Evolution of Complexity

Laiyuan Zhang
31035698
lz4y19@soton.ac.uk

January 2019

Contents

1 Introduction

It is easy for us to know that cooperative behaviours could benefit the whole group by looking around. For example, bacteria could use resource efficiently by growing at a reduced rate [1–4]. Cooperative individuals value long-term team benefits rather than current personal benefits. However, selfish individuals tend to survive more easily cause they use more resource. So, here is the difficulty in explaining the existence of collaborators and they are much more than selfish individuals in nature [5].

In previous work, parameters which benefit cooperative behaviours were fixed. The model we reimplement allows selections to operate based on the conditions themselves. This model is mainly composed of two parts, one is large groups and the other is small groups. Allocating resources for each group is determined by the size of each group. The large groups will have more resources than small groups. The individuals will have a preference for attending small group ore large group. Also, cooperative individuals have lower growth and lower resource consumption compared with selfish individuals.

What we need to know is there is still the possibility that the cluster of cooperators might be exploited by the mutant selfish individuals [3]. Wilson has offered his famous trait-group aggregation and dispersal model which could purge selfish individuals effectively [6–8]. This is the reason why we need to return the progeny of each group to the migrant pool [5]. Through the experiment, Powers et al. proved cooperators which favour small groups will survive in the final.

1.1 Model Details

The parameters in Table 1 are all the parameters we will use to implement the experiment.

Parameter	Value
Growth rate (cooperative), Gc	0.018
Growth rate (selfish), Gs	0.02
Consumption rate (cooperative), Cc	0.1
Consumption rate (selfish), Cs	0.2
Population size, N	4000
Number of generations, T	120
Number of timesteps, t	4
Death rate, K	0.1
Resource for small groups, R_small	4
Resource for large groups, R_large	50
Small group Size,Ns	4
Large group Size,Nl	40

Table 1: Parameter settings used throughout.

As I said above, there are two types of group in the model, one is large and the other one is small. And in each group, there will be two genotypes, cooperative and selfish. So, there are four combines in total, cooperative + small, cooperative + large, selfish + small and selfish + large.

The first step in the experiment is initialisation. Individuals will be evenly distributed to each combination.

In the second step, we will form small and large groups according to the preference of individuals in the migrant pool. And this preference will change over time.

Next, the different groups will allocate resources and perform reproduction. We will use equation (1) to calculate the resources each genotype could get. Then, we can get the population of new generation according to equation (2).

$$r_i = \frac{n_i G_i C_i}{\sum_j n_i G_i C_i} R \tag{1}$$

$$n_i(t+1) = n_i(t) + \frac{r_i}{C_i} - K n_i(t) \tag{2}$$

After the reproduction, all the individuals and their progeny will go back to the migrant pool. We then need to rescale the migrant pool back to size N to maintain the global carrying capacity. Of course, we will keep the proportion of each genotype during the process of rescaling.

In this experiment, these steps will iterate 120 times.

2 Reimplementated Results

Based on the paper,it was decided to set the small group size at 4 and the large at 40. Here are the results of reimplementation.

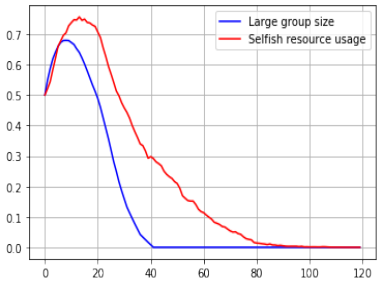

Figure 1: reimplementated results

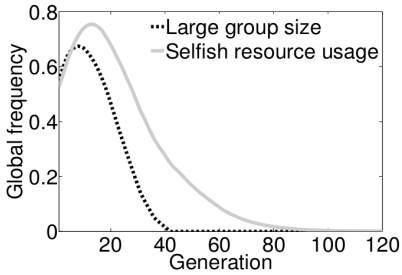

Figure 2: original results

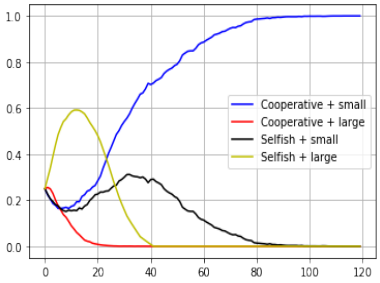

Figure 3: reimplementated results

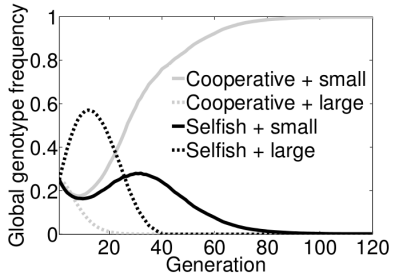

Figure 4: original results

As we can see from the figures above, the reimplementation results are very close to the original results. We can learn from the first two figures that large group size and selfish behaviours are favoured at the beginning. However, around the 15th generation, the individuals favouring small group size and cooperative behaviours started to increase. About 40th generation, all the individuals favour small groups and around 80th generation, selfish individuals eliminated.

The last two figures show the proportions of each combine. The selfish individuals in large groups quickly dominated the population. However, just

after 15 generations, the proportions started to decrease and cooperative individuals in small groups started to flourish. Cooperative individuals in small groups dominated the population completely around 80th generation.

This model proves that cooperative behaviours and small group size could evolve via individual selection whatever their initial preferences when cooperative and selfish strategies and the environmental conditions are both adaptable [5].

3 Extension

The model we reimplemented is quite simple, the death rate and growth rate are all fixed, which is impossible in the nature. So, as the extension part, I will try to change these parameters to make the model more close to the nature.

Gompertz has revealed that the mortality rate of infants and young children is generally relatively high, and then it will decline year by year. The mortality rate reaches the lowest point at about 10-15 years old; Mortality will soon rise again, with the rate of increase doubling approximately every 8 years (sometimes 10 years), a trend that can continue until the age of 80. The same rule applies to other ethnic groups in nature [9].

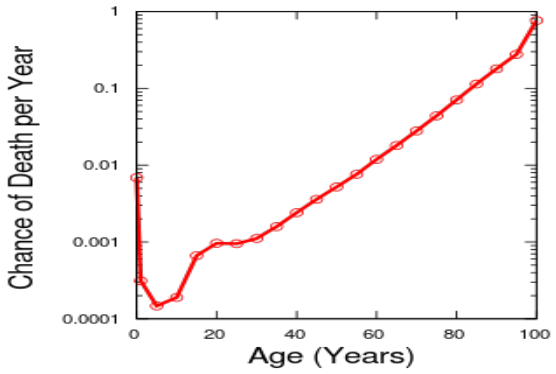

Figure 5: Gompertz–Makeham law of mortality

To simple the model, I will define a deathRate list including 4 number corresponding to timesteps to simulate individual mortality at different

stages.

$$deathRate = [0.1, 0.01, 0.001, 0.2] \tag{3}$$

Despite this, the large environment also affects group mortality and growth rates, such as seasonal changes, differences in dimensions, whether there is influenza, etc. To facilitate verification, we just use the sine and cosine functions. For example, in Spring, the growth rate will grow which means the environment is benefits the reproduction with mortality rate declining. So we can use these two opposite functions to represent the impact of the environment on them. The formulas for allocating resources and growing populations will therefore change accordingly.

$$r_i = \frac{n_i G_i (1 + sin(T)) C_i}{\sum_j n_i G_i (1 + sin(T)) C_i} R \tag{4}$$

$$n_i(t+1) = n_i(t) + \frac{r_i}{C_i} - deathRate[timestep](1 + cos(T)) n_i(t) \tag{5}$$

Because of the influence of the environment, the group size will also change. We know that as the size of the group increases, so does its ability to resist risks.

$$Ns + (sin(T) - cos(T))) \tag{6}$$

$$Nl + (sin(T) - cos(T))) \tag{7}$$

From the formula above, the range of small groups 3-5 and the range of the large group is 49-51. Relative to their respective bases (4 and 50), changes in small groups are much larger than large groups.

My hypothesis is the overall trend is similar but the proportion of different genotype evolution have periodic floating because of the growth rate and mortality generally cyclical changes, and this is due to the niche construction.

4 Results of extension

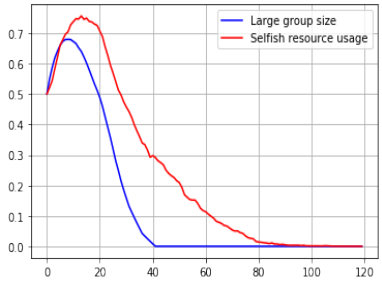

Figure 6: reimplementated results

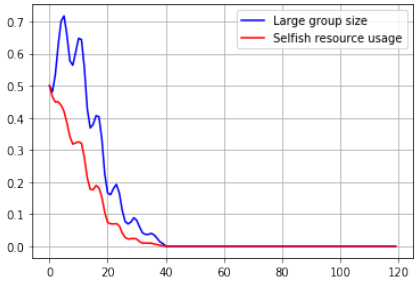

Figure 7: extension results

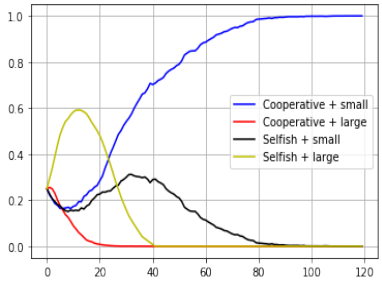

Figure 8: reimplementated results

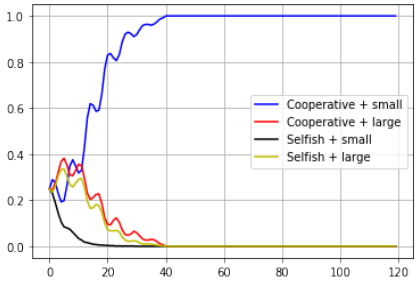

Figure 9: extension results

We can see that the overall trend is similar as we guessed. In the process of evolution, the proportion of each genotype does show a cyclical change. However, we can see from figure 6 and figure 7 that selfish individuals started to decline at the beginning which is different from the experiment we reimplement which means the initial environmental conditions favour cooperation.

5 Conclusion

This paper mainly reimplement the experiment of Powers et al. The experiment proves that although selfish individuals could exploit cooperative individuals, but it will eventually move towards cooperation + small based on the niche construction and the ways of purging cheaters. The extension part, I investigated individual mortality and the effect of the overall environment

on group growth and mortality on evolution. And the experiment proves that only if the niche construction keeps the same in general, although the evolution process will fluctuate with the influence of these factors, the overall evolution direction is still developing towards the genotypes favored by the environment.

However, my idea of the impact of the environment on growth rate and mortality, and the impact of growth rate and mortality on group size is a very rough idea. The mathematical model is also very simple. So this model still cannot really represent the real situation in nature. In the future, I may use some more accurate mathematical models to investigate this problem.

References

[1] T. Pfeiffer, S. Schuster, and S. Bonhoeffer, "Cooperation and competition in the evolution of atp-producing pathways," *Science*, vol. 292, no. 5516, pp. 504–507, 2001.

[2] T. Pfeiffer and S. Bonhoeffer, "An evolutionary scenario for the transition to undifferentiated multicellularity," *Proceedings of the National Academy of Sciences*, vol. 100, no. 3, pp. 1095–1098, 2003.

[3] J.-U. Kreft, "Biofilms promote altruism," *Microbiology*, vol. 150, no. 8, pp. 2751–2760, 2004.

[4] J.-U. Kreft and S. Bonhoeffer, "The evolution of groups of cooperating bacteria and the growth rate versus yield trade-off," *Microbiology*, vol. 151, no. 3, pp. 637–641, 2005.

[5] S. T. Powers, A. S. Penn, and R. A. Watson, "Individual selection for cooperative group formation," in *European Conference on Artificial Life*. Springer, 2007, pp. 585–594.

[6] D. S. Wilson, "A theory of group selection," *Proceedings of the national academy of sciences*, vol. 72, no. 1, pp. 143–146, 1975.

[7] ——, *The natural selection of populations and communities*. Benjamin/Cummings Pub. Co., 1980.

[8] E. Sober and D. S. Wilson, *Unto others: The evolution and psychology of unselfish behavior*. Harvard University Press, 1999, no. 218.

[9] B. Gompertz, "Xxiv. on the nature of the function expressive of the law of human mortality, and on a new mode of determining the value of life contingencies. in a letter to francis baily, esq. frs &c," *Philosophical transactions of the Royal Society of London*, no. 115, pp. 513–583, 1825.

6 Code

6.1 Reimplemention Part

```
import numpy as np
import matplotlib.pyplot as plt
import random
import math

#all list,like migrantPool,resource,1st index is the resource
    allocated to cs(coopeartive small), 2nd is cl, 3rd is ss, 4th
    is sl
K = 0.1
# death rate
R_small = 4
# Resource for small group
R_large = 50
# Resource for large group
Gc = 0.018
#Growth rate (cooperative)
Gs = 0.02
#Growth rate (selfish)
Cc = 0.1
# Consumption rate (cooperative)
Cs = 0.2
# Consumption rate (selfish)
N = 4000
# Population size
T = 120
#Number of generations
Ns = 4
#small group size
```

```
Nl = 40
#large group size
migrantPool = [1000.0,1000.0,1000.0,1000.0]
#inital migrantPool,total 4000
resourcesPool = [0.0,0.0,0.0,0.0]
coopSmallGroupsFre = list();
coopLargeGroupsFre = list();
selfSmallGroupsFre = list();
selfLargeGroupsFre = list();
largeGroupFre = list();
selfGroupFre = list();
t = 4
# timesteps

def calculateResource(group, resource):
    total = (group[0] * Gc * Cc) + (group[1] * Gc *Cc) + (group[2]
        * Gs * Cs) + (group[3] * Gs * Cs)
    resourcesPool[0] = (group[0] * Gc * Cc * resource) / total
    resourcesPool[1] = (group[1] * Gc * Cc * resource) / total
    resourcesPool[2] = (group[2] * Gs * Cs * resource) / total
    resourcesPool[3] = (group[3] * Gs * Cs * resource) / total
    return resourcesPool

def calculatePopulation(group, resourcesPool):
    group[0] = group[0] + resourcesPool[0] / Cc - x * group[0]
    group[1] = group[1] + resourcesPool[1] / Cc - K * group[1]
    group[2] = group[2] + resourcesPool[2] / Cs - K * group[2]
    group[3] = group[3] + resourcesPool[3] / Cs - K * group[3]
    return group

for i in range(T):
    #record the frequency for ploting
    coopSmallGroupsFre.append(migrantPool[0] / N )
    coopLargeGroupsFre.append(migrantPool[1] / N )
    selfSmallGroupsFre.append(migrantPool[2] / N )
    selfLargeGroupsFre.append(migrantPool[3] / N )
    largeGroupFre.append((migrantPool[1]+migrantPool[3]) / N )
    selfGroupFre.append((migrantPool[2]+migrantPool[3]) / N )

    #create groups according to the proportions
```

```
#and divided them into small group and large group for the
    convenience of calculating population
smallGroupNumber = int(( migrantPool[0] + migrantPool [2]) / Ns
    )
largeGroupNumber = int(( migrantPool[1] + migrantPool [3]) / Nl
    )
smallGroups = list()
largeGroups = list()
if(smallGroupNumber):
    small_cooperative_rate = migrantPool[0]/(migrantPool[0] +
        migrantPool[2])
if(largeGroupNumber):
    large_cooperative_rate = migrantPool[1]/(migrantPool[1] +
        migrantPool[3])
for i in range(smallGroupNumber):
    group = [0,0,0,0]
    for i in range(Ns):
        if(random.random() < small_cooperative_rate):
            group[0] += 1
        else:
            group[2] += 1
    smallGroups.append(group)
for i in range(largeGroupNumber):
    group = [0,0,0,0]
    for i in range(Nl):
        if(random.random() < large_cooperative_rate):
            group[1] += 1
        else:
            group[3] += 1
    largeGroups.append(group)

# reproduction
for group in smallGroups:
    for i in range(t):
        smallGroupResource = calculateResource(group, R_small)
        calculatePopulation(group, smallGroupResource)
for group in largeGroups:
    for i in range(t):
        largeGroupResource = calculateResource(group, R_large)
        calculatePopulation(group, largeGroupResource)
```

```
    # Rescale the migrant pool
    migrantPool = [0.0, 0.0, 0.0, 0.0]
    for group in (smallGroups):
        for i in range(4):
            migrantPool[i] += group[i]
    for group in (largeGroups):
        for i in range(4):
            migrantPool[i] += group[i]
    a = N / sum(migrantPool)
    for i in range(4):
        migrantPool[i] = migrantPool[i] * a
    print(migrantPool)

plt.plot(range(T), largeGroupFre, 'b', range(T), selfGroupFre, 'r')
plt.grid(True)
plt.gca().legend(('Large group size','Selfish resource usage'))
plt.plot(range(T), coopSmallGroupsFre, 'b', range(T),
    coopLargeGroupsFre, 'r',
         range(T), selfSmallGroupsFre, 'k', range(T),
             selfLargeGroupsFre, 'y')
plt.grid(True)
plt.gca().legend(('Cooperative + small','Cooperative +
    large','Selfish + small','Selfish + large'))
```

6.2 Extension Part

```
def newCalculatePopulation(group, resourcesPool, T, deathRate):
    group[0] = group[0] + resourcesPool[0] / Cc - deathRate *
        (1+math.cos(T)) * group[0]
    group[1] = group[1] + resourcesPool[1] / Cc - deathRate *
        (1+math.cos(T)) * group[1]
    group[2] = group[2] + resourcesPool[2] / Cs - deathRate *
        (1+math.cos(T)) * group[2]
    group[3] = group[3] + resourcesPool[3] / Cs - deathRate *
        (1+math.cos(T)) * group[3]
    return group
```

```
def newCalculateResource(group, resource, T):
    total = (group[0] * (Gc * (1+math.sin(T))) * Cc) + (group[1] *
        (Gc * (1+math.sin(T))) * Cc)+ (group[2] * (Gc *
        (1+math.sin(T))) * Cs) + (group[3] * (Gc * (1+math.sin(T)))
        * Cs)
    resourcesPool[0] = (group[0] * (Gc * (1+math.sin(T))) * Cc *
        resource) / total
    resourcesPool[1] = (group[1] * (Gc * (1+math.sin(T))) * Cc *
        resource) / total
    resourcesPool[2] = (group[2] * (Gc * (1+math.sin(T))) * Cs *
        resource) / total
    resourcesPool[3] = (group[3] * (Gc * (1+math.sin(T))) * Cs *
        resource) / total
    return resourcesPool

deathRate = [0.1,0.01,0.001,0.2]
for tt in range(T):
    #record the frequency for ploting
    coopSmallGroupsFre.append(migrantPool[0] / N )
    coopLargeGroupsFre.append(migrantPool[1] / N )
    selfSmallGroupsFre.append(migrantPool[2] / N )
    selfLargeGroupsFre.append(migrantPool[3] / N )
    largeGroupFre.append((migrantPool[1]+migrantPool[3]) / N )
    selfGroupFre.append((migrantPool[2]+migrantPool[3]) / N )

    #create groups according to the proportions
    #and divided them into small group and large group for the
        convenience of calculating population
    smallGroupNumber = int(( migrantPool[0] + migrantPool [2]) /
        (Ns+(math.sin(tt)-math.cos(tt))) )
    largeGroupNumber = int(( migrantPool[1] + migrantPool [3]) /
        (Nl+(math.sin(tt)-math.cos(tt))) )

    smallGroups = list()
    largeGroups = list()
    if(smallGroupNumber):
        small_cooperative_rate = migrantPool[0]/(migrantPool[0] +
            migrantPool[2])
    if(largeGroupNumber):
```

```
    large_cooperative_rate = migrantPool[1]/(migrantPool[1] +
        migrantPool[3])
for i in range(smallGroupNumber):
    group = [0,0,0,0]
    for i in range(int(Ns+(math.sin(tt)-math.cos(tt)))):
        if(random.random() < small_cooperative_rate):
            group[0] += 1
        else:
            group[2] += 1
    smallGroups.append(group)
for i in range(largeGroupNumber):
    group = [0,0,0,0]
    for i in range(int(Nl +(math.sin(tt)-math.cos(tt)))):
        if(random.random() < large_cooperative_rate):
            group[1] += 1
        else:
            group[3] += 1
    largeGroups.append(group)
# reproduction
for group in smallGroups:
    for i in range(t):
        smallGroupResource = newCalculateResource(group,
            R_small,tt)
        newCalculatePopulation(group,
            smallGroupResource,tt,deathRate[i])
for group in largeGroups:
    for i in range(t):
        largeGroupResource = newCalculateResource(group,
            R_large,tt)
        newCalculatePopulation(group,
            largeGroupResource,tt,deathRate[i])

# Rescale the migrant pool
migrantPool = [0.0, 0.0, 0.0, 0.0]
for group in (smallGroups):
    for i in range(4):
        migrantPool[i] += group[i]
for group in (largeGroups):
    for i in range(4):
        migrantPool[i] += group[i]
```

```
    a = N / sum(migrantPool)
    for i in range(4):
        migrantPool[i] = migrantPool[i] * a
    print(migrantPool)

plt.plot(range(T), largeGroupFre, 'b', range(T), selfGroupFre, 'r')
plt.grid(True)
plt.gca().legend(('Large group size','Selfish resource usage'))
plt.plot(range(T), coopSmallGroupsFre, 'b', range(T),
    coopLargeGroupsFre, 'r',
         range(T), selfSmallGroupsFre, 'k', range(T),
             selfLargeGroupsFre, 'y')
plt.grid(True)
plt.gca().legend(('Cooperative + small','Cooperative +
    large','Selfish + small','Selfish + large'))
```