Evolution of Cooperation

Alun Meredith

January 7, 2016

Abstract

In this paper we test the reproducibility of Powers' work [1] on individual selection for cooperative group formation. In which Powers et. al. build a model to show that selection can not only operate in favour of cooperative individuals but selection can act on the environmental conditions which enable evolution to show the conditions which favour cooperation are evolvable despite being initially selected against.

We then expand on this model exploring a critique that the binary model may be the source of many of the features observed and hypothesising a coevolutionary algorithm style arms race in a more continuous system. However we observe consistent decrease in selfishness despite the environmental conditions selecting for it, supporting Powers' arguement that "any component of selection on structuremodifying traits that is due to social behaviour must be in the direction of increased cooperation" [10].

1 Introduction

Cooperation can be defined as individuals taking some behaviour which doesn't directly benefit themselves in order to benefit the group as a whole; often tied to the idea that cooperative individuals may look for long term rewards at the expense of short term rewards.

It is possible to try and explain cooperation through other mechanisms such as product of communication for intelligent animals. Or when the organism lives long enough to see the yield of cooperation within the time-frame of reproductive cycles cooperation improves immediate fitness so can be considered an optimal action in the time scale evolution acts upon. However cooperative behaviour is observed through many different levels of biological complexity and lifespans such as bacteria moderating their consumption of a limited resource [2, 3, 4, 5]. Through examples like these it can be inferred cooperation may be an evolved trait.

The idea that cooperation can emerge as an evolutionary trait is not a trivial one. One of the core features of evolution is that selection acts upon what is immediately beneficial regardless of the future so individuals with higher immediate benefit will be selected for. Therefore greedy individuals maximising immediate gains will be selected for.

To produce an evolutionary model which allows for the emergence of cooperative behaviour the positive effect of cooperating must outweigh the selective pressure to be greedy. Consider two isolated populations of individuals with no method of variation, one of greedy individuals and one of cooperative ones. The cooperative population would consume a limited resource more efficiently than a group population of only greedy individuals and therefore have higher fitness than the greedy population. Therefore it can be said that spatial structures, like isolated groups can allow cooperative behaviour to thrive [2, 3, 4, 6].

The example above only worked by denying variation, especially through mutation to occur. This is because the presence of a single greedy individual in the cooperative group would corrupt the entire group to be cooperative by yielding the rewards of the cooperative groups actions while not suffering the reduced growth rate of the cooperative individuals. It has been proposed that continuously breaking up the groups and reforming them can offset the emergence of greedy mutations by creating some groups with no greedy individuals [7, 8, 9].

It can be considered that the balance of cooperative to greedy individuals therefore is based on the between individual selection to produce greedy individuals and the between group selection to produce cooperative groups.

The paper tries to produce conditions allowing cooperation to be selected for despite being initially selected against. In addition it allows the conditions which allow for cooperation themselves to be selected for by having an allele governing the size of the group each individual is placed in. With larger groups having less sampling error and therefore lower chance to have none/few greedy individuals.

1.1 Model

The model takes the above concepts having populations of individuals form isolated groups and then grow based on their consumption of a limited resource shared to each group before dispersing and reforming the group every (t) generations.

The individuals in the model do not mutate and asexually reproduce, creating copies of themselves. Additionally the fitness of the individual is dictated by their ability consume a limited resource which is shared with individuals in the group.

There are genotypes of two binary loci. The first gene codes if the individual is greedy or cooperative, with greedy individuals requiring a larger amount of the resource to reproduce (C), but also taking a larger proportion of the resource. The second loci dictates the size of the group the individual will be placed in when group formation occurs.

The four genotypes are initially distributed evenly before being randomly sampled without replacement into groups of their preferred size any remaining individuals are discarded, which should have negligible effect for large population size. This produces groups with variance in their ratio of greedy and cooperative individuals due to sampling error. The groups are left to grow for t generations before the population is normalised back to N individuals and re-sampled into new groups. This cycle is repeated until T generations have transpired.

During each generation the genotypes grow in number and a fraction of them die (K). Each group is given a total amount of resource (R) based on the size of the group. Large groups are given slightly more resource per individual than small groups to ensure large groups are initially selected for.

The share of the groups resource each individual gets is based on the product of their growth rate and consumption rate relative to the other individuals in the group such that (1) shows the amount of resource each individual will receive:

Parameter		**Value**
Growth rate (cooperative)	G_c	0.018
Growth rate (selfish)	G_s	0.02
Consumption rate (cooperative)	C_c	0.1
Consumption rate (selfish)	C_s	0.2
Population size	N	4000
Number of generations	T	120
Generations between groups	t	4
Death rate	K	0.1
Resource per group (large)	R_l	50
Resource per group (small)	R_s	4

Table 1: Parameters used to reproduce Powers' findings [1]

$$r_i = \frac{G_i C_i}{\sum_j (G_j C_j)} R \tag{1}$$

The consumption rate C represents the amount of resource required to reproduce whereas the growth rate G represents a weight for the amount of resource that phenotype takes in combination with consumption rate. So the number of times an individual reproduces is $\frac{r}{C}$. We can aggregate these changes to show the change in number of individuals belonging to a genotype n_i in one generation, defined in (2).

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

1.2 Reproduction

The reproducibility of Powers' results have been tested [1], starting with the analysis of equilibrium states as a function of the parameters *initial group size* and *time spent in groups before mixing* (t).

