

RICE UNIVERSITY

**Evolution of Altruism and Eusociality: Toward a Cost/Benefit
Analysis of Fitness and Genetic Relatedness**

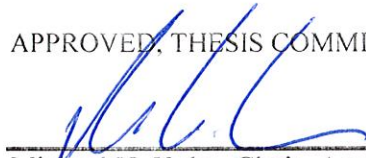
by

Xiaoyun Liao

A THESIS SUBMITTED
IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE

Doctor of Philosophy

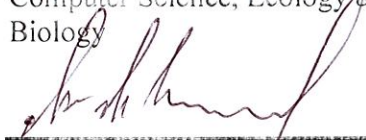
APPROVED, THESIS COMMITTEE



Michael H. Kohn, Chair, Associate Professor of
Ecology & Evolutionary Biology



Luay K. Nakhleh, Associate Professor of
Computer Science, Ecology & Evolutionary
Biology



Marek Kimmel, Professor of Statistics



Nicholas H. Putnam, Assistant Professor of
Ecology & Evolutionary Biology

HOUSTON, TEXAS

December, 2013

RICE UNIVERSITY

**Evolution of Altruism and Eusociality: Toward a Cost/Benefit
Analysis of Fitness and Genetic Relatedness**

By

Xiaoyun Liao

A THESIS SUBMITTED
IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE

Doctor of Philosophy

APPROVED, THESIS COMMITTEE

Michael H. Kohn, Chair, Associate Professor of
Ecology & Evolutionary Biology

Luay K. Nakhleh, Associate Professor of
Computer Science, Ecology & Evolutionary
Biology

Marek Kimmel, Professor of Statistics

Nicholas H. Putnam, Assistant Professor of
Ecology & Evolutionary Biology

HOUSTON, TEXAS
December, 2013

ABSTRACT

Evolution of Altruism and Eusociality: Toward a Cost/Benefit Analysis of Fitness and Genetic Relatedness

By

Xiaoyun Liao

Altruism is a behavior that benefits others at a cost to one's own ability of survival and/or reproduction; that is, individual fitness. Thus, altruism poses great challenges to Darwin's theory of evolution by natural selection on individual fitness. Altruistic behaviors are commonly performed in eusocial animals, such as nearly all hymenoptera (including bees, wasps, and ants), termites, ambrosia beetles, and so on. Inclusive fitness theory predicts that altruistic behavior can evolve when sufficient fitness benefits are given to relatives even though individual fitness is reduced. A different modeling approach has led to a challenge to this theory. The modelers claim that relatedness is not causal, that eusocial behavior is very hard to evolve requiring more workers before the queen increased fitness, and that there is no conflict involved. Here I showed that, even within the terms of this modeling framework, inclusive fitness thinking leads to insights that completely change these conclusions. I showed that relatedness and inclusive fitness indeed are causal and that eusociality does evolve more readily. With regard to the latter this means eusociality can be favored under a lower benefits threshold. I concluded that multiple modeling approaches are useful and that efforts to synthesize them are

better than asserting that one is universally better than the other. Moreover, either greenbeard effects or genetic kin recognition requires genetic polymorphisms as cues on which recognition is based. Previous models showed that selection eliminates rare cue alleles and a common allele gets fixed, i.e. altruism cannot persist. So it is unclear how genetic recognition for altruism persists under a Darwinian selection framework. Here, I designed a novel model with three types of genetic components (production, perception, and action). I analyzed my recognition model theoretically toward a cost/benefit analysis of fitness and genetic relatedness. I predicted the stability of recognition for altruism based on my model. Furthermore I tested my recognition model through various computational and biological simulations. My simulation results consistently showed altruism could maintain multiple recognition cues and be evolutionarily stable; given the assumptions of my model. I concluded that cost/benefit of fitness and genetic relatedness both play critical roles in the evolution of altruism and eusociality, and therefore can maintain the stability of recognition for altruism.

Acknowledgments

I would like to express my sincere gratitude to my advisors, Dr. Michael H. Kohn, Dr. David C. Queller and Dr. Joan E. Strassmann, for the continuous support of my Ph.D. study and research, for their patience, motivation, enthusiasm, and immense knowledge. I really appreciate all of them.

I would like to appreciate my committee members, Dr. Marek Kimmel, Dr. Luay K. Nakhleh, and Dr. Nicholas H. Putnam, who provided encouraging and constructive feedback. I am grateful for their thoughtful and detailed comments.

I would also like to thank the Department of Ecology and Evolutionary Biology at Rice University for helping me through some special times.

I would also like to thank my family and all the friends who have helped and supported me along this long but fulfilling road.

Contents

Acknowledgments.....	iv
Contents	v
List of Figures	viii
List of Tables	x
List of Equations	xi
Nomenclature	12
Introduction to Altruism and Eusociality	14
1.1. Hamilton's Rule and Kin Selection	14
1.2. Crozier's Paradox.....	18
1.3. Nowak et al.'s Model of the Evolution of Eusociality	19
1.4. Conclusions.....	19
1.5. References.....	21
The Evolution of Eusociality and Kin Selection Redux	25
2.1. Abstract	25
2.2. Introduction.....	27
2.3. Results	30
2.4. Discussion	35
2.5. Methods	41
2.6. References.....	47
2.7. Figure legends	52
2.8. Supplemental Materials	56
2.8.1. Altruism is Easy to Evolve under High Relatedness.....	56
2.8.2. Altruism cannot Evolve without Relatedness.....	63
2.8.3. The Higher the Relatedness, the Easier Altruism to Evolve	67
2.8.4. Eusociality is Easy to Evolve under Flexible Strategies of Workers.....	71
2.8.5. Conclusions	77
Evolutionary Dynamics of Genetic Kin Recognition: a General Model	78
3.1. Abstract	78

3.2. Introduction.....	79
3.3. Negative Feedbacks from <i>Judge</i> to <i>Cue</i>	80
3.4. Genetic Relatedness at Each of Three Loci	82
3.5. Mathematical Analysis of the Model	83
3.6. Prediction of Stability of Evolution of Altruism and Eusociality.....	88
3.7. Conclusions.....	90
3.8. References.....	91
3.9. Table legends.....	98
3.10. Figure legends	101
Exploring Fitness Cost/Benefit to Solve the Crozier's Paradox	110
4.1. Abstract	110
4.2. Introduction.....	110
4.3. Recognition Bases on Multiple Cues	115
4.4. Simulation of Pure Greenbeard	117
4.5. Negative Feedbacks can Maintain Multiple Cue with Intermediate Frequencies.....	117
4.6. Stable Recognition Model under High Fitness Cost/Benefit and High Cost	118
4.7. The Recognition is Unstable When Cue Alleles Start Rare	119
4.8. Conclusions.....	120
4.9. References.....	122
4.10. Figure legends	129
Applications of New Perspective of Hamilton's rule: $r > C/B$.....	140
5.1. Abstract	140
5.2. Introduction.....	141
5.3. The genetic relatedness and evolution of eusociality.....	142
5.4. The general model for genetic kin recognition	142
5.5. The fitness cost/benefit and the Crozier's Paradox	143
5.6. Conclusions.....	143
5.7. References.....	144
Appendix A	148
A.1 Scripts for Altruism to Evolve with Genetic Relatedness.....	148
A.2 Scripts for Evolutionary Dynamics of Genetic Kin Recognition: a General Model	171

