sound.systemSound(true, 4);

ArrayList<indi> tempGen = new ArrayList<>();

writer w = new writer();

int countInd = 0;

indi i1 = null;

indi i2 = null;

for(indi ind: gen) {

if(!ind.isParent) {

if(countInd == 0) {

i1 = ind;

ind.isParent = true;

countInd++;

} else if(countInd == 1) {

i2 = ind;

ind.isParent = true;

tempGen.add(dichC(i1,i2));

countInd = 0;

}

}

}

w.message("LengthAfterCross:" + tempGen.size());

return tempGen;

}

indi dichC(indi i1, indi i2) {

indi result = new indi(0,0,0,0,i1.pok+1);

writer w = new writer();

Random rnd = new Random();

for(String tcoef : coefNames) {

w.message(tcoef);

if(rnd.nextInt(2)==0) {

result.coef.remove(tcoef);

result.coef.put(tcoef, i1.coef.get(tcoef));

} else if(rnd.nextInt(2) == 1) {

result.coef.remove(tcoef);

result.coef.put(tcoef, i2.coef.get(tcoef));

}

}

if(mut) {

result = mutate(result);

}

return result;

}

void run() {

writer w = new writer();

for(int j = 0; j<=n-1; j++) {

for(int i = 0; i<curGen.size(); i++) {

curGen.get(i).run();

w.fw(curGen.get(i), fPath);

Button.waitForAnyPress();

}

w.message("Generating new gen");

Sound.systemSound(true, 3);

curGen = selection();

w.message("CurGenLength:" + curGen.size());

curGen = cross(curGen);

Sound.systemSound(true, 2);

w.message("new gen completed");

}

}

boolean isAlive(indi ind) {

writer w = new writer();

w.message("IA TIME:" + ind.time + " REASON:" + ind.reason + " MAXTIME:" + maxtime);

if(ind.time < maxtime && ind.reason) {

Sound.playTone(1200, 2000);

return true;

} else {

Sound.playTone(500, 500);

Sound.playTone(400, 500);

Sound.playTone(300, 500);

Sound.playTone(200, 500);

return false;

}

}

}

public class main {

public static void main(String[] args) {

Button.ESCAPE.addButtonListener(new ButtonListener() {

@Override

public void buttonPressed(Button b) {

System.exit(1);

}

@Override

public void buttonReleased(Button b) {

}});

ArrayList<indi> pok1 = new ArrayList<>();

pok1.add(new indi(1.7, 3.8, 0.003, 40, 1));

pok1.add(new indi(1.6, 3.9, 0.003, 45, 1));

pok1.add(new indi(1.3, 3.6, 0.004, 70, 1));

pok1.add(new indi(0.9, 4.0, 0.001, 41, 1));

pok1.add(new indi(1.8, 3.7, 0.002, 40, 1));

pok1.add(new indi(1.5, 3.5, 0.005, 60, 1));

pok1.add(new indi(1.2, 2, 0.003, 42, 1));

pok1.add(new indi(1.1, 3.1, 0.006, 55, 1));

pok1.add(new indi(1.3, 3.6, 0.004, 38, 1));

pok1.add(new indi(1.6, 3.9, 0.03, 44, 1));

Button.waitForAnyPress();

experiment e = new experiment(pok1, "result", 10, true);

e.run();

}

}