In figure 1 we can see the same general shape in both Powers' and the reproduction graphs however there is some variance. The reproduction appears to systematically favour cooperative individuals slightly, most noticeably when the initial group size is two. Part of this could be explained by sampling error the reproduction only had 5 repetitions

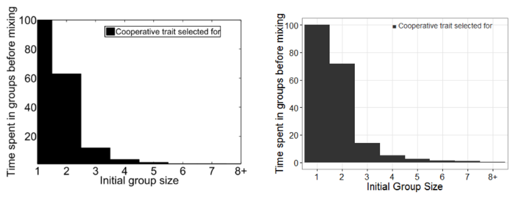

Figure 1: Equilibrium state of cooperative and greedy individuals as a function of initial group size and number of generations before mixing. Black regions show parameter combinations which select for cooperative trait; white areas for greedy trait. Left graph shows the original [1] and right graph shows the reproduction.

which could lead to some noticeable variance. However the difference in the two results appears systematic, one hypothesis for this is the way in which rounding is treated in both algorithms. The reproduction model aggregates decimals until after the groups are about to be reformed. By doing this artefacts caused by rounding thresholds in the small group size are far less likely. It is not known how Powers et. al. treats rounding in their model and was the biggest barrier to a perfect reproduction while building the model.

The purpose of this figure is to find appropriate values for group size, ensuring the large group favours greedy individuals while the small group favours the cooperative individual. It is interesting that the small group was chosen to be size 4 with time between group mixing also 4 as this point is on or very close to an equilibrium point in Powers' reproduction. While it does favour cooperative individuals it would be safer to pick a value not on the equilibrium threshold.

Figure 2 from Powers' paper has also been reproduced. Here the differences between the reproduction and original are far more subtle and hard to identify. It appears as though the slight differences in construction which lead to figure 1 being different have little impact on the overall dynamics of the system.

Figure 2 (right) shows how the different genotypes change frequency over time. As expected from fig:1 the selfish large genotype outcompetes the cooperative and large, while the cooperative and small outcompetes the selfish and small. In addition due to the extra resource given to the large

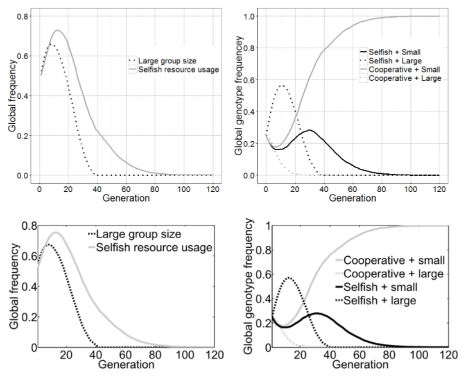

Figure 2: Left hand plots shows the prevalence of each allele over time. Right hand plots shows the prevalence of each genotype over time. Top images show the original [1], bottom plots the reproduction.

group size the selfish large genotype grows much more rapidly at first until the population of cooperative and large individuals on which it was exploiting becomes extinct. At this point they too quickly become extinct leaving the cooperative and small genotype to flourish. The selfish and small genotype lasts much longer than the other dominated genotypes, because there is such a proportion of cooperative individuals each selfish individual has a larger advantage than in an even environment narrowing the gap between the two but ultimately still decreasing in population.

In essence the above model is a competitive coevolutionary model as the fitness of each individual is based only on their ability to compete with the other individuals in the population for a fixed resource, for example if a slightly more greedy phenotype was introduced under the same conditions you would expect the current selfish phenotype to be dominated by that individual as it is out-competed.

Powers' shows a model for which selection acts on the conditions which produce cooperative behaviour i.e. the favoured group size to produce cooperative behaviour despite being initially selected against. While it achieves this there is no inclusion of variation in the model, in particular no mutation which is surprising considering the purpose of using Wilson's trait group aggregation and dispersal method is to counter the effects of mutation in the population. Although similar models have been shown to work with mutation [10]

2 Extension

For the extension of this Paper we consider in more detail some of the limitations of Power's model. In Powers' thesis it is argued that "any component of selection on structuremodifying traits that is due to social behaviour must be in the direction of increased cooperation" [10]. We would like to test probe this hypothesis in more detail. Consider in figure 2 the reason for the selfish and large phenotype to become extinct was that it no longer had less selfish individuals to exploit. We therefore hypothesise that the model will fail upon introducing a pseudo-continuous set of cooperative traits with mutation. Here there will be a non-binary distribution of cooperation in individuals so that in cases where selfishness is rewarded selfish individuals can continue to exploit the more cooperative individuals in their distribution.

Making the analogy to a co-evolutionary model where it is expected the cooeprative trait will experience an arms race until it reaches the ceiling. It is this point we see in Powers' model similar to disengagement when there is no longer a more cooperative part of the population to exploit. During this arms race efficiency of consumption (C) would continue to deteriorate so we could expect a crash premature to reaching the ceiling.

Consider an adaptation to Powers' model for figure 2. The individuals still have a loci with two traits for either large or small group size but 11 traits for cooperation. The values for consumption rate and growth rate scale linearly between the selfish and cooperative values in table 1. Before redistributing the population to groups allow random mutation of the cooperative loci up or down 1 step along the chain at a rate of $M = 0.1$.

3 Results

In this section we reproduce figures 1 and 2 for our new model. Exploring the conditions chosen in figure 1 to find the areas of dominance of different levels of greed. The parameters for C_i used is the sequence from $0.1 - 0.2$ by 0.01 and for G_i is the sequence from $0.018 - 0.02$ by 0.002. Population size, N has also been reduced to 1000 to improve computing performance.

After redistributing the population to the 11 factors of greed figure 3 showing the change in dominant genotype over time shows some notable similarities and differences to fig.1. The general structure of the dark cooperative levels (Greed = 1 or 2) is very similar to the binary equilibrium states so it can be concluded that the conditions supporting highly cooperative action are consistent.

The level of cooperation increases again after the initial group size of 5. This effect is very subtle, with only 5 repeat runs taken there could be significant sampling error here and would need exploring in more detail with a larger variety of parameters. However this does indicate an effect that ever increasing group sizes may not be uniforms detrimental to cooperativity supported by those parameters, especially when time between mixing kept low.

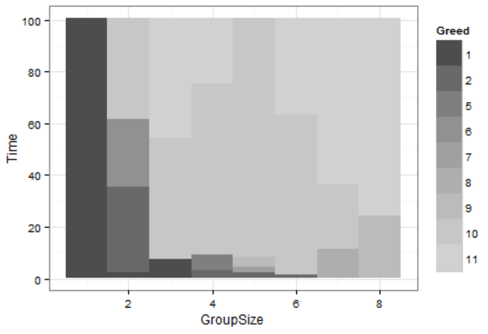

Figure 3: Equilibrium state of cooperative levels as a function of initial group size and number of generations before mixing. Black regions show parameter combinations which select for cooperative trait.