A.3 Scripts for Exploring Fitness Cost/Benefit to Solve the Crozier's Paradox	186
---	-----

List of Figures

Figure 2.1 Eusocial evolution when the offspring decision rules is “stay if colony size is below w ”	52
Figure 2.2 Eusocial evolution under threshold fitness functions with no benefits to working below colony size 3 (two workers) (2) (solid circles) versus stepped functions where the one worker contributes half the benefit of two workers (open and solid circles).	53
Figure 2.3 Eusocial evolution in the Nowak et al. (2) haploid model, but adding variable relatedness r (equations 3).....	54
Figure 2.4 Eusocial evolution under offspring versus maternal control.	55
Figure 3.1 Genetic Relatedness at Each of Three loci.....	101
Figure 3.2 Genetic Relatedness for all alleles at three loci.....	102
Figure 3.3 Prediction of Stability of Evolution of Altruism and Eusociality ..	103
Figure 3.4 Genetic Relatedness with Two Cue Alleles.....	104
Figure 3.5 Genetic Relatedness with Ten Cue Alleles.....	105
Figure 3.6 Plots of Shannon’s diversity index for two cue and judge alleles ($c/b=0.82$).....	106
Figure 3.7 Plots of Shannon’s diversity index for two cue and judge alleles ($c/b=0.9$).	107
Figure 3.8 Plots of Shannon’s diversity index for three cue and judge alleles ($c/b = 0.65$).....	108
Figure 3.9 Plots of Shannon’s diversity index for three cue and judge alleles ($c/b = 0.9$).	109
Figure 4.1 Evolutionary dynamics of Genotype Frequencies in Unstable Kin Recognition	129
Figure 4.2 Evolutionary dynamics of Allele Frequencies in Unstable Kin Recognition	130

Figure 4.3 Evolutionary dynamics of Genotype Frequencies in Stable Kin Recognition	131
Figure 4.4 Evolutionary dynamics of Allele Frequencies in Stable Kin Recognition	132
Figure 4.5 The stability of recognition.	133
Figure 4.6 The stability of recognition in space of frequencies.	134
Figure 4.7 Limited regions of stability: low p.....	137
Figure 4.8 The space of frequency spaces under various recombination rates and cost-to-benefit ratios.....	138
Figure 4.9 The space of frequency spaces under various initialized allele with highly linked loci.	139

List of Tables

Table 3.1 Alleles Indexes at Three Loci.....	98
Table 3.2 Payoff matrix of altruism	99
Table 3.3 Genetic Relatedness at Each of Three loci	100

List of Equations

Equation 1.1 Hamilton's Rule	16
Equation 2.8.1 Evolutionary Dynamics under High Relatedness.....	58
Equation 2.8.2 Average Birth Rate Among Colonies Without Genetic Relatedness	64
Equation 2.8.3 Evolutionary Dynamics without Genetic Relatedness	64
Equation 2.8.4 Average Birth Rate Among Colonies With Various Genetic Relatedness	67
Equation 2.8.5 Evolutionary Dynamics under Various Genetic Relatedness ..	68
Equation 2.8.6 Average Birth Rate Among Colonies With Flexible Strategies of Workers.....	73
Equation 2.8.7 Evolutionary Dynamics With Flexible Strategies of Workers ..	75
Equation 3.1 Frequency of Altruism in the Population	84
Equation 3.2 The Frequency of Altruism by Descent	85
Equation 3.3 The Mean Fitness of the Population after Altruism	85
Equation 3.4 The Mean Fitness of Individuals	85
Equation 3.5 The Frequencies of Genotypes	87
Equation 3.6 The Frequencies of Cue Alleles.....	87
Equation 3.7 The Frequencies of Judge Alleles.....	88
Equation 3.8 The Frequencies of Genotypes	88
Equation 3.9 Shannon's diversity index for cue and judge alleles.....	90

Nomenclature

B : fitness benefit

C : fitness cost

r : genetic relatedness

q : a fraction that the offspring of the eusocial queen stay with the nest

i : the number of individuals at the colony including the queen

m : a critical colony size

b_0 : general birth rates

b_i : colony-size specific birth rates

d_0 : general death rates

d_i : colony-size specific death rates

α : worker mortality rates

η : population density dependence

e_i : the number of colonies of size i headed by a eusocial queen

s_i : the number of colonies of size i headed by a solitary queen

X : the total population size including workers

η : a parameter that scales the size of the system

p_r : a fraction that partner is clonemate

C_1, C_2, K_1, K_2 : cue alleles

J_1, J_2 : judge alleles

A_1, A_2 : action alleles

p_1, p_2 : frequencies of cue allele C_1, C_2

q_1, q_2 : frequencies of judge allele J_1, J_2

T: units of simulation time

h: recombination rate

Chapter 1

Introduction to Altruism and Eusociality

1.1. Hamilton's Rule and Kin Selection

Altruism is a behavior that lowers the Darwinian fitness (number of offspring produced in one's lifetime) of the actor and increases that of the recipient. The most extreme forms of altruism are observed in eusocial insects such as ants, bees, wasps and termites (Fletcher and Michener 1987; Frank 1998; Liebert et al. 2004). In such eusocial insect workers have little or no reproductive ability with undeveloped or reduced ovaries. They generally stay within nests to help others raise offspring and only one or a small number of individuals reproduce. Darwin recognized in 1859 that paradox of sterile workers is "... one special difficulty, which at first appeared to me insuperable, and actually fatal to my theory" (Darwin 1859). The evolution of the altruistic behaviors remained without any satisfactory explanation for a hundred years.

Darwin also indicated that "... [the problem] disappears when it is remembered that selection may be applied to the family, as well as the individual and may thus gain the desired end" (Darwin 1859). So what is the unit of selection, a group or a single individual? In 1962 Wynne-Edwards argued that natural selection acts only at the level of groups (Wynne-Edwards 1962). This idea was once popular in 1960's and was rejected in 1970's, but now there are reasonable versions (Koeslag 1997; Koeslag and Terblanche 2003).

In 1964 W. D. Hamilton produced an elegant formal theory that provided a potential solution to this problem of reconciling the apparent sacrifice made by individuals at their own fitness expense and Darwin's evolution by natural selection (Hamilton 1964). Specifically, Hamilton argued that altruistic acts to relatives could be favored by natural selection, because relatives share the same gene as helpers. Thus, Hamilton expanded the definition of individual fitness to include inclusive fitness, which is the sum of a direct benefit through producing offspring and an indirect benefit through aiding genetic relatives. Hamilton made these two components additive by devaluing each offspring or relative by the genetic relatedness to them.

From this Hamilton predicted that altruism will be favored by natural selection when the inequality

$$rB - C > 0$$

Equation 1.1 Hamilton's Rule

is satisfied, where B is the benefit of the act of altruism to the recipient, C the cost of the act to the actor and r the genetic relatedness between the actor and the recipient (Hamilton 1964). Inclusive fitness is applicable not only to helping but also to any behavior (West et al. 2007b). This inequality has now come to be known as 'Hamilton's Rule'. Hamilton's theory is also frequently referred to as the inclusive fitness theory or kin selection theory.

In the past half century Hamilton's work attracted high attention in the social evolution study field, in that the theory prompted extensive empirical and theoretical explorations; notably tests of the prediction that altruism should be correlated with relatedness (West et al. 2007).

