HALAL

Ráncsik Áron

2020. Május 31.

Evolúciós algoritmus függvényközelítés

Bemeneti változók

```
GeneticAlgo(
        string input,
        int population,
        int mutation,
        int parents,
        int elitism,
        double goalFitness
);
GeneticAlgo("FuncAppr1.txt",300,5,2,100,60);
```

Implementáció

```
#include <algorithm>
#include <iostream>
#include <iomanip>
#include "GeneticAlgo.h"

GeneticAlgo::GeneticAlgo(string input, int population, int
   mutation, int parents,int elitism, double goalFitness)
{
mMutation =mutation;
mInput =input;
mPopulation=population;
mParents = parents;
mElitism=elitism;
mGoalFitness=goalFitness;
setFunction(mInput);
}
GeneticAlgo::~GeneticAlgo()
{

}
void GeneticAlgo::setFunction(string)
{
funcApproximation = FunctionApproximation();
```

```
funcApproximation.loadKnownValuesFromFile(mInput);
}
void GeneticAlgo::Solve()
{
vector<vector<double>> pop = initPopulation();

best = mGetBest(pop);
while (mCalcFitness(best) > mGoalFitness){
vector<vector<double>> newPop = mElite(pop);
while(pop.size()!= newPop.size()) {
vector<vector<double>> parents = mGetParents(pop);
vector<double> chrome = mCrossover(parents);
chrome = mMutate(chrome);
newPop.push_back(chrome);
}
pop = newPop;
best = mGetBest(pop);

for(auto b : best){
cout << b << "␣";
}
cout << "␣fit:␣␣" << mCalcFitness(best)<< endl;
}



}
vector<vector<double>> GeneticAlgo::initPopulation()
{
vector<vector<double>> pop = vector<vector<double>>();
for(int i = 0; i < mPopulation; i++){
vector<double> current = vector<double>();
for(int j = 0; j < 5; j++){
current.push_back(randomUniform(0.0f,7.0f));
}
pop.push_back(current);
}
return pop;
}
vector<double> GeneticAlgo::mGetBest(vector<vector<double>>
   population)
{
vector<double> max = vector<double>(5,0);
for(auto p : population){
double cur = mCalcFitness(p);
double old = mCalcFitness(max);
if(cur < old){
max = p;
```

```
}
}
return max;
}


vector<double>
   GeneticAlgo::mCrossover(vector<vector<double>> parents)
{
vector<double> child = vector<double>();
child.push_back(parents[1][0]);
child.push_back(parents[1][1]);
child.push_back(parents[0][2]);
child.push_back(parents[0][3]);
child.push_back(parents[0][4]);
return child;
}
vector<double> GeneticAlgo::mMutate(vector<double>
   chromosome)
{
chromosome[0] *= genMutate();
chromosome[1] *= genMutate();
chromosome[2] *= genMutate();
chromosome[3] *= genMutate();
chromosome[4] *= genMutate();
return chromosome;
}
double GeneticAlgo::genMutate()
{
double min = 1.0f - (mMutation / 100.0f);
double max = 1.0f + (mMutation / 100.0f);
double r = randomUniform(0.0f,1.0f);
return (min+(max-min)*r);
}
double GeneticAlgo::mCalcFitness(vector<double>  chromosome)
{
return funcApproximation.objective(chromosome);
}

vector<vector<double>>
   GeneticAlgo::mElite(vector<vector<double>> population)
{
sort(population.begin(), population.end(), [this](const
   vector<double> & a, const vector<double> & b) -> bool
{
return mCalcFitness(a) < mCalcFitness(b);
});
```

```
vector<vector<double>> eliteCrew =vector<vector<double>>();
for (int i = 0; i < mElitism; ++i) {
eliteCrew.push_back(population[i]);
}
return eliteCrew;
}
vector<vector<double>>
   GeneticAlgo::mGetParents(vector<vector<double>>
   population)
{
sort(population.begin(), population.end(), [this](const
   vector<double> & a, const vector<double> & b) -> bool
{
return mCalcFitness(a) < mCalcFitness(b);
});

vector<vector<double>> parents =vector<vector<double>>();
for (int i = 0; i < mParents; ++i) {
parents.push_back(population[i]);
}
return parents;
}
```

Mérési eredmények

```
0.618766 1.48905 2.05244 4.88465 5.52702  fit:  29777.6
...
...
...
0.570953 1.02322 1.96845 4.68738 5.27573  fit:  25643.6
0.561551 1.05421 1.91757 4.49947 5.01497  fit:  15236.1
0.534307 1.08964 1.85674 4.30761 5.22576  fit:  7469.78
0.516582 1.11512 1.7656 4.09491 5.04041  fit:  1939.85
0.506224 1.15044 1.69475 4.0304 4.94235  fit:  316.893
0.491449 1.10863 1.61454 3.87468 4.78747  fit:  107.631
0.490178 1.12002 1.63977 3.88042 4.55718  fit:  104.855
0.490178 1.12002 1.63977 3.88042 4.55718  fit:  104.855
0.492517 1.07546 1.57269 3.99702 4.52308  fit:  100.782
0.492517 1.07546 1.57269 3.99702 4.52308  fit:  100.782
0.492517 1.07546 1.57269 3.99702 4.52308  fit:  100.782
...
...
...
0.492806 1.14366 1.67846 3.84254 4.5927  fit:  70.1396
0.492806 1.14366 1.67846 3.84254 4.5927  fit:  70.1396
0.492806 1.14366 1.67846 3.84254 4.5927  fit:  70.1396
0.492806 1.14366 1.67846 3.84254 4.5927  fit:  70.1396
0.492806 1.14366 1.67846 3.84254 4.5927  fit:  70.1396
```

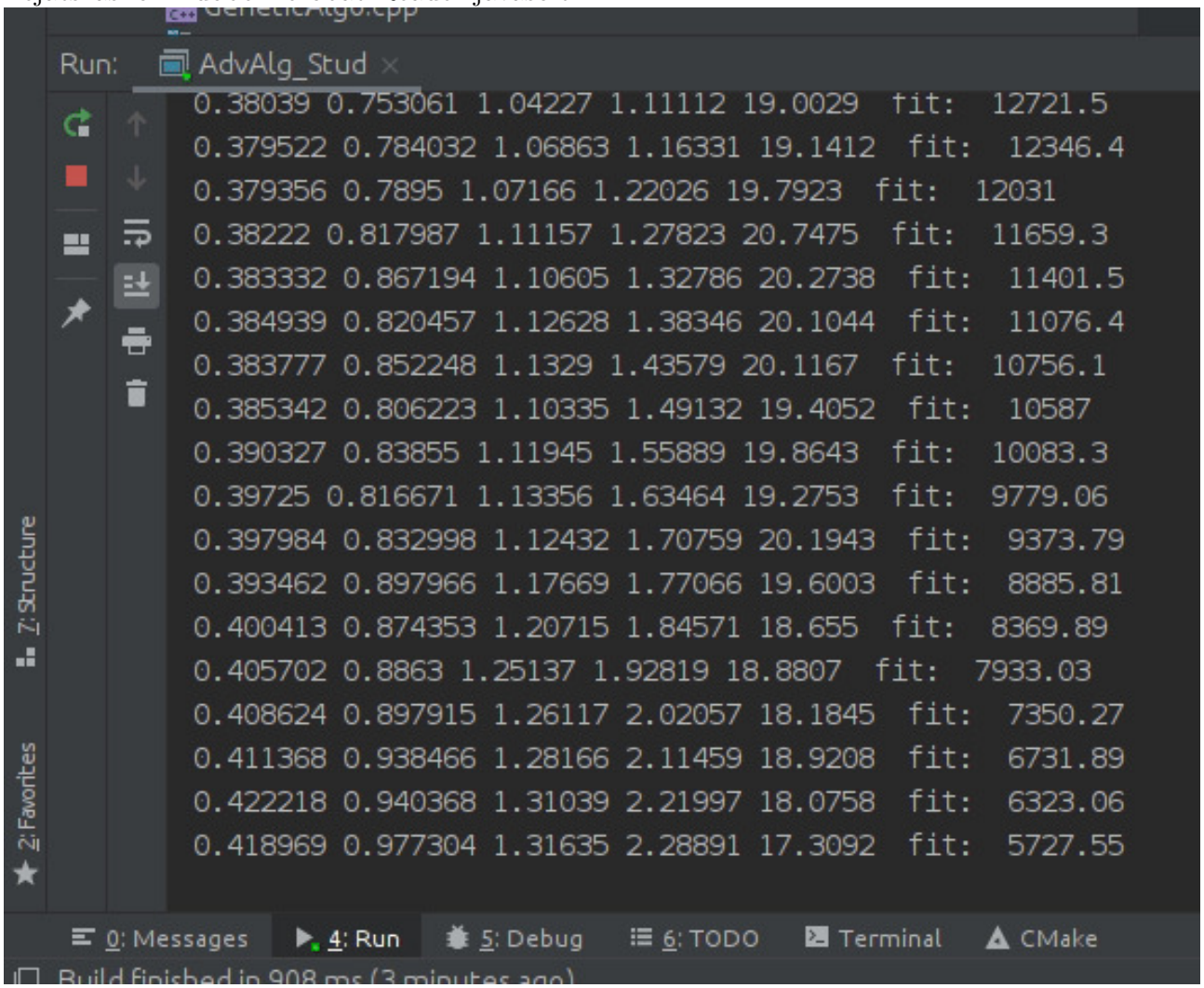
```
0.496042 1.19049 1.74759 3.80178 4.56571  fit:  52.622

Time elapsed: 131.43039 sec

Process finished with exit code 0
```