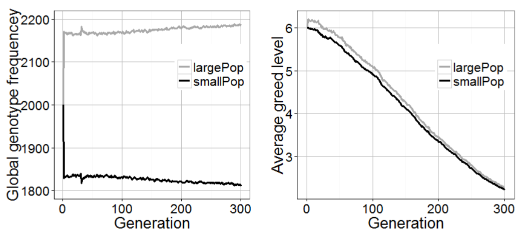

Figure 4: Beep Boop

The initial jump at the beginning in the first generation is likely an artefact of the code, having explored some potential causes the discarding of group sampling accounts for 0.5% difference which cannot cause the discrepancy observed. The jump is formed in the initial growth stages and it is not clear if this in an artefact of the code or real because as a rule of thumb step functions don't exist in nature but a possible explanation is that the populations quickly adapt to the increased resource given to the large groups as the ratio between the group sizes at this point of 1.17:1 is close to the ratio between resource per individual of 1.2:1.

In contrast to the binary model the genotype frequency for group size is roughly stable with no large changes. While the large population does increase slowly over time, it is possible that given enough generations a similar pattern to the binary model could be observed.

Strikingly the average greed level from 1 (cooperative) to 11 (selfish) systematically decays over time for both large and small population. This effect is uniform for both the large population and small population despite the cooperative bias for small group size.

It appears that the within group competition which benefits greedy individuals is being suppressed by the variety of different cooperation levels. An example of this is that in the 11 factor model, a single individual with the same level of greed as the binary model exploits the rest of the group to a lower level because the expected level of the group is approximately half way between the cooperative and selfish levels of the binary model.

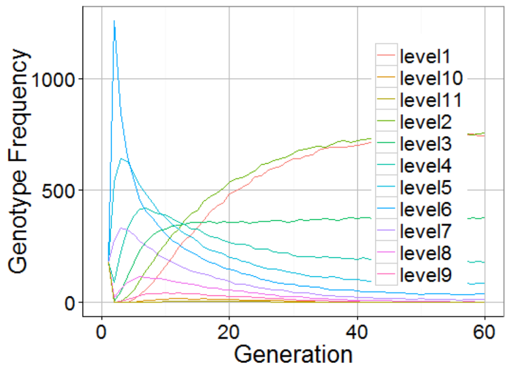

Figure 5: Showing the different genotype frequencies of each genotype in the large group size over time

Observing figures 5 we observe similar effects to the binary model where "greedy" individuals initially peak but then crash leaving the cooperative factors of 1,2 and 3 to ultimately dominate. The final distribution becomes quite ordered by the initial 20 generations appears chaotic. Very high greed levels such as 8 + have very small peaks presumably due to the low efficiency of those levels combined with the average selfishness of population never reaching high numbers. The individuals which take advantage of the current population trade off efficiency and resource share. It appears that level 5 has the best combination of these variables however as the average selfishness decreases it shifts this ideal level down lower. While the same factors in play in the binary model eventually allow the cooperative genotypes to do well.

4 Conclusion

While we haven't explored the limits of different conditions or run the model simulation for long enough to provide certainty in our results. We have shown support for the hypothesis of cooperation from social niche construction in spite of conditions such as group size selecting against it.

We hypothesised an arms race increasing greediness as the ideal level of greed is only slightly above the average of the population due to decreased efficiency so individuals would have to be slightly more and more greedy but observed the opposite where the average greed of the population slowly deteriorates shifting the ideal level of greed to capitalise on this down. Reinforced by the maintained high population of level 2 selfishness in the population despite trends of increased cooperation over time.

References

[1] Simon T. Powers, Alexandra S. Penn, Richard A. Watson, *Individual Selection for Cooperative Group Formation*, Proceedings of the 9th European Conference on Artificial Life, (2007), 585-594.

[2] Pfeiffer, T., Schuster, S., Bonhoeffer, S., *Cooperation and competition in the evolution of ATP-producing pathways*, Science **292**(5516), (2001), 504-507.

[3] Pfeiffer, T., Bonhoeffer, S., *An evolutionary scenario for the transition to undiferentiated multicellularity*, PNAS **100**(3), (2004), 2751-2760.

[4] Kreft, J.U., *Biofilms promote altruism.*, Microbiology **150**, (2004), 2751-2760.

[5] Kreft, J.U., Bonhoeffer, S., *The evolution of groups of cooperating bacteria and the growth rate versus yield trade-off*, Microbiology **151**, (2005), 636-641,

[6] Nowak, M.A., May, R.M, *The spatial dilemmas of evolution*, International Journal of Bifurcation and Chaos **3**(1), (1993), 35-78,

[7] Wilson, D.S, *A theory of group selection*, PNAS **72**(1), (1975), 143-146,

[8] Wilson, D.S, *The Natural Selection of Populations and Communities*, Benjamin/Cummings, (1980),

[9] Wilson, D.S, *The Evolution and Psychology of Unselfish Behaviour*, Harvard University Press, Cambridge, MA **3**(1), (1998), 35-78,

[10] Simon T. Powers, *Social nice construction: evolutionary explanations for cooperative group formation*, University of Southampton, School of Electronics and Computer Science, Doctral Thesis, 186pp, (2010),

5 Appendix

The following code is written in the R language.