Relatedness measures genetic similarity between any two individuals of a given species in a population. Historically, the definition of relatedness has received several refinements. Hamilton initially defined relatedness as the coefficient of relationship (Hamilton 1964; Wright 1922). In most situations (e.g. no inbreeding), it is the probability that a random allele the actor has at a focal locus is shared by the recipient via a common ancestor. These genes are called identical by descent. There are two reasons for why two alleles can be identical: identity by descent and non-identity by descent or identity by state. What really matters is identity above

random population frequencies. Coefficient of relationship is calculated from a pedigree. Thus in a diploid, the relatedness between full siblings is $1/2$, cousins is $1/8$ as J.B.S. Haldane says, “I would jump into a river to save two brothers or eight cousins.”

Later in 1970 Hamilton revised the definition of relatedness as the regression coefficient of recipients' genotypes on actors' genotypes, which is the slope of the regression line in the least squares method (Hamilton 1970). Therefore, the regression relatedness measures similarity of two individuals' genotypic values which are the frequency of the focal allele in individuals. One of the merits of regression relatedness is that one does not necessarily need a pedigree to calculate the regression relatedness and can estimate it from genetic markers, which is a wider scope than the 1964-version (Goodnight and Queller 1999). And regression relatedness is highly compatible with the Price equation (Grafen 1985; Price 1970; Queller 1992).

However, testing Hamilton's rule without measuring the cost and benefit of eusociality is an inadequate test. Focusing only on relatedness and neglecting the cost and benefit terms usually takes the form of assuming implicitly that $B = C$. In my thesis I designed a novel model of genetic kin recognition and explained how eusociality could evolve and persist.

1.2. Crozier's Paradox

One problem with inclusive fitness theory is the recognition of relatedness by individuals acting in an altruistic fashion. Ideas to resolve this issue involve recognition genes and traits; i.e. cues. Altruism based on genetic cues requires multiple recognition cues, yet altruism is predicted to erode this genetic variation. Common cues get favored and eventually dominate the population while rare cues are disfavored and go extinct. The genetic cues of the production component are shown to be greenbeard genes recognizing copies of themselves in others, regardless of relatedness at other loci. The greenbeard nature of these alleles is responsible for what is known as Crozier's paradox, the observation that selection favors common cue alleles and thereby removes the variation that is required for discrimination (Crozier RH 1986, Crozier RH 1987, Crozier RH & Pamilo P 1996). Altruistic greenbeard alleles are outlaw genes because, by causing altruism towards others who are not relatives, they act against the interest of other genes in the genome. This can lead to intragenomic conflict, with other genes being selected to eliminate the extra altruism, if they can do so without also eliminating themselves as targets of altruism. Therefore, the origin and maintenance of multiple genetic recognition cues remains incompletely understood. Here in my thesis I explored the range of fitness costs and benefits with the model to explain how eusociality could evolve and persist.

1.3. Nowak et al.'s Model of the Evolution of Eusociality

Inclusive fitness theory and Hamilton's Rule have found broad support. However, recently Nowak et al. challenged this theory with a different modeling approach, claiming that relatedness is not causal, that eusocial behavior is very hard to evolve, and that there is no conflict between queens and workers (Nowak et al. 2010,). This publication prompted a hot debate in the field. There were strong critiques against Nowak et al.'s conclusions, but none of them looked at the models of Nowak et al. and provided solid evidences to support their verbal arguments. (Abbot P, et al. 2011, Ferriere R & Michod RE, 2011, Nowak MA, Tarnita CE, & Wilson EO, 2011, Strassmann JE, Page RE, Robinson GE, & Seeley TD, 2011). So I examined Nowak et al.'s model in greater depth, and showed that relatedness is causal, that eusociality is not so difficult to evolve, and explored conflicts between queens and workers.

1.4. Conclusions

In chapter 2 of my thesis I examined Nowak et al.'s model in greater depth. I showed that all of its novel conclusions are overgeneralized from narrow and often inappropriate assumptions and that the insights of kin selection theory stand. I showed that, even within the terms of their modeling framework, that inclusive fitness theory and Hamilton's Rule is supported as I showed that relatedness, once incorporated in the model as a variable, alters the probability of eusocial behaviors. I

also showed that relatedness is causal, that eusociality is not so difficult to evolve; that is, eusociality can be favored under a lower benefits threshold. Lastly, I explore the effects conflicts between queens and workers have on the theory.

In chapter 3 I designed a novel model of genetic kin recognition. I analyzed my recognition model theoretically toward a cost/benefit analysis of fitness and genetic relatedness. I predicted the stability of recognition for altruism according to relatednesses at three kinds of loci and Hamilton's rule.

In chapter 4 I explored the range of fitness costs and benefits with the model to explain how eusociality could evolve and persist. I tested my recognition model through various computational and biological simulations. My simulation results consistently showed altruism can maintain multiple recognition cues and be evolutionarily stable. Thus, I found a model and parameter space that could resolve Croxier's paradox that stands in great conflict with inclusive fitness theory and the need for recognition mechanisms among kin.

In chapter 5 I discussed the applications of new perspective of Hamilton's rule, not only focusing on genetic relatedness, but also emphasizing the importance of fitness cost and benefit.

1.5. References

Abbot P, et al. (2011) Inclusive fitness theory and eusociality. *Nature* 471(7339):E1-E4.

Boomsma JJ, et al. (2011) Only full-sibling families evolved eusociality. *Nature* 471(7339):E4-E5.

Crozier, R. H. (1986). Genetic clonal recognition abilities in marine invertebrates must be maintained by selection for something else. *Evolution* 40, 1100-1101.

Crozier, R. H. (1987). Genetic aspects of kin recognition: concepts, models, and synthesis (New York: Wiley).

Crozier RH & Pamilo P (1996) *Evolution of Social Insect Colonies: Sex Allocation and Kin Selection* (Oxford University Press, Oxford).

Darwin CD (1859) *On the Origin of Species* (Harvard Univ. Press, Cambridge).

Ferriere R & Michod RE (2011) Inclusive fitness in evolution. *Nature* 471(7339):E6-E8.

Fletcher, D. J. C., and Michener, C. D. (1987). *Kin Recognition in Animals*: John Wiley & Sons).

Foster KF (2011) Social behavior in microorganisms. *Social behaviour: genes, ecology and evolution*, eds Szekely T, Moore A, & Komdeur J (Cambridge University Press, Cambridge), pp 331-356.

Frank, S. A. (1998). *Foundations of Social Evolution* (Princeton, New Jersey: Princeton University Press).

Frank SA (1995) Mutual policing and repression of competition in the evolution of cooperative groups. *Nature* 377:520-522.

Goodnight, K. F., and Queller, D. C. (1999). Computer software for performing likelihood tests of pedigree relationship using genetic markers. *Molecular Ecology* 8, 1231-1234.

Grafen, A. (1985). A geometric view of relatedness, In *Oxford Surveys in Evolutionary Biology*, R. Darwins, and M. Ridley, eds. (Oxford University Press).

Grafen, A. (1990). Do animals really recognize kin? *Anim Behav* 39, 42–54.

Hamilton WD (1964) The genetical evolution of social behaviour. I-II. *J. Theor. Biol.* 7:1-52.

