Genetic Algorithm - Prisoners Dilemma Progress

Marek Skrzypecki, INF, sem. 6, GKiO2

Arguments taken from user

Number of cycles - The amount of cycles of evolution that the program will run (as integer)

Number of individuals - The number of different individuals in population (as integer)

Number of moves - The number of moves that will be played in a single match of Iterated Prisoner's Dilemma between individuals (as integer)

Chance of crossover and mutation - Chance that crossover/mutation will occur (as double, 0.95 and 0.1 by default)

Reward, sucker, temptation, penalty - The amount of points received by individuals during a single match of Iterated Prisoner's Dilemma (as integer, 3, 0, 5, 1 by default)

Chromosome

According to plan, Chromosome is represented as 70-letter long string, with every allele being randomly C or D. The initial population consists of given number of individuals represented like this:

```
Starting Population:
CCCCDC  CDCDCDCDCDDDCDCDCCCDCCDDCDDCDDDDCCDCCDDDCDDDDDCCDCDDDDCCDDCCDDDC
CCCDDC  DDDDCCCCCCDDDCCCCCCDCCCDCCCCDDCDDDDCCDCCCCDDCCDDDDCDCDCCCDCDCDDD
DDDCDD  CCCDCCCDDCDCCDCCDDDCCCDCCDCDCCDDDCCDDCCDCCDDCCCDDDDDCDDDDDDCCCDC
CCDDDD  CDDDDCDCCDDDDDDCDDDDDDCDCCDDDCCDDDCCDDDCCCCCDCDDDDDCDCCDDDDDDDCD
DDCCDC  CDCCDDCDCDCCDDDDDDCCDDDCDCDCCCCCCCCCCDCCCCDCDDCCDCCCCDDCDDCDDCCD
CCCDDC  DCCCDCCCCCCCDCCDCCCCDDDDDDDCCDDDDDCCCCCDCCDDCCCDDDDDCDCCCDDCDDCC
DDDCDD  CDDCCCCCDDCDDDCCCDCCCCCDDCDDDDCCDCDCDCDCCCCDDDDCDDCCDCDCDCDDCDCD
DDDDDC  DCCDDCDDCCCCCDCCCDDDDCDCDDCCCDDDCDCCCDDDCCDCCDDCDDCDCCCDCDDCDDCD
CDDCDC  CCDDDDDCCDCCDCCDDCDCDCCDDDDDDCCDDDCDCDCCCCCCCCCCDCCCCDCDDCCDCCCC
.
.
.
```

Chromosome

```
class Chromosome{
private:
    string strategy;
    int rawScore;
    double fitness;
    int timesReproduced;
```

```
public:
    Chromosome();
    Chromosome(string _strategy);
    ~Chromosome();
    char& operator[](int index);
    void setFitness(double _fitness);
    double getFitness() const;
    string getStrategy();
    void display();
    void writeToFile(fstream &file);
    void reproduced();
    void addToRawScore(int score);
    int getRawScore();
    int getTimesReproduced();
};
```

Fitness Function

As planned the fitness function is simply the sum of point received by individuals during the round-robin tournament, accordingly to game payoffs (reward, sucker, temptation, penalty).

In the program it is referred to as rawScore and is a data member of Chromosome class.

Fitness Scaling

The originally chosen fitness scaling algorithm was replaced by following one:

$$f' = af + b$$

$$a = \frac{f_{avg}}{f_{avg} - f_{min}} \qquad b = -f_{min} \cdot \frac{f_{avg}}{f_{avg} - f_{min}}$$

This scaling does not only help to prevent early dominance of high scoring individuals but also distinguishes between mediocre and above average strategies in the later stages of evolution. In those stages there are few low scoring strategies, while the best scoring and average strategies have a very close raw fitness. Applying the original algorithm could result in negative fitness for those low scoring individuals.

The new algorithm adjusts the scaling coefficients to scale the weak strategies to zero and scale the other strategies as much as possible.

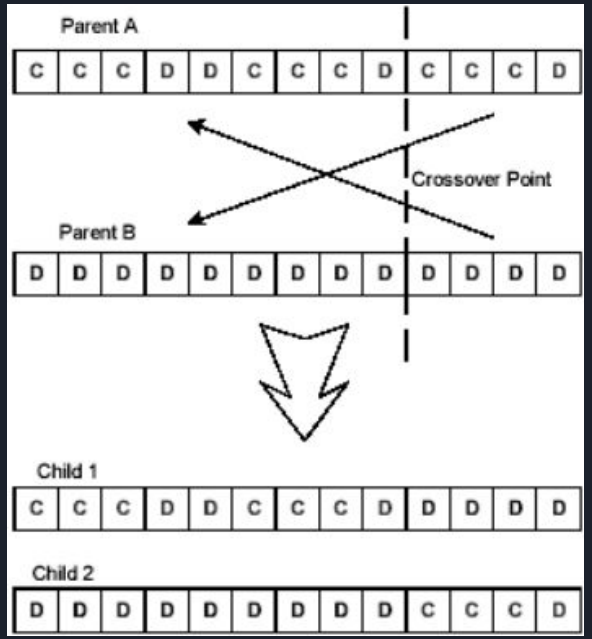
Fitness Scaling

The results of this scaling are stored as a data member of Chromosome class and referred to as fitness.

It is used to determine which individuals are most suited to reproduce.

Crossover and Mutation

Crossover and mutation were implemented as planned and didn't cause any problems.



Results

For now the results of the program's operation may be listed on the console or written to .txt file which looks like this:

```
Finished Population:
CCCCDC  CCCCDCCDCDDDCDCDDCDDDDDCCDCDDDCDCDCDCCCCCDCCDDDDCCCCDDCDDDDDDDDD
CCCCDC  CCCDDCDDCDDDCDCDDCDCDDDCCDDDCDDDCDCCCDCCCCDCDCCDCDDDDDCDDDDDDCCC
DCCCCC  CCCDDCCCCDDDCDCDDDDDDDDCCDCCDDDDCCCCDCCCCCDCDCCDCDDDCCDDDDDDDCDC
CCCCDD  CDCCCDCDCDDDCDCDCCDCDDCCDDDCDDDDCCCCDCCCCCDCCDCDCDCDDDCDCCDCCCDC
CCCCDC  CCCCDDCCCDCCCDCCDCDCDDCCDDDCDDDDCCCCDCCDCCDCCDCDCDCCDCCDDDDDDCCC
DCCCCC  CDCDCCCCCDCCCDCDDCDDDDDCCDCDDDCDCDCDCCCCCCDDCCCDCCDDCDDDDDDDDCDD
CCCCDC  CCCDDCDCCDDDDDCCDCDCDDCCDDDCDDDDCCCCDCCCCCDCCDCDCDCDDDCDCDDDDCCC
.
.
.
```

Planned to do

- Add the possibility to load the initial population from file (?)
- Optimization
- More testing
- Create a program in C# to compare resulting strategies in the Iterated Prisoner's Dilemma match

Thank you!

Author:
Marek Skrzypecki