5.1 Appendix 1. Header file for reproduction

This header file contains he major algorithmic functions to reproduce Powers' model.

```
# packages
if (!require(data.table)){
  install.packages("data.table")
  library(data.table)
}
if (!require(ggplot2)){
  install.packages("ggplot2")
  library(ggplot2)
}


#------- STEP 2. GROUP FORMATION -------
groupFormation <- function () {

  # Total individuals with each group allele.
  dispersedPop[, Sum := rowSums(.SD), .SDcols = 2:3]
  smallGroups <- floor(dispersedPop[group == "small", Sum]
                      / groupSizeSmall)
  largeGroups <- floor(dispersedPop[group == "large", Sum]
                      / groupSizeLarge)
  if (length(largeGroups) == 0) largeGroups = 0L
  if (length(smallGroups) == 0) smallGroups = 0L


  # Add each small individual to a pool and create a sample queue
  poolSmall <- c(rep(1L, dispersedPop[group == "small", greedy]),
                 rep(0L, dispersedPop[group == "small", coop]))
  poolSmall <- sample(poolSmall, dispersedPop[group == "small", Sum],
                      replace = F)

  # Populate groups from the sample

  for (i in 0:(smallGroups - 1)) {
    numGreedy <- (sum(poolSmall[(i*groupSizeSmall):
                              ((i + 1)*groupSizeSmall-1)]))
    population[i + 1, 2:4 := .("small", numGreedy, groupSizeSmall
                              - numGreedy)]
  }

  if (length(largeGroups) == 0) largeGroups = 0L
  if (largeGroups > 0) {
    # Repeat for large groups
    poolLarge <- c(rep(1L, dispersedPop[group == "large", greedy]),
                   rep(0L, dispersedPop[group == "large", coop]))
    poolLarge <- sample(poolLarge, dispersedPop[group == "large", Sum],
                        replace = F)

    # Populate groups from the sample
    for (i in 0:(largeGroups-1)){
    numGreedy <- (sum(poolLarge[(i*groupSizeLarge):
                              ((i+1)*groupSizeLarge-1)]))
    population[i+smallGroups+1, 2:4 :=
                 .("large", numGreedy, groupSizeLarge - numGreedy)]
    }
  }
}


#-------- STEP 3 Repdroduction ---------


reproduction <- function() {

invisible(sapply(1:maxGroups, function(x) {
  greedy <- population[x, greedy]
  cooperative <- population[x, coop]

  # R is based on group size
  ifelse(population[x, (group == "large") & !is.na(group)],
         R <- Rlarge,
         R <- Rsmall
         )

  # Calculate share of the resource.
  s <- greedy * Gs * Cs
  c <- cooperative * Gc * Cc
  rs <- R * s/(s+c)
  rc <- R * c/(s+c)

  # Calculate new population size
  newGreedy <- greedy + rs/Cs - K*greedy
  newCooperative <- cooperative + rc/Cc - K*cooperative

  # Set new population sizes
  population[x, "greedy" := newGreedy]
  population[x, "coop" := newCooperative]
  }))
}

buildMigrantPool <- function() {
  #---- STEP 4 MIGRANT POOL FORMATION ----

  dispersedPop <- (population[, .(greedy = sum(greedy, na.rm=T),
                                  coop = sum(coop, na.rm=T)),
                              by = .(group)])

  # STEP 5 MAINTAINING GLOBAL CARRYING CAPACITY
  dispersedPop[, Sum := rowSums(.SD), .SDcols = 2:3]
  scalingFactor <- sum(dispersedPop$Sum) / N
  dispersedPop[, greedy := round(greedy / scalingFactor)]
  dispersedPop[, coop := round(coop / scalingFactor)]
  dispersedPop[, Sum := rowSums(.SD), .SDcols = 2:3]

  return (dispersedPop)
  }
```

5.2 Appendix 2. Script for reproducing figure 1.

This script reproduces figure 1 of Powers' paper. The code for the graph is also included. The script is adapted from the reproduction of figure 2 script (appendix 3.)

```
source('reproductionHeader.r')

# Initialise variables
```

```
N <- 1000
groupSizeSmall <- 4L
groupSizeLarge <- 40L
Rsmall = 4
Rlarge = 40
t <- 3
Gc <- 0.018
Gs <- 0.02
Cc <- 0.1
Cs <- 0.2
Tgen <- 100
R = 1
K = 0.1
Time <- matrix(nrow = 3, ncol = 8)

for (i in 1:5) {
  for (groupSizeSmall in 1:8) {
    dispersedPop <- data.table(group = c("small", "large"),
                               greedy = c(N/2, 0),
                               coop = c(N/2, 0))

    # Initialise population table with maximum number of group sizes
    maxGroups <- floor(N/groupSizeSmall)
    population <- data.table(id = 1:maxGroups, group = NA_character_,
                             greedy = NA_real_, coop = NA_real_)

    genotypes <- data.table(greedySmall = rep(NA_real_, Tgen) ,
                            greedyLarge = NA_real_,
                            coopSmall = NA_real_, coopLarge = NA_real_)
    groupFormation()
    for (j in 1:Tgen)# iterate for T generations
    {
      reproduction()
      dominant <- sum(population$greedy) - sum(population$coop)
      if (dominant < 0) print(paste("coop␣dominant␣at␣time", j))
      else {
        print(paste("greedy␣dominant␣at␣time", j))
        break
      }
      Time[i,groupSizeSmall] <- j
    }
  }
}

## Draw Graph
library("extrafont")
font_import()
png(filename = "./bin/EquilibriumState.png", width = 480, height = 360)

Time[is.na(Time)] <- 0
```

```
x <- apply(Time, 2, mean)

science_theme = theme(panel.grid.major =
                        element_line(size = 0.5, color = "grey"),
                      panel.grid.minor.y = element_blank(),
                      axis.line = element_line(size = 0.7,
                                               color = "black"),
                      legend.position = c(0.85,0.7),
                      text = element_text(size = 14))
plot <- ggplot(data.frame(x),aes(seq_along(x),x)) +
  geom_bar(stat = "identity", width = 1) +
  xlab("Initial␣Group␣Size") +
  ylab("Time␣spent␣in␣groups␣before␣mixing") +
  scale_x_discrete( limits = c(1,2,3,4,5,6,7,8),
                    labels = c(1,2,3,4,5,6,7,"8+"),
                    breaks = c(1,2,3,4,5,6,7,8)) +
  scale_y_continuous( labels = c(0,20,40,60,80,100),
                      breaks = c(0,20,40,60,80,100),
                      limits = c(0,100),
                      minor_breaks = NULL) +
  science_theme +
  theme_bw(base_size = 20) +
  annotate("text", x = 6, y = 100,
           label = "\u25a0␣Cooperative␣trait␣selected␣for",
           color = "grey20")
plot
dev.off()

save(genotypes, file = "./bin/equilibriumStats.Rda")
```

5.3 Appendix 3. Script file for reproducing figure 2.

This file runs the functions described in the header in order to produce figure 1 and figure 2 in Powers' paper. The code for plotting the graphs is also present.

```
# Initialise variables
N <- 4000
groupSizeSmall <- 4L
groupSizeLarge <- 40L
R = 1
Rsmall = 4
Rlarge = 50
# Rlarge = 1.05^groupSizeLarge * groupSizeLarge
# Rsmall = 1.05^groupSizeSmall * groupSizeSmall
t <- 4
Gc <- 0.018
Gs <- 0.02
Cc <- 0.1
Cs <- 0.2
```

```r
Tgen <- 120
K = 0.1
source('reproductionHeader.r')


#--------------------------------------
#-- STEP 1. INITIALISE N INDIVIDUALS ---
#--------------------------------------

# Because we are using R we describe individuals in a vectorised mannor.
# Each element of the vector is an allele, with a 3rd variable stating
# which group the group and 4th the size of the genotype.

dispersedPop <- data.table(group = c("small", "large"),
                           greedy = c(N/4, N/4),
                           coop = c(N/4, N/4))

# Initialise population table with maximum number of group sizes
maxGroups <- floor(N/groupSizeSmall)
population <- data.table(id = 1:maxGroups, group = NA_character_,
                         greedy = NA_real_, coop = NA_real_)

genotypes <- data.table(greedySmall = rep(NA_real_, Tgen + 1) ,
                        greedyLarge = NA_real_,coopSmall = NA_real_,
                        coopLarge = NA_real_)
genotypes[1, greedySmall := N/4]
genotypes[1, greedyLarge := N/4]
genotypes[1, coopSmall := N/4]
genotypes[1, coopLarge := N/4]



for (j in 1:Tgen) { # iterate for T generations
  groupFormation()
  for (i in 1:t) reproduction() # wait time t
  dispersedPop <- buildMigrantPool()
  print(dispersedPop)
  population <- data.table(id = 1:maxGroups, group = NA_character_,
                           greedy = NA_real_, coop = NA_real_)

  # Save population values after each generation
  x <- dispersedPop[group == "small", greedy]
  if (length(x) == 0) x = 0
  genotypes[j + 1, greedySmall := x]

  x <- dispersedPop[group == "large", greedy]
  if (length(x) == 0) x = 0
  genotypes[j + 1, greedyLarge := x]

  x <- dispersedPop[group == "small", coop]
  if (length(x) == 0) x = 0
  genotypes[j + 1, coopSmall := x]

  x <- dispersedPop[group == "large", coop]
  if (length(x) == 0) x = 0
  genotypes[j + 1, coopLarge := x]

  # Print current state to console
  print(paste("Generation:", j))
  print(genotypes[j])
}


#---------------------------------------
#--------------- Draw plots ---------------
#---------------------------------------
png(filename = paste("./bin/", "Figure2", ".png", sep = ""),
    width = 900, height = 360)
library(gridExtra)
genotypes$index <- 0:Tgen


science_theme = theme(panel.grid.major
                      = element_line(size = 0.5, color = "grey"),
                      panel.grid.minor.y = element_blank(),
                      axis.line = element_line(size = 0.7,
                                               color = "black"),
                      legend.position = c(0.7,0.5),
                      text = element_text(size = 26))

graph1 <- ggplot(genotypes, aes(x = index, y = value)) +
  geom_line(aes(y = greedySmall/N, col = "Selfish␣+␣Small",
                linetype = "Selfish␣+␣Small",
                size = "Selfish␣+␣Small")) +
  geom_line(aes(y = greedyLarge/N, col = "Selfish␣+␣Large",
                linetype = "Selfish␣+␣Large",
                size = "Selfish␣+␣Large")) +
  geom_line(aes(y = coopSmall/N, col = "Cooperative␣+␣Small",
                linetype = "Cooperative␣+␣Small",
                size = "Cooperative␣+␣Small")) +
  geom_line(aes(y = coopLarge/N,col = "Cooperative␣+␣Large",
                linetype = "Cooperative␣+␣Large",
                size = "Cooperative␣+␣Large")) +
  xlab("Generation") +
  ylab("Global␣genotype␣frequencey") +
  theme_bw() +
  science_theme +
  scale_x_continuous( breaks = c(0,20,40,60,80,100,120),
                      limits = c(0,120)) +
  scale_y_continuous(breaks = c(0,0.2,0.4,0.6,0.8,1.0),
                     limits = c(0,1)) +
  scale_colour_manual("",
                      breaks = c("Selfish␣+␣Small", "Selfish␣+␣Large",
                                 "Cooperative␣+␣Small",
                                 "Cooperative␣+␣Large"),
                      values = c("Dark␣grey", "Dark␣grey",
                                 "Black", "Black")) +
  scale_linetype_manual("",
                        breaks = c("Selfish␣+␣Small",
                                   "Selfish␣+␣Large",
                                   "Cooperative␣+␣Small",
                                   "Cooperative␣+␣Large"),
                        values = c(3, 1, 3, 1)) +
  scale_size_manual("",
                    breaks = c("Selfish␣+␣Small", "Selfish␣+␣Large",
                               "Cooperative␣+␣Small",
                               "Cooperative␣+␣Large"),
                    values = rep(1.4,4))
## plot2
Large <- apply(genotypes, 1, function(x) {
  sum((x["greedyLarge"] + x["coopLarge"]) / sum(x))
})
Greedy <- apply(genotypes, 1, function(x) {
  sum((x["greedyLarge"] + x["greedySmall"]) / sum(x))
})
Secondplot <- data.frame(Large = Large, Greedy = Greedy)

graph2 <- ggplot(data = Secondplot,
                 aes(y = seq(0,1,0.2), x = seq(1,120,20))) +
  geom_line(aes(y = Large, x = 1:length(Large),
                linetype = "Large␣group␣size",
                size = "Large␣group␣size",
                col = "Large␣group␣size")) +
  geom_line(aes(y = Greedy, x = 1:length(Greedy),
                linetype = "Selfish␣resource␣usage",
                size = "Selfish␣resource␣usage",
                col = "Selfish␣resource␣usage")) +
  theme_bw() +
  science_theme +
  scale_linetype_manual("",
                        breaks = c("Large␣group␣size",
                                   "Selfish␣resource␣usage"),
                        values = c(3,1)) +
  scale_size_manual("",
                    breaks = c("Large␣group␣size",
                               "Selfish␣resource␣usage"),
                    values = c(1.2, 1.2)) +
  scale_color_manual("",
                     breaks = c("Large␣group␣size",
                                "Selfish␣resource␣usage"),
                   values = c("Black", "Dark␣grey")) +
  xlab("Generation") +
  ylab("Global␣frequency") +
  scale_x_continuous(breaks = seq(0,120,20),
                   limits = c(0,120)) +
  scale_y_continuous(breaks = seq(0,1,0.2),
                      limits = c(0,0.8))
grid.arrange(graph2, graph1, nrow=1, ncol=2)
dev.off()

save(genotypes, file = paste0("./bin/" ,name, ".Rda"))
```

5.4 Appendix 4. Header file for extension

This header file contains the major algorithmic functions for the extension. It inherits many of its features from Appendix 1, adapting the same core algorithm for variable cooperation levels.

```
# packages
if (!require(data.table)){
  install.packages("data.table")
  library(data.table)
}
if (!require(ggplot2)){
  install.packages("ggplot2")
  library(ggplot2)
}

test1 <- 0
test2 <- 0
test3 <- 0
#------- STEP 2 GROUP FORMATION -------
groupFormation <- function() {

  # Total individuals with each group allele.
  dispersedPop[, Sum := rowSums(floor(.SD)), .SDcols = 2:(levels+1)]
  smallGroups <- floor(dispersedPop[group == "small", Sum]
                    / groupSizeSmall)
  largeGroups <- floor(dispersedPop[group == "large", Sum]
                    / groupSizeLarge)
  if (length(largeGroups) == 0) largeGroups = 0L
  if (length(smallGroups) == 0) smallGroups = 0L


  # Add each small individual to a pool and create a sample queue
  poolSmall <- vector()
  invisible(sapply(1:levels, function(x) {
    block <- rep(as.integer(x),
                 dispersedPop[1, get(paste0("level", x))])
    poolSmall <<- c(poolSmall, block)
  }))
  poolSmall <- sample(poolSmall, length(poolSmall), replace = F)

  # Populate groups from the sample
  for (i in 1:(smallGroups - 1)) {
    startIndex <- groupSizeSmall * i
    endIndex <- groupSizeSmall * (i+1) - 1
    popRow <- as.numeric(sapply(1:levels, function(x) {
      sum(poolSmall[startIndex:endIndex] == x)
    }))
    population[i, group := .("small")]
    invisible(sapply(1:levels, function(x) {
      population[i, x + 2 := popRow[x]]
    }))
  }

  if (length(largeGroups) == 0) largeGroups = 0L
  if (largeGroups > 0) {
    # Repeat for large groups
    poolLarge <- vector()
    invisible(sapply(1:levels, function(x) {
      block <- rep(as.integer(x),
                   dispersedPop[1, get(paste0("level", x))])
      poolLarge <<- c(poolLarge, block)
    }))
    poolLarge <- sample(poolLarge, length(poolLarge), replace = F)

    # Populate groups from the sample
    for (i in 1:(largeGroups - 1)) {
      startIndex <- groupSizeLarge * i
      endIndex <- groupSizeLarge * (i+1) - 1
      popRow <- as.numeric(sapply(1:levels, function(x) {
        sum(poolLarge[startIndex:endIndex] == x)
      }))
      population[i + smallGroups + 1, group := .("large")]
      invisible(sapply(1:levels, function(x) {
        population[i + smallGroups + 1, x + 2 := popRow[x]]
      }))
    }
  }
}

#------- STEP 3 Repdroduction ---------


reproduction <- function() {

  invisible(sapply(1:maxGroups, function(x) {

    n <- population[x, .SD, .SDcols = 3:(levels+2)]
    # R is based on group size
    ifelse(population[x, (group == "large") & !is.na(group)],
           R <- Rlarge,
           R <- Rsmall
    )

    # Calculate share of the resource.
    ngc <- n * G * C
    NGC <- sum(ngc)
    r <- R * ngc / NGC

    # Calculate new population sizes
    nNext <- n + r/C - K*n

    # Set new population sizes
    invisible(sapply(1:levels, function (y) {
      population[x, y + 2 :=  as.numeric(nNext[[y]])]
    }))
  }))
}

buildMigrantPool <- function() {
#------- STEP 4 MIGRANT POOL FORMATION ---------
  unformatted <- population[, colSums(.SD, na.rm = T),
                            .SD = 1:levels+2, by = .(group)]
  vars <- names(dispersedPop)[2:(levels+1)]
  set(dispersedPop, 1L, vars,
      as.list(unformatted[ group == "small"]$V1))
  if(nrow(unformatted[group=="large"])>0 ) {
    set(dispersedPop, 2L, vars,
        as.list(unformatted[ group == "large"]$V1))
  }

  # STEP 5 MAINTAINING GLOBAL CARRYING CAPACITY
  dispersedPop[, Sum := rowSums(.SD), .SDcols = 2:(levels+1)]
  scalingFactor <- sum(dispersedPop$Sum) / N

  set(dispersedPop, 1L, vars,
      (dispersedPop[1L, .SD, .SDcols = (vars)] / scalingFactor))
  set(dispersedPop, 2L, vars,
      (dispersedPop[ 2L, .SD, .SDcols = (vars)] / scalingFactor))

  dispersedPop[, Sum := rowSums(.SD), .SDcols = 2:(levels+1)]

  return(dispersedPop)
}

####### Mutation ############
mutate <- function(rate) {
  numberOfGroups <- floor(dispersedPop[group == "small", Sum]
                          / groupSizeSmall) +
    floor(dispersedPop[group == "large", Sum] / groupSizeLarge)

  invisible(sapply(1:(numberOfGroups-1), function(i) {
    row <- population[i, .SD, .SDcols = 3:(levels+2)]
    if (sum(is.na(row) > 0)) return(row)
    # Calculate mutations going up
    end <- TRUE
    up <- sapply(row, function(x){
      if(x < 0) return(0)
      rand <- sample(ceiling(2/rate), floor(x), replace = TRUE)
      ifelse(end == TRUE, {
             mutateUp <- sum(rand == 1 | rand == 2)
             end <<- FALSE },
             mutateUp <- sum(rand == 1))
      return(mutateUp)
    })
    up[levels] <- 0

    # Calculate mutations going up
    end <- TRUE
    down <- rev(sapply(rev(row), function(x){
      if (x < 0) { return(0)}
      rand <- sample(ceiling(2/rate), floor(x), replace = TRUE)
      ifelse(end == TRUE,
             mutateDown <- sum(rand == 1 | rand == 2),
             mutateDown <- sum(rand == 1))
      return(mutateDown)
    }))
    down[1] <- 0

    movement <- c(0,up[1:(levels-1)]) +
      c(down[2:(levels)],0) - up - down
    newRow <- movement + row

    vars <- names(population)[3:(levels+2)]
    set(population, as.integer(i), vars, newRow)
  }))
}
```

5.5 Appendix 5. Script for extension

This script contains code to produce figures [COMPLETE THIS DESCRIPTION]

```
# Initialise variables
N <- 4400
groupSizeSmall <- 4L
groupSizeLarge <- 40L
```

```
R = 1
Rsmall = 4
Rlarge = 50
t <- 4
levels = 11
G <- seq(0.018, 0.02, length.out = levels)
C <- seq(0.1, 0.2, length.out = levels)
Tgen <- 400
K = 0.1
source('extensionHeader.r')


#---------------------------------------
#-- STEP 1. INITIALISE N INDIVIDUALS ---
#---------------------------------------

# Because we are using R we describe individuals in a vectorised mannor.
# Each element of the vector is an allele, with a 3rd variable stating
# which group the group and 4th the size of the genotype.

initial <- N/(levels*2)
dispersedPop <- data.table(group = c("small", "large"))
invisible(sapply(1:levels, function (x) {
  dispersedPop[, paste0("level",x) :=  rep(initial, 2)]
}))

# Initialise population table with maximum number of group sizes
maxGroups <- floor(N/groupSizeSmall)
population <- data.table(id = 1:maxGroups, group = NA_character_)
invisible(sapply(1:levels, function(x) {
  population[, paste0("level", x) := NA_real_]
}))

genotypes <- data.table(level1small = c(initial, rep(NA_real_, Tgen)))
invisible(sapply(1:levels, function(x) {
  genotypes[, paste0("level", x, "small") :=
              c(initial, rep(NA_real_, Tgen))]
}))
invisible(sapply(1:levels, function(x) {
  genotypes[, paste0("level", x, "large") :=
              c(initial, rep(NA_real_, Tgen))]
}))

for (j in 1:Tgen) { # iterate for T generations
  groupFormation()
  for (i in 1:t) reproduction() # wait time t before mixing
  buildMigrantPool()
  print(dispersedPop)

  # Empty the groups
  invisible(sapply(1:levels, function(x) {
    population[, paste0("level", x) := NA_real_]
  }))

  # Save population values after each generation
  x <- dispersedPop[1, setdiff(colnames(dispersedPop),
                               c("group","Sum")), with=FALSE]
  y <- dispersedPop[2, setdiff(colnames(dispersedPop),
                               c("group","Sum")), with=FALSE]
  row <- cbind(x,y)
  row[is.na(row)]<-0
  set(genotypes, as.integer(j + 1),
      names(genotypes), as.list(round(row)))


  # Print current state to console
  print(paste("Generation:", j))
  print(genotypes[j])
  save(genotypes, file = "genotypes.Rda")
}
```

5.6 Appendix 6. Script for extension including mutation

```
# Initialise variables
N <- 4000
groupSizeSmall <- 4L
groupSizeLarge <- 40L
R = 1
Rsmall = 4
Rlarge = 50
t <- 4
levels = 11
G <- seq(0.018, 0.02, length.out = levels)
C <- seq(0.1, 0.2, length.out = levels)
Tgen <- 400
K = 0.1
source('extensionHeader.r')


#----------------------------------------
#-- STEP 1. INITIALISE N INDIVIDUALS ---
#----------------------------------------

# Because we are using R we describe individuals in a vectorised mannor.
# Each element of the vector is an allele, with a 3rd variable stating
# which group the group and 4th the size of the genotype.

initial <- N/(levels*2)
```

```r
dispersedPop <- data.table(group = c("small", "large"))
invisible(sapply(1:levels, function (x) {
  dispersedPop[, paste0("level",x) :=  rep(0, 2)]
}))
dispersedPop$level6 <- rep(N/2,2)

# Initialise population table with maximum number of group sizes
maxGroups <- floor(N/groupSizeSmall)
population <- data.table(id = 1:maxGroups, group = NA_character_)
invisible(sapply(1:levels, function(x) {
  population[, paste0("level", x) := NA_real_]
}))

genotypes <- data.table(level1small = c(initial, rep(NA_real_, Tgen)))
invisible(sapply(1:levels, function(x) {
  genotypes[, paste0("level", x, "small") :=
              c(initial, rep(NA_real_, Tgen))]
}))
invisible(sapply(1:levels, function(x) {
  genotypes[, paste0("level", x, "large") :=
              c(initial, rep(NA_real_, Tgen))]
}))

for (j in 1:Tgen) { # iterate for T generations
  groupFormation()
  for (i in 1:t) {
    reproduction() # wait time t before mixing
    mutate(0.1)
  }
  buildMigrantPool()
  print(dispersedPop)

  # Empty the groups
  invisible(sapply(1:levels, function(x) {
    population[, paste0("level", x) := NA_real_]
  }))

  # Save population values after each generation
  x <- dispersedPop[1, setdiff(colnames(dispersedPop)
                               ,c("group","Sum")), with=FALSE]
  y <- dispersedPop[2, setdiff(colnames(dispersedPop),
                               c("group","Sum")), with=FALSE]
  row <- cbind(x,y)
  row[is.na(row)]<-0
  set(genotypes, as.integer(j + 1),
      names(genotypes), as.list(round(row)))


  # Print current state to console
  print(paste("Generation:", j))
  print(genotypes[j])
  save(genotypes, file = "genotypes2.Rda")
}
```

5.7 Appendix 7. Script for plotting Figure 3

```
library(data.table)
library(ggplot2)
library(gridExtra)
load("~/Southampton/EvolutionOfComplexity/CW2/genotypes.Rda")
genotypes[, smallPop := rowSums(.SD, na.rm = T), .SDcols = 1:11]
genotypes[, largePop := rowSums(.SD, na.rm = T), .SDcols = 12:22]

colNamesSmall <- grep("level.*small",names(genotypes), value = T)
colNamesLarge <- grep("level.*large",names(genotypes), value = T)


newPlot <- genotypes[, smallCoop :=
            level1small +
            2*level2small +
            3*level3small +
            4*level4small +
            5*level5small +
            6*level6small +
            7*level7small +
            8*level8small +
            9*level9small +
            10*level10small +
            11*level11small
          ]



science_theme = theme(panel.grid.major =
                        element_line(size = 0.5, color = "grey"),
                      panel.grid.minor.y = element_blank(),
                      axis.line =
                        element_line(size = 0.7, color = "black"),
                      legend.position = c(0.8,0.5),
                      text = element_text(size = 26))

graph2 <- ggplot(data = genotypes, aes(x = 1:401, y = largePop)) +
  xlim(0,301) +
  ylim(1800,2200) +
  theme_bw() +
  science_theme +
  geom_line(aes(y = largePop, col = "largePop",
                linetype = "largePop", size = "largePop")) +
  geom_line(aes(y = smallPop, col = "smallPop",
                linetype = "smallPop", size = "smallPop")) +
  xlab("Generation") +
  ylab("Global␣genotype␣frequencey") +
  scale_colour_manual("",
                      breaks = c("largePop", "smallPop"),
                      values = c("Dark␣grey", "Black")) +
  scale_linetype_manual("",
                        breaks = c("largePop", "smallPop"),
                        values = c(1, 1)) +
  scale_size_manual("",
                    breaks = c("largePop", "smallPop"),
                    values = rep(1.4,2))

graph1 <- ggplot(data = genotypes, aes(x = 1:401, y = largePop)) +
  xlim(0,301) +
  theme_bw() +
  science_theme +
  geom_line(aes(y = largeCoop, col = "largePop",
                linetype = "largePop", size = "largePop")) +
  geom_line(aes(y = smallCoop, col = "smallPop",
                linetype = "smallPop", size = "smallPop")) +
  xlab("Generation") +
  ylab("Average␣greed␣level") +
  scale_colour_manual("",
                      breaks = c("largePop", "smallPop"),
                      values = c("Dark␣grey", "Black")) +
  scale_linetype_manual("",
                        breaks = c("largePop", "smallPop"),
                        values = c(1, 1)) +
  scale_size_manual("",
                    breaks = c("largePop", "smallPop"),
                    values = rep(1.4,2))

grid.arrange(graph2, graph1, nrow=1, ncol=2)

cols <- grey.colors(11)
ggplot(newPlot, aes(x = 1:401)) +
  geom_line(aes(y = level1large, col = "level1")) +
  geom_line(aes(y = level2large, col = "level2")) +
  geom_line(aes(y = level3large, col = "level3")) +
  geom_line(aes(y = level4large, col = "level4")) +
  geom_line(aes(y = level5large, col = "level5")) +
  geom_line(aes(y = level6large, col = "level6")) +
  geom_line(aes(y = level7large, col = "level7")) +
  geom_line(aes(y = level8large, col = "level8")) +
  geom_line(aes(y = level9large, col = "level9")) +
  geom_line(aes(y = level10large, col = "level10")) +
  geom_line(aes(y = level11large, col = "level11")) +
  xlim(0,60) +
  theme_bw() +
  science_theme +
  xlab("Generation") +
  ylab("Genotype␣Frequency") +
theme(legend.title=element_blank())
```

5.8 Appendix 8. Script for plotting Figure 4

```
load('resultsList.rda')
library(RColorBrewer)
levels <- 11


####

test <- resultsList[[8]]
test <- test %>% select(-Sum, -group)
test <- as.matrix(test)
test2 <- apply(test, 1, which.max)
dim(test2) <- c(100,8,5)
dim(test) <- c(100,8,5,11)

# Calcualte average over 5 repeats
average <- apply(test, MARGIN = c(1,2,4), mean)

# Find currently dominant greed level
dom <- apply(average, MARGIN = c(1,2), which.max)

# Draw graph
heatmap(dom, Rowv = NA, Colv = NA)

dat_long <- reshape2::melt(dom)
dat_long$value <- factor(dat_long$value)
names(dat_long) <- c("Time", "GroupSize", "Greed")

gg <- ggplot(dat_long) +
  geom_tile(aes(x=GroupSize, y=Time, fill=Greed)) +
  scale_fill_manual(values = grey.colors(11)) +
  theme_bw() +
  scale_y_continuous(breaks = seq(0,100,20))
gg
```