Hamilton, W. D. (1970). Selfish and Spiteful Behaviour in an Evolutionary Model. *Nature* 228, 1218-&.

Hamilton, W. D. (1987). *Discriminating nepotism: expectable, common, overlooked* (New York: Wiley).

Herre EA & Wcislo WT (2011) In defence of inclusive fitness theory. *Nature* 471(7339):E8-E9.

Nowak MA, Tarnita CE, & Wilson EO (2010) The evolution of eusociality. *Nature* 466:1057-1062.

Nowak MA, Tarnita CE, & Wilson EO (2011) Nowak et. al reply. *Nature* 471(7339):E9-E10.

Price, G. R. (1970). Selection and Covariance. *Nature* 227, 520-&.

Queller DC (1992) Quantitative genetics, inclusive fitness, and group selection. *Am. Nat.* 139(3):540-558.

Strassmann JE, Page RE, Robinson GE, & Seeley TD (2011) Kin selection and eusociality. *Nature* 471:E5-E6.

Taylor PD & Frank SA (1996) How to make a kin selection model. *Journal of Theoretical Biology* 180(1):27-37.

Trivers RL & Hare H (1976) Haplodiploidy and the evolution of the social insects. *Science* 191:249-263.

West, S. A., Griffin, A. S., and Gardner, A. (2007a). Evolutionary explanations for cooperation. *Curr Biol* 17, R661-672.

West, S. A., Griffin, A. S., and Gardner, A. (2007b). Social semantics: altruism, cooperation, mutualism, strong reciprocity and group selection. *Journal of Evolutionary Biology* 20, 415-432.

Wright, S. (1922). Coefficients of inbreeding and relationship. *American Naturalist* 56, 330-338.

Wynne-Edwards, V. C. (1962). *Animal Dispersion in Relation to Social Behavior* (London: Oliver & Boyd).

Chapter 2

The Evolution of Eusociality and Kin Selection Redux

The chapter is based on a manuscript co-authored with David C. Queller and Stephen Rong (from Biology Department, Washington University in St. Louis, One Brookings Drive, St. Louis, MO 63130 USA). David Queller and Xiaoyun Liao designed study; Xiaoyun Liao and Stephen Rong conducted simulations; David Queller wrote text.

2.1. Abstract

The evolution of sterile worker castes in the eusocial insects was a major problem until Hamilton (1) developed a method called inclusive fitness. He used it to show that sterile castes could evolve via kin selection, in which a gene for altruistic sterility is favored when the altruism sufficiently benefits relatives carrying the gene. A recent paper (2) argued that the general method of inclusive fitness was wrong and advocated an alternative method using differential equations.

It also used these alternative methods to model the evolution of eusociality and purported to overturn much of what was previously understood about the topic. Specifically, it claimed that eusociality was difficult to evolve, that genetic relatedness was unimportant, and that workers were not independent agents but instead are mere extensions of the queen. Here we report a more thorough examination of such differential equation models for the evolution of eusociality and show that all three of these conclusions derive from over-generalizing from narrow assumptions or parameter values. For example, all of their models implicitly assumed high relatedness but modifying the model to allow lower relatedness shows that relatedness is essential and causal. Contrary to their claims, their modeling strategy, properly applied, generally confirms the insights of inclusive fitness studies of kin selection.

Eusocial insects have sterile castes that are thought to have evolved by kin selection. A gene for giving up reproduction by workers can spread when the workers help to rear relatives who share that gene. This solution to a crucial evolutionary problem was challenged in a recent paper. The paper generated much controversy, but no one has contested its new model of the evolution of eusociality, which purported to overturn much of what was previously thought to be true from kin selection theory. Here we examine this model in greater depth. We show that all of its novel conclusions are overgeneralized from narrow and often inappropriate assumptions and that the insights of kin selection theory stand.

2.2. Introduction

The eusocial insects have occupied an important place in biology because of their extraordinary levels of cooperation (1, 3-5). In ants, termites, some bees, some wasps, and a few other taxa, some individuals, called workers, give up their own reproduction in order to help others reproduce. Darwin was vexed over the question of how such reproductive altruism evolves or indeed how any traits of sterile workers evolve, but he believed the answer lay in some form of selection at the family level or at the group level (6). Hamilton provided the first rigorous treatment of this idea, with a key insight being the importance of genetic relatedness. A conditional gene causing a worker to give up reproduction could be favored if it provided sufficient help to a relative who would share that gene at above-random levels (1). He showed that this process, which became known as kin selection, could be analyzed by summing up an actor's fitness effects, each multiplied by the actor's relatedness to the individual receiving the fitness effect. When this sum, called inclusive fitness, is positive, the trait should be favored by selection. For giving up one's reproduction (fitness cost c) to benefit another (fitness gain b) related by r , the inclusive fitness condition is $-c + rb > 0$.

Kin selection and inclusive fitness became the dominant modes of thinking about the evolution of eusocial insects (5, 7, 8) and the success in this area has led to them being applied to many other problems in social evolution (9-13). Recently, this paradigm was criticized by Nowak et al. (2) who argued that inclusive fitness

was an inaccurate and unnecessary method and that kin selection was not a very useful way to think about social evolution. Both of these conclusions have in turn been extensively criticized as depending on multiple misconceptions (14-20). We concur with many of these criticisms, but here we offer a different kind of critique of the Nowak et al. paper. Nowak et al. (2) also developed their own mathematical model of the evolution of eusociality, presenting it as a superior alternative to inclusive fitness modeling. However, as has been recently pointed out (21), this eusociality model has scarcely been addressed.

Although this approach to modeling could be useful, we show here that its implementation in Nowak et al. (2) led to serious errors of interpretation. We demonstrate this by using the exact modeling approach recommended by Nowak et al. (2) to see if it generally supports their conclusions. In particular, we examine their three conclusions that seem at greatest variance with the conventional kin selection view of the evolution of eusociality. In each case, we will show that the kin selection view is essentially restored (hence the word “redux” in our title).

First, Nowak et al. (2) claim that eusociality is harder to evolve than has been appreciated. They write that “a key observation of our model is that it is difficult to evolve eusociality, because we need very favorable parameters” and “despite the obvious and intuitive advantages of eusociality, it is very hard for a solitary species to achieve it” (2). If there is any novelty in this conclusion, it must be that eusociality is harder to evolve than has been thought previously, that is, it is harder

to evolve than predicted from inclusive fitness ($-c + rb > 0$). In our first models, we explore how this conclusion changes with reasonable alterations in the fitness functions and the worker decision rules.

Second, Nowak et al. (2), following earlier work by Wilson (22, 23), claimed that relatedness was not an essential element in the evolution of eusociality. They wrote that “relatedness is better explained as a consequence rather than as the cause of sociality”, that “grouping by family hastens the spread of eusocial alleles but it is not a causative agent” and that “relatedness does not drive the evolution of eusociality” (2). This claim has already been criticized by pointing out that the Nowak et al. model was based on groups of relatives, with no comparable model of unrelated individuals being presented (15). Nowak et al. appear to have partially accepted this point: “One, we do not argue that relatedness is unimportant. Relatedness is an aspect of population structure, which affects evolution” (24). But this response leaves unanswered exactly how it affects evolution. Wilson (25) continues to assert that relatedness only hastens the spread of alleles and that it is not causal. Here we will use their own style of modeling to investigate these points.

Finally, where inclusive fitness theory has emphasized that cooperation occurs in the face of potential and actual conflicts among colony members (5, 8, 26, 27), Nowak et al. (2) assert that the colony as a whole is all that matters. They argue “that the workers are not independent agents”, that “their properties are

determined by the alleles that are present in the queen (both in her own genome and in that of the sperm she has stored), that “the workers can be seen as ‘robots’ that are built by the queen” and that they “are part of the queen’s strategy for reproduction” (2). Nor, contrary to earlier work by Wilson (22, 23), do they brook any conflicts between levels of selection: “there is only one level of selection, the hymenopteran colony, which is treated as an extension of the queen, whose genes are the units of selection” (2). Yet curiously, their eusociality model is what would normally be considered a worker control model, because it assumes expression of decision genes in workers. To test whether there might be worker-queen conflict, we construct the necessary parallel models where genes expressed in mothers determine whether their offspring stay and help.

2.3. Results

We modify the Nowak et al. (2) haploid model, which is simpler than the haplodiploid one, but sufficient to demonstrate the important points. This model includes solitary and eusocial genotypes expressed in offspring, where solitaires always leave to reproduce, while eusocials stay and help their mother with probability q and leave to reproduce with probability $1-q$. Mothers and offspring are genetically identical. Differential equations describe the numbers of solitary individuals and eusocial colonies based on colony-size specific birth rates (b_i) and death rates (d_i), as well as worker mortality rates (α) and density dependence (η)

(see Methods, eqn. set [1]). If larger colony size (added workers) increases the queen's birthrate sufficiently, the eusocial type will be favored over solitary reproduction under some probabilities of staying q . Using these equations, we recovered results indistinguishable from those of Nowak et al. (2) (e.g. their Fig. 4). We then explored the effects of various assumptions by changing them one by one.

The first claim that we examine, that eusociality is hard to evolve (2), is difficult to compare to inclusive fitness because the models are not expressed in the same terms. But the claim appears to be based on particular and arguably odd choices for fitness functions and worker decision rules. The fitness function that they generally explored was a threshold function where workers add no fitness gains to the queen below a colony of size m , and add a fixed gain (increasing queen b or decreasing d) in colonies at or above size m , regardless of how many workers are added. This means that workers in colonies below that threshold contribute nothing until enough further workers join and workers above the threshold also add nothing extra unless other workers die, returning the colony to the threshold. If most workers are contributing nothing, then it is not surprising that eusociality would be hard to evolve. In the example most explored (e.g. their Fig. 4), the threshold colony size m was set at 3, such that two workers were needed to raise the queen's birthrate from $b_0=0.5$ to $b=4$ and to lower her death rate from $d_0=0.1$ to $d=0.01$ (they also let $\alpha=0.1$ and $\eta=0.01$). This eight-fold increase in the queen's birthrate allowed eusociality to evolve for some values of q but lower values of b did

not. Not surprisingly, requiring more workers before the queen increased fitness (higher m thresholds) made eusociality even more difficult to evolve.

However, as noted above, the assumption that workers must stay with probability q , regardless of the state of the colony, means they may be maladaptively staying in colonies that are too large to gain further benefits. It should be easy for workers to avoid this problem. For example, they might instead implement the rule to stay when the colony is below some threshold size w and leave when it is at or above that size. We implemented differential equations to model this change in assumption (see Methods, eqn. set [2]) and show that eusociality does evolve more readily (Fig. 1). For example, for the same parameter values as in Fig. 4 of Nowak et al., eusociality can now be favored under a lower benefits threshold ($b=3$), that is, when helped queens get a six-fold advantage.

In addition, the threshold fitness function assumed by Nowak et al. (2) prevents the earliest workers from contributing anything. But it is easy to envision advantages that would come from having only a single worker can contribute something to the colony (23, 28). To view this effect in isolation, we return to the Nowak et al (2) decision rule (stay with probability q) and to their parameter values given above, but allow a single worker to add half the contribution to the queen that two workers add (for both birth rate and death rate) ($m=3$, $b_0 = 0.5$, $d_0 = 0.1$, $d = 0.01$, $\alpha = 0.1$, $\eta = 0.01$). This simple change (see Methods) makes it much easier to evolve

eusociality, with $b=1.5$ or only a three-fold increase required (Fig. 2) versus eightfold with the threshold model.

The models of Nowak et al. (2) also assumed eusocial offspring stay with their mother so that there was always genetic relatedness among participants. In the haploid model, this meant that offspring were genetically identical to their mother and to the siblings they raised ($r=1$). To vary genetic relatedness in the haploid model, we allowed some offspring to move between mothers before implementing their genetic helping rules. Each offspring has a probability r of staying with her own mother before deciding whether to help her or leave to reproduce, and a probability $1-r$ of moving. r is equivalent to relatedness to the new mother (after movement) because it represents identity to that mother above chance levels; a fraction r is identical to the head of their colony and her offspring ($r=1$), while the remainder are randomly associated with colonies ($r=0$). After this temporary mixing, offspring execute the original Nowak et al. strategies: the solitary genotype always leaves to reproduce alone and offspring with the eusocial genotype stay and help their colony with probability q . Differential equations implementing this model are given in the Methods (eqn. sets [3] and [4]).

Fig. 3 shows when eusociality is favored under varying relatedness r , worker-assisted queen birthrate b , and probability of staying q (other parameters continue to match the standard Nowak *et al.* Fig. 4 parameter values: $m=3$, $b_0 = 0.5$, $d_0 = 0.1$, $d = 0.01$, $\alpha = 0.1$, $\eta = 0.01$). At a given value of b , eusociality is favored over a

wider range of q values at high relatedness than at lower relatedness until, below some critical value of relatedness, eusociality is never favored. When all offspring move to random colonies so that relatedness is zero, even $b=500$ (a 1000-fold increase in the queen's birthrate due to helpers) is insufficient to favor eusociality. As expected from inclusive fitness theory, relatedness is causal in the sense that some relatedness appears necessary for eusociality and increasing r allows eusociality to evolve under more conditions.

To address the issue of whether worker offspring are independent agents or simply robots carrying out the queen's interests, we need to compare models of control by different agents. This means comparing models where the decision is made by genes in worker bodies to those where it is made by genes in queen bodies. Though Nowak et al. (2) seem to argue for queen control, their models are really all for offspring control because they assume that genes expressed in worker bodies determine the decision to stay or leave. Our models, including the variable relatedness model above (eqn. set [4]) retain that assumption.

However, inclusive fitness theory predicts that queen control will generally be more favorable for evolving eusociality (8) unless relatedness is one, in which case no conflict is expected. To model queen control under varying relatedness in the haploid model, we allowed offspring to mix exactly as in the offspring control model above, but then allowed maternal genotypes to determine if their mixed offspring pool helps or not. If the mother has the solitary genotype, all of her mixed

pool disperses to become reproductives; if the mother has the eusocial genotype, she causes a fraction q of her offspring pool to stay and help her, independent of offspring genotype. Differential equations governing this system are given in the methods (eqn. set [5]). As predicted by inclusive fitness theory, eusociality evolves much more easily under queen control (Fig. 4). In fact, there is an opposite relationship with relatedness; the less related the queen's offspring are to her (lower r) the more the queen is selected to cause them to be workers. The only exception, as expected under inclusive fitness theory, is when there is no mixing between nests so $r = 1$. Only then are the two decision rules selected identically.

2.4. Discussion

The controversy over the Nowak et al paper has mostly been conducted at rather abstract levels; different researchers favor different modeling strategies and interpret the evidence differently (2, 14-20, 24). We take a different and more concrete approach by investigating their model for the evolution of eusociality more deeply. If their methods are superior and raise novel insights, we should welcome them. If instead their methods lead to errors, it tends to undermine the grander claims that the model is used to buttress.

We have therefore followed the recommendation of Nowak et al. (2) for modeling social evolution, and in particular eusociality, using deterministic evolutionary dynamics described by ordinary differential equations. However,

stimulated by inclusive fitness thinking, we have sought to understand apparent differences between their results compared to previous models. In every case, we find that their rejection of accepted results is premature, and that in fact the insights known from inclusive fitness theory also emerge using their method.

First, the claim that eusociality is difficult to evolve (2) hinges on assumptions that are heavily biased towards that conclusion, but little justification was given for why we should accept these narrow assumptions. In particular, assumptions are made that imply that many workers waste their efforts. First, their model allowed offspring to stay with probability q , independent of any information that might be available about the need for workers. One advantage of inclusive fitness thinking is that it induces researchers to think of workers as agents being selected to get their better outcomes (higher inclusive fitness) using whatever information is available to them. One such piece of information is the number of workers already present on the nest. In the threshold fitness model, there is no inclusive fitness gain to be had from staying above that threshold, unless some workers die, so we asked if there was some obvious better decision rule than stay with probability q . We therefore tested decision rules that have workers staying when the colony is below a threshold size (not necessarily the fitness threshold), and leaving when the colony is above that size. Not surprisingly, we find that this class of decision rules makes it easier to evolve eusociality, because fewer workers are making wasteful decisions to stay in large. Such a rule seems well within the capabilities of workers. They need not count adults. They simply need to be able to

assess some reasonable correlate of the count, something that even microbes do when using quorum sensing to change their behavior. For social insects, the mechanism might be the frequency with which they contact other adults or the hunger demands of offspring. Thus part of the assertion that eusociality is difficult to evolve stems from an assumption that workers are rather stupid.

Similarly, the threshold fitness model assumed by Nowak et al. greatly devalues worker behavior at the other, low, end of colony sizes. In most of the model implementations, it was assumed that it was necessary to have two workers to provide any benefit at all to the queen ($m=3$). That means that a first worker to join a colony provides nothing. However is easy to imagine situations where the first worker to join would provide real benefits (28). The simplest is that at this point one individual can guard the nest while the other forages (23). Empirical evidence suggests that first helpers do provide benefits (29-34). If we modify the Nowak et al threshold model to a step model where each worker below the threshold adds a fixed benefit, so that the effects of unjoined first workers are not wasted, eusociality evolves much more easily. We have not jointly modeled both of modifications – the stepped fitness function and the altered worker decision rule – but it seems certain that this would lead to even easier evolution of eusociality. But note that the stepped function by itself gets quite close to the inclusive fitness prediction which, given that relatedness is one, is that each worker needs to contribute to the queen more than what she herself loses. The stepped-model eusociality threshold of $b=1.5$ from having two workers matches that requirement,

because each worker adds 0.5 units, and gives up 0.5 of her own by not reproducing. The comparison is not perfect both because the inclusive fitness comparison does not figure in the workers' reduction of queen death rate in the model (which would decrease the b required for eusociality) and because some workers still join larger colonies and waste their effort (which would increase it). But it does show that, with reasonable assumptions, eusociality evolves in the model about as easily as inclusive fitness theory predicts.

The claims that relatedness only hastens the spread of eusocial alleles and that relatedness is not causal (2, 25) are also shown by our models to be false. The proposition could not be tested in the Nowak et al. (2) models because they did not examine any low-relatedness case (15). We have modeled variable relatedness and shown that high relatedness broadens the range of conditions (b, q) allowing eusociality to evolve. Relatedness affects not just speed of selection but whether it is favored at all. Furthermore, when relatedness is zero, eusociality does not evolve even with very high benefits (increasing queen birthrate 1000-fold). This shows that relatedness plays an essential and causal role. Of course these are not surprising findings because the importance of relatedness was previously well understood from many kinds of models using inclusive fitness (1, 8), population genetics (35-37), quantitative genetics (38-40) and game theory (41, 42) as well as supported by much empirical evidence (8, 10, 43, 44). What is surprising is that a contrary view would be advanced on the slim evidence of a model (2) that did not even investigate variable relatedness.

Finally, the assertion that workers are robots and simply part of the queen's reproductive success (2) cannot be made without testing and contrasting queen and worker decision rules. Nowak et al. (2) tested only offspring control models. It is a longstanding result of inclusive fitness theory that there can be parent-offspring conflict (26, 45). In particular, for the eusociality context, inclusive fitness predicts that offspring will be selected to help their mothers under a narrower range of conditions than the mothers would favor (eusociality evolves more readily if mothers control the helping of their offspring) (8 pp. 58-63). This is because of differences in relatedness. For example, for singly mated diploids, a worker gains siblings ($r=1/2$) at the cost of offspring ($r=1/2$) while the mother gains offspring ($r=1/2$) but loses only grand offspring ($r=1/4$). To examine this question, one must compare selection of offspring agency (genes expressed in the offspring determine whether she becomes a worker) versus maternal agency (genes expressed in mothers determine whether her offspring become workers). We constructed and compared haploid models for both offspring and maternal control. As predicted by inclusive fitness theory, they evolve differently and can be in conflict; mothers favor helping by their offspring under a broader range of conditions than the offspring themselves favor, except when mothers and offspring are genetically identical. In fact, when relatedness is low and eusociality is very difficult to evolve under worker control, it is very easy to evolve under queen control, because the queen is unrelated to most of the workers who pay the fitness cost. This also shows that if queens really were in control from the beginning, their best option would be to force

unrelated offspring to help, which is contradicted by phylogenetic studies showing that relatedness was always high at the various origins of eusociality (43). In contrast, the standard kin selection model of worker control predicts this observation.

The method advocated by Nowak et al. (2) offers the advantage of specifying parameters like birth and death rates explicitly and following their effects over time while allowing the some features, like colony size, to change. We expect that these methods can be used to generate interesting results. However, they are more complex and less intuitive than inclusive fitness thinking so considerable care is needed to fully understand them. The common thread in the three errors pointed out in this paper is overgeneralization from narrow assumptions and particular parameter values. Eusociality was said to be difficult to evolve based on specific and questionable assumptions about the fitness function and offspring decision rules. Relatedness was said to be unimportant even though the models did not vary relatedness. The assertion that workers are not independent agents was made in the absence of models that compared decision rules of different agents. The more complex the model, the easier it is to be misled by particular results that are not general. Besides being sure to explore multiple assumptions and parameter values, another way to avoid such problems is to understand prior work and relate it carefully to ostensibly new results. In this case, the new model missed not just minor details, but some of the most important generalizations known from the last five decades of theory and empirical study. Lack of agreement with prior inclusive

fitness results should have triggered more than a quick rejection of inclusive fitness and kin selection; it should have led to a questioning of why the results were, or seemed to be, different. When examined more closely, models of the type advocated by Nowak et al. (2) do not overturn, but instead reaffirm, principles of social evolution discovered through inclusive fitness.

2.5. Methods

Our models are all based on the haploid model of Nowak et al. (2). They modeled the evolution of eusociality with systems of differential equations tracking the number of solitary queens (x_0) and eusocial colonies of size i (x_i). We use a modified notation because our low relatedness models require us to keep also track colonies headed by solitary-genotype queens. We therefore let e_i be the number of colonies of size i headed by a eusocial queen (that is with $i-1$ workers) and s_i be the number of colonies of size i headed by a solitary queen. With this modified notation, equation set [58] of Nowak et al. (2) can be written as:

$$\dot{s}_1 = (b_1\phi - d_1)s_1$$

$$\dot{e}_1 = \sum_{i=1}^{\infty} b_i\phi(1-q)e_i - b_1\phi qe_1 - d_1e_1 + \alpha e_2 \quad [1]$$

$$\dot{e}_i = b_{i-1}\phi qe_{i-1} - b_i\phi qe_i - d_i e_i - \alpha(i-1)e_i + \alpha i e_{i+1} \quad \text{for } i \geq 2$$

where b_i and d_i are the birth and death rates of colonies of size i , q is the

probability that an offspring of a eusocial colony stays as a worker (offspring of solitary colonies never stay), α is the worker mortality rate, and ϕ is a density-dependent correction factor equal to $1/(1+\eta X)$, with X being the total population size including workers and η is a parameter that scales the size of the system.

For specific examples, Nowak et al. (2) usually used assumed birth rates and death rates were governed by a simple threshold function: below some threshold colony size m , $b_i = b_o$ and $d_i = d_o$; at or above colony size m , $b_i = b$ and $d_i = d$. Using Euler's method for numerical simulation in R, we reproduced the results of Fig. 4 in Nowak et al. (2)

We then tested an alternative worker decision rule. Instead of staying with probability q , eusocial offspring always stay when colony size $i < w$ and always leave when $i \geq w$. The equations are:

$$\dot{s}_1 = (b_1\phi - d_1)s_1$$

$$\dot{e}_1 = \sum_{i=w}^{\infty} b_i\phi e_i - b_1\phi e_1 - d_1e_1 + \alpha e_2$$

$$\dot{e}_i = b_{i-1}\phi e_{i-1} - b_i\phi e_i - d_i e_i - \alpha(i-1)e_i + \alpha i e_{i+1} \quad \text{for } 2 \leq i < w \quad [2]$$

$$\dot{e}_1 = b_{i-1}\phi e_{i-1} - d_i e_i - \alpha(i-1)e_i + \alpha i e_{i+1} \quad \text{for } i = w$$

$$\dot{e}_i = -d_i e_i - \alpha(i-1)e_i + \alpha i e_{i+1} \quad \text{for } i > w$$

We also altered the fitness function from a single threshold to a step function in which each added worker adds the same amount, up to the maximum b attained at colony size m . The maximum gain in both models is the same, but now each worker up to size m adds something. We can model this with equations [1]; if b_0 is the birthrate of a solitary queen and b is the birthrate of a eusocial queen in colony size m , then let the birthrate of queens in smaller colony sizes $i < m$ be $b_0 + (b - b_0)/(i - 1)$.

The Nowak et al. models all assumed high relatedness. We modify their haploid model to incorporate a parameterized mixing step, which allows us to vary the degree of relatedness between queens and workers. The mixing occurs before offspring decide to be workers or reproductive queens. We implemented lower relatedness in two (similar) ways. First, in our initial model, we allowed offspring from eusocial colonies to move to other mothers, eusocial or solitary, with probability r . 1) The offspring of the eusocial queens leave the colony to build their own eusocial ones with probability $1 - q$. 2) Of the rest probability q , the offspring of the eusocial queens stay with the nest with probability rq (r is relatedness to the mother they help because r of the time she is identical). 3) They migrate randomly among all nests (including eusocial and solitary ones) with probability $(1-r)q$. Thus each mother receives $\mu = \sum_{i=1}^{\infty} (1-r)b_i q e_i / (\sum_{i=1}^{\infty} e_i + \sum_{i=1}^{\infty} s_i)$ migrating eusocial offspring,

(the total number of migrating eusocial workers divided by the number of eusocial and solitary mothers). After movement, eusocial offspring, on both eusocial and solitary nests, execute their staying rule (stay with probability q). r is relatedness to the mother they help because r of the time she is identical, and $1-r$ of the time she is genetically random or unrelated. The differential equations describing this system are then:

$$\begin{aligned}\dot{e}_1 &= \sum_{i=1}^{\infty} b_i \phi (1-q) x_i - r b_1 \phi q x_1 - \mu \phi e_1 - d_1 e_1 + \alpha e_2 \\ \dot{e}_i &= r b_{i-1} \phi q e_{i-1} + \mu \phi e_{i-1} - r b_i \phi q e_i - \mu \phi e_i - d_i e_i - \alpha (i-1) e_i + \alpha i e_{i+1} \quad \text{for } i \geq 2 \\ \dot{s}_1 &= \sum_{i=1}^{\infty} b_i \phi s_i - \mu \phi s_1 - d_1 s_1 + \alpha s_2 \\ \dot{s}_i &= \mu \phi s_{i-1} - \mu \phi s_i - d_i s_i - \alpha (i-1) s_i + \alpha i s_{i+1} \quad \text{for } i \geq 2\end{aligned} \quad [3]$$

Here e_i and s_i still represent numbers after decision rules are executed, and do not reflect the numbers in the transient mixing stage. These equations are used in Fig. 3. To solve the ordinary differential equations above for the equilibrium status of the dynamic populations a first-order numerical procedure with the step size 0.1 were implemented in MATLAB. The procedure were started with equal initial number ($n=100$) of solitary females and eusocial queens, and were terminated when 1) the solitary populations were extinct (defined as less than 0.05) and thus eusociality evolved; 2) the eusocial populations were extinct (defined as

less than 0.05) and thus eusociality didn't evolve; 3) both the solitary and eusocial populations did not change any more and thus solitary and eusociality co-exist; or 4) after a maximum of 200,000 time steps.

In the model above movement of solitary offspring was not included but doing so would not change the outcome because either way they would ultimately to become reproductive. But for comparing maternal and offspring control (Fig. 4), we needed a second implementation of variable relatedness. We let *all* offspring, eusocial and solitary move to a new mother with probability r . In this second model, an offspring that initially mixes by moving to another colony is replaced by a eusocial or a solitary offspring with probabilities f_e and f_s , which are simply the proportions of such offspring produced in the population:

$$f_e = \frac{\sum_i b_i e_i}{\sum_i b_i \phi(s_i + e_i)}$$

$$f_s = \frac{\sum_i b_i s_i}{\sum_i b_i \phi(s_i + e_i)}$$

Thus, a minor difference with the previous model is that here each colony receives not a fixed number of migrants but the same number that it donates. But the main difference is that it can allow maternal control over offspring groups consisting of partly her own offspring and partly random offspring (including solitary). We therefore can compare offspring control and mother control. First, for offspring control, offspring genotype determines whether it stays with probability q

(eusocial offspring) or always leaves (solitary offspring). The equations describing changes in colony types are then:

$$\begin{aligned}
 \dot{s}_1 &= \sum_i (b_i \phi(1-r) f_s e_i) + \sum_i (b_i \phi(r + (1-r) f_s) s_i) - d_1 s_1 + \alpha s_2 & [4] \\
 \dot{s}_i &= b_{i-1} \phi(1-r) f_e q s_{i-1} - b_i \phi(1-r) f_e q s_i - d_i s_i + \alpha s_2 \\
 \dot{e}_1 &= \sum_i (b_i \phi(r + (1-r) f_e)(1-q) e_i) + \sum_i (b_i \phi(1-r) f_e(1-q) s_i) - b_1 \phi(r + (1-r) f_e) q e_i - d_1 e_1 + \alpha e_2 \\
 \dot{e}_i &= b_{i-1} \phi(r + (1-r) f_e) q e_{i-1} - b_i \phi(r + (1-r) f_e) q e_i - d_i e_i - \alpha(i-1) e_i + \alpha i e_{i+1}
 \end{aligned}$$

These equations are used in Fig. 4 for comparison with the maternal control model. For both models Euler's method was used in R to numerically determine the equilibrium population* of the system, using a time step of $h=0.1$ a maximum colony size of $n=50^{**}$, terminating when either E or S population/number of individuals was less than $\epsilon=0.1$ or after a maximum of 50000 time steps.

For maternal control therefore modified the variable-relatedness model in the previous section so that the mother's genotype determines whether the offspring in her colony (some of them resulting from mixing from other colonies) stay and help.

Mixing occurs as above, with a fraction $1-r$ of offspring initially moving to another mother. After the mixing step, the differentiation of offspring into workers and queens depends not on the genotype of the offspring, but by the genotype of the queen in the same colony. Thus, if the queen is eusocial, her (mixed) offspring will become new workers with probability q or new queens with probability $1-q$. If the queen is solitary, then all offspring will become reproductive.

$$\begin{aligned}
\dot{s}_1 &= b_1\phi(p + (1-p)f_s)s_1 + \sum_i (b_i\phi(1-p)f_s)(1-q)e_i - d_1s_1 \quad [5] \\
\dot{e}_1 &= \sum_i (b_i\phi(p + (1-p)f_e)(1-q)e_i) + b_1\phi(1-p)f_es_1 - b_1\phi qe_1 - d_1e_1 + \alpha e_2 \\
\dot{e}_i &= b_{i-1}\phi qe_{i-1} - b_i\phi qe_i - d_ie_i - \alpha(i-1)e_i + \alpha ie_{i+1}
\end{aligned}$$

Note that, unlike the worker model, there are no solitary colonies larger than one (after transient mixing stage) because a solitary queen always causes her offspring pool to disperse and become reproductive.

2.6. References

1. Hamilton WD (1964) The genetical evolution of social behaviour. I-II. *J. Theor. Biol.* 7:1-52.
2. Nowak MA, Tarnita CE, & Wilson EO (2010) The evolution of eusociality. *Nature* 466:1057-1062.
3. Hamilton WD (1972) Altruism and related phenomena, mainly in the social insects. *Annu. Rev. Ecol. Syst.* 3:193-232.
4. Wilson EO (1971) *The Insect Societies* (Harvard, Boston, Mass.).
5. Queller DC & Strassmann JE (1998) Kin selection and social insects. *Bioscience* 48(3):165-175.
6. Darwin CD (1859) *On the Origin of Species* (Harvard Univ. Press, Cambridge).
7. Bourke AFG & Franks NR (1995) *Social Evolution in Ants* (Princeton University Press, Princeton, USA).

8. Crozier RH & Pamilo P (1996) *Evolution of Social Insect Colonies: Sex Allocation and Kin Selection* (Oxford University Press, Oxford).
9. Wilson EO (1975) *Sociobiology: The new synthesis*. (Cambridge MA).
10. Bourke AFG (2011) *Principles of social evolution* (Oxford University Press, Oxford) p 288.
11. Haig D (2002) *Genomic imprinting and kinship* (Rutgers University Press, New Brunswick) p 240.
12. Alexander RD (1987) *The biology of moral systems* (Aldine de Gruyter, Hawthorne NY) p 301.
13. Foster KF (2011) Social behavior in microorganisms. *Social behaviour: genes, ecology and evolution*, eds Szekely T, Moore A, & Komdeur J (Cambridge University Press, Cambridge), pp 331-356.
14. Rousset F & Lion S (2011) Much ado about nothing: Nowak et al.'s charge against inclusive fitness theory. *J. Evol. Biol.* 24:1386-1392.
15. Herre EA & Wcislo WT (2011) In defence of inclusive fitness theory. *Nature* 471(7339):E8-E9.
16. Ferriere R & Michod RE (2011) Inclusive fitness in evolution. *Nature* 471(7339):E6-E8.
17. Bourke AFG (2011) The validity and value of inclusive fitness theory. *Proceedings of the Royal Society B: Biological Sciences* 278:3313-3320.
18. Abbot P, et al. (2011) Inclusive fitness theory and eusociality. *Nature* 471(7339):E1-E4.

19. Strassmann JE, Page RE, Robinson GE, & Seeley TD (2011) Kin selection and eusociality. *Nature* 471:E5-E6.
20. Boomsma JJ, *et al.* (2011) Only full-sibling families evolved eusociality. *Nature* 471(7339):E4-E5.
21. Owen RE (in press) RA Fisher and Social Insects: The Fisher-Darwin Model of the Evolution of Eusociality. *Biological Theory* published online before print.
22. Wilson E & Hölldobler B (2005) Eusociality: origin and consequences. *Proceedings of the National Academy of Sciences USA* (102).
23. Wilson EO (2008) One giant leap: how insects achieved altruism and colonial life. *Bioscience* 58:17-25.
24. Nowak MA, Tarnita CE, & Wilson EO (2011) Nowak et. al reply. *Nature* 471(7339):E9-E10.
25. Wilson E (2012) *The Social Conquest of the Earth* (W. W. Norton, London).
26. Trivers RL & Hare H (1976) Haplodiploidy and the evolution of the social insects. *Science* 191:249-263.
27. Ratnieks FLW, Foster KR, & Wenseleers T (2006) Conflict resolution in insect societies. *Annu. Rev. Entomol.* 51:581-608.
28. Queller DC (1994) Extended parental care and the origin of eusociality. *Proceedings of the Royal Society of London, Series B* 256:105-111.
29. Noonan KM (1981) Individual strategies of inclusive-fitness-maximizing in *Polistes fuscatus* foundresses. *Natural Selection and Social Behavior*, eds Alexander RD & Tinkle DW (Chiron, New York), pp 18-44.

30. Brand N & Chapuisat M (2014) Impact of helpers on colony productivity in a primitively eusocial bee. *Behav. Ecol. Sociobiol.* 68(2):291-298.
31. Shakarad M & Gadagkar R (1995) Colony founding in the primitively eusocial wasp, *Ropalidia marginata* (Hymenoptera: Vespidae). *Ecological Entomology* 20(3):273-282.
32. Shreeves G & Field J (2002) Group size and direct fitness in social queues. *The American Naturalist* 159(1):81-95.
33. Stevens MI, Hogendoorn K, & Schwarz MP (2007) Evolution of sociality by natural selection on variances in reproductive fitness: evidence from a social bee. *BMC Evol. Biol.* 7(1):153.
34. Yagi N & Hasegawa E (2012) A halictid bee with sympatric solitary and eusocial nests offