3º trabalho de MAD

Pedro Cunha (up201406160), Rodrigo Flaminio Ribeiro (up201407863)

19/05/2017

Contents

1 Assunções e informações gerais acerca do trabalho

Para fazer diversos testes, criámos várias estratégias diferentes. Fizemos duas versões do tournment, uma em que todas as estratégias jogam duas a duas com todas as outras (inclusivé contra elas mesmas, nome RoyalRumble.py) e outra em que as várias estratégias criadas por nós jogam duas a duas contra todas as que o professor deu no enunciado (nome UsandThem.py). No evolution usámos a primeira versão previamente referida do tournment e assumimos que quando uma estratégia sai do jogo num dado turno (isto é, nenhum jogador a escolhe), tem 0 de probabilidade de voltar a jogar no próximo jogo (uma vez que a performance dessa estratégia nesse turno foi nula, visto que não participou).

2 Estratégias criadas

Foram criadas 6 estratégias. Algumas delas já existia e foram apenas implementadas. Segue a descrição de cada uma:

Strats - Joga cooperate (C) enquanto o oponente não der defect (D);

Strats2 - Joga C a menos que o oponente tenha jogado D nos ultimos dois turnos. A partir do turno 50, tem uma probabilidade proporcional ao número de jogadas total de jogar D;

Strats3 - Jogadas semelhantes às do Pavlov, mas tem 40% de probabilidade de jogar C no turno imediatamente a seguir ao oponente jogar D.

mad guy - Joga sempre D;

nice guy - Joga sempre C;

Two-Tit-For-Tat - joga D quando o adversário joga D dois turnos seguidos.

3 Primeiro Exercício

Na primeira versão do tournment, no ficheiro RoyalRumble.py, como dito, jogam todas as estratégias. Um exemplo de output de uma execução do programa é o seguinte:

1. lugar : Coop10:53.45882935834144 average is :4.112217642949341
2. lugar : Strats3:51.820242642162036 average is :3.9861725109355413
3. lugar : nice guy:51.811797752808985 average is :3.9855229040622295
4. lugar : CTFT:50.715324792853195 average is :3.9011788302194765
5. lugar : GTFT:50.58778398627448 average is :3.8913679989441907
6. lugar : Tit-For-Tat:50.44347254700659 average is :3.880267119000507
7. lugar : pavlov:50.387650739866146 average is :3.875973133835857
8. lugar : Two-Tit-For-Tat:50.08662645083956 average is :3.852817419295351
9. lugar : Strats:49.0575531986773 average is :3.773657938359792
10. lugar : Strats2:47.23407408896442 average is :3.6333903145357245
11. lugar : Random:46.53897525574062 average is :3.579921173518509
12. lugar : mad guy:33.33258252470513 average is :2.564044809592702
13. lugar : Defect10:29.556198380859556 average is :2.273553721604581

Com mais execuções do programa, os primeiros 3 lugares vão variando entre as 3 estratégias que as ocupam no output, enquanto que os últimos 2 lugares não mudam.

Na segunda versão do tournment, no ficheiro UsandThem.py, jogam as estratégias que criamos contra algumas que o professor nos deu no enunciado, nomeadamente, Coop10, Def10, Random e TFT. Segue um exemplo de output após uma execução:

1. Strats3 17.00774754346183average is : 3.4015495086923657
2. nice guy 16.46358723623263average is : 3.292717447246526
3. Strats2 14.90566378066378average is : 2.981132756132756

4. Strats 14.71485067040999average is : 2.9429701340819983

5. mad guy 9.546264129595832average is : 1.9092528259191663

Os 2 primeiros lugares são sempre do Strats3 e do nice guy, embora o Strats3 ganhe mais vezes. O último lugar é sempre do mad guy.

4 Segundo Exercício

O programa evolution está implementado no ficheiro Evolution.py. As estratégias que o professor apresentou no enunciado foram implementadas e usadas na competição. A estratégia Generous Tit-for-Tat (GTFT) foi implementada de tal modo que, quando o oponente joga D há uma chance de 40% do GTFT jogar D.

Começando 5 jogadores com cada estratégia e tendo 12 estratégias em jogo, temos um total de 60 jogadores. Iterámos em 100 rondas, mas entre 40 e 50 rondas o ecossistema estabiliza (isto é, todos os jogadores usam a mesma estratégia). Ao fim de várias rondas de evolution, tivemos sempre diversos vencedores, pelo que não há nada a concluir sobre as estratégias. O evolution foi implementado por forma a que, no final de cada iteração, gera-se uma probabilidade com base no score total (para cada jogador) tal que a estratégia dele é decidida conforme o desempenho acumulado de todos os outros.

```
if(iteration!=0): # update strats used
    aux=[]
    for i in range(0,n_players): # for each player
        a = random.uniform(0,scoresum)
        temp=0
        for j in range(0,n_players): #find new strat
            temp+=board[j][0] #score from previus iteration
            if(temp >= a):
                aux.append([Strat[j],Name[j]])
                break
    for i in range(0,n_players):  # update players
        Score[i]=0
        Strat[i]=aux[i][0]
        Name[i]=aux[i][1]
```

5 Terceiro Exercício

Escolhemos a estratégia Strat3, referida anteriormente.

```
def Strat3(own, oth):
    if(len(own)==0):
        return "C"
    elif(oth[-1]=="C"):
        return own[-1]
    else:
        if(random.uniform(0, 1)<0.1):
            return "C"
        return "D"
```