Screenshot animáció

Lejátszáshoz Adobe Acrobat Reader javasolt



Véletlen optimalizálással utazó ügynök probléma

Bemeneti változók

```
RandomOptAlgo(
        int mMean,
        int mVariance,
        double mGoalFitness,
        const string &filename
)
RandomOptAlgo(0,2,4000,"Towns.txt");
```

Implementáció

```
#ifndef RANDOMOPTALGO_H
#define RANDOMOPTALGO_H

#include "Random.h"
#include "TravellingSalesman.h"

#include <iostream>

using namespace std;

class RandomOptAlgo
{


public:
        RandomOptAlgo(
                int mMean,
                int mVariance,
                double mGoalFitness,
                const string &filename)
                : mMean(mMean), mVariance(mVariance),
                   mGoalFitness(mGoalFitness),
                   filename(filename)
```

```
                {
                        tsp = TravellingSalesmanProblem();
                        tsp.loadTownsFromFile(filename);
                }
                virtual ~RandomOptAlgo()
                {

                }
                void Solve(){
                        auto route = tsp.getTowns();
                        while (CalculateFitness(route) >
                           mGoalFitness){
                                cout << "current: fitt: " <<
                                   CalculateFitness(route)<<"\n";

                                vector<Town> newRoute =
                                   GenerateRoute(route);


                                if(CalculateFitness(newRoute)
                                   <
                                   CalculateFitness(route)){
                                        route = newRoute;
                                        cout << "fitt: "
                                 <<
                                    CalculateFitness(route)<<"\n";
                                }
                        }
                        cout << "fitt: " <<
                           CalculateFitness(route)<<"\n";
                }


private:
        int mMean, mVariance;
        double mGoalFitness;
        string filename;
        TravellingSalesmanProblem tsp;
        double CalculateFitness(vector<Town> route){
                tsp.objective(route);
        }

        vector<Town> MixRouteNormalDist(vector<Town> route)
        {
                int n = route.size();
                while (n > 1){
                        n--;
                        int shift = randomNormal(mMean,
```

```
                        mVariance);
                    int randomIndex = n + shift;
                    if(randomIndex < 0 )
                    {
                            randomIndex =0;
                    }
                    if(randomIndex >= route.size())
                    {
                            randomIndex = route.size()-1;
                    }
                    Town town = route[randomIndex];
                    route[randomIndex] = route[n];
                    route[n] = town;
                }
                return route;
        }

vector<Town> GenerateRoute(vector<Town> route)
{
        vector<Town> newRoute = route;
        newRoute = MixRouteNormalDist(newRoute);

        return newRoute;
}
};


#endif //RANDOMOPTALGO_H
```

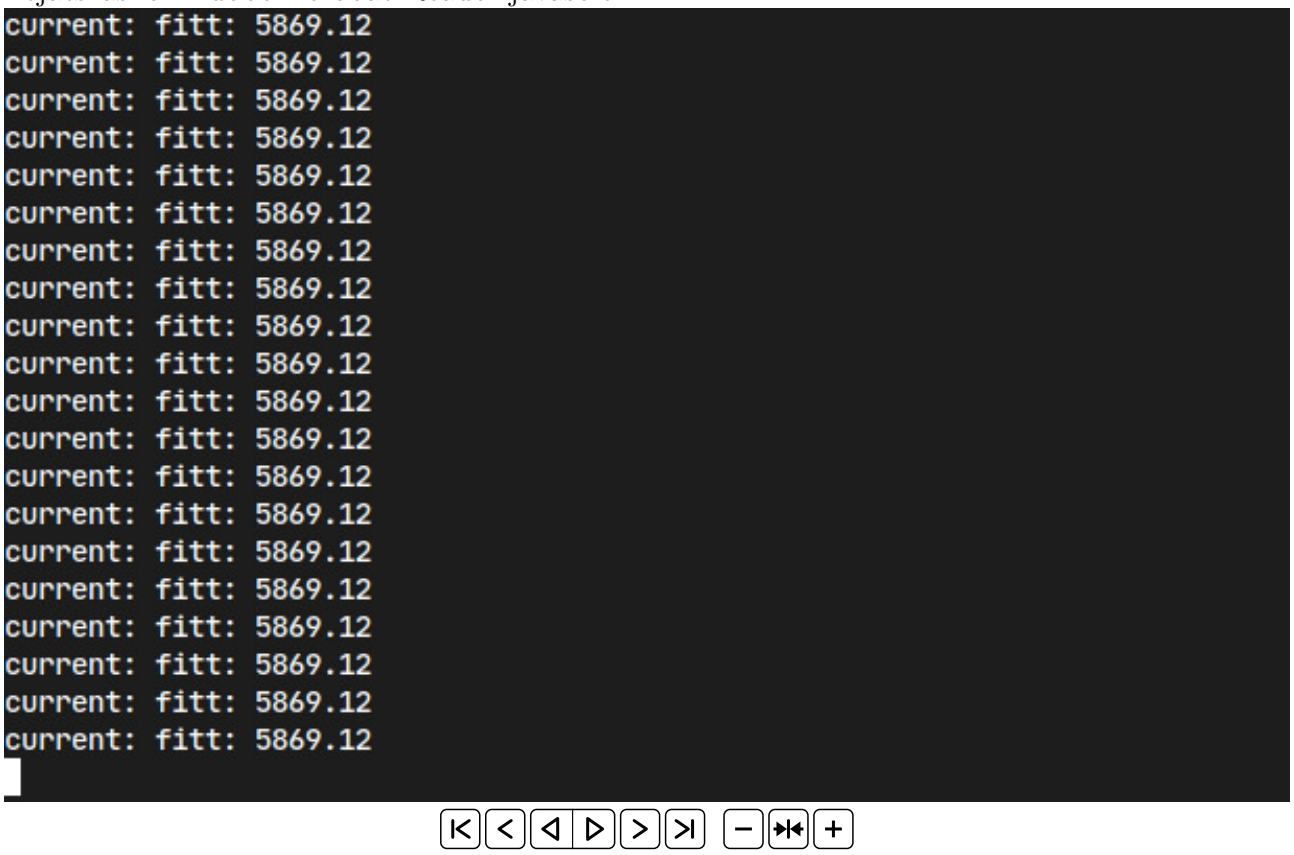
Mérési eredmények

```
...
...
...
current: fitt: 4021.08
current: fitt: 4021.08
current: fitt: 4021.08
current: fitt: 4021.08
current: fitt: 4021.08
current: fitt: 4021.08
current: fitt: 4021.08
current: fitt: 4021.08
current: fitt: 4021.08
current: fitt: 4021.08
current: fitt: 4021.08
current: fitt: 4021.08
fitt: 3972.15
fitt: 3972.15
```

```
Time elapsed: 4.49370 sec
```

Screenshot animáció

Lejátszáshoz Adobe Acrobat Reader javasolt



Hegymászó algoritmus stochasztikus megvalósítása legkisebb körülíró poligon megoldására

Bemeneti változók

```
HillClimbAlgo(
        int mEpsilon,
        int mDimension,
        int mMaxCoordinates,
        int mGoalFitness,
        const string &filename
)
HillClimbAlgo(10, 3, 400, 1, "Points.txt");
```

Implementáció

```
#ifndef HILLCLIMBALGO_H
#define HILLCLIMBALGO_H


#include <iostream>
#include "Random.h"
#include "SmallestBoundaryPolygon.h"

using namespace std;

class HillClimbAlgo
{
public:
        HillClimbAlgo(int mEpsilon, int mDimension, int
           mMaxCoordinates, int mGoalFitness, const string
           &filename)
        : mEpsilon(mEpsilon), mDimension(mDimension),
           mMaxCoordinates(mMaxCoordinates),
           mGoalFitness(mGoalFitness),
        filename(filename)
        {
```

```
                sbpp.loadPointsFromFile(filename);
        }
                ~HillClimbAlgo(){

                }
        void Solve(){

                vector<Point> polygon = InitPolygon();

                while (CalculateFitness(polygon) >
                   mGoalFitness){
                        vector<Point> newPolygon =
                           GenerateRandomPolygon(polygon);
                        if(CalculateFitness(newPolygon)
                        <
                        CalculateFitness(polygon)){
                                polygon = newPolygon;
                                cout << "new poly: \n";
                                cout << ToString(polygon) <<
                                   " fitt: "<<
                                   CalculateFitness(polygon)<<"\n";
                        }
                }
                cout << ToString(polygon) << " fitt: "<<
                   CalculateFitness(polygon);
        }
private:

        int mEpsilon, mDimension, mMaxCoordinates,
           mGoalFitness;
        string filename;
        SmallestBoundaryPolygonProblem sbpp =
           SmallestBoundaryPolygonProblem();

                double CalculateFitness (vector<Point>
                   polygon){
                double fitt = 0;
                if(sbpp.outerDistanceToBoundary(polygon) !=
                   0){
                        fitt = sbpp.objective(polygon);
                }else{
                        fitt = 10000000;
                }
                return fitt;
        }

        string ToString (vector<Point> polygon){
                string ret="";
```

```
                for(auto p : polygon){
                        ret += "(" + to_string(p.x) + "," +
                           to_string(p.y) + ")␣";
                }
                return ret;
        }

        vector<Point> GenerateRandomPolygon(vector<Point>
           polygon)
        {
                vector<Point> newPoly = vector<Point>();
                        for(auto p : polygon){
                        Point newPoint = Point();
                        newPoint.x = p.x +
                           randomUniform(-1*mEpsilon,
                           mEpsilon-1);
                        newPoint.y = p.y +
                           randomUniform(-1*mEpsilon,
                           mEpsilon-1);
                        newPoly.push_back(newPoint);
                }
                return  newPoly;
        }


        vector<Point> InitPolygon(){
                static vector<Point> polygon =
                   vector<Point>();
                for (int i = 0; i < mDimension; ++i) {
                        auto p = Point();
                        p.x =
                           randomUniform(0,mMaxCoordinates-1);
                        p.y =
                           randomUniform(0,mMaxCoordinates-1);
                        polygon.push_back(p);
                }
                return polygon;
        }
};


#endif //HILLCLIMBALGO_H
```

Mérési eredmények

```
/home/aaron/Documents/OE/6felev/halal/AdvAlg-Stud/cmake-build-debug/Adv
   2
new poly:
```

```
(46.000000,158.000000) (134.000000,283.000000)
   (341.000000,32.000000)  fitt: 478.215
new poly:
(38.000000,161.000000) (126.000000,282.000000)
   (342.000000,36.000000)  fitt: 476.988
new poly:
...
...
...
new poly:
(197.000000,221.000000) (197.000000,220.000000)
   (197.000000,221.000000)  fitt: 2
new poly:
(196.000000,227.000000) (197.000000,228.000000)
   (197.000000,228.000000)  fitt: 1.41421
new poly:
(190.000000,224.000000) (190.000000,223.000000)
   (190.000000,223.000000)  fitt: 1
Time elapsed: 0.69850 sec
```

Screenshot animáció

Lejátszáshoz Adobe Acrobat Reader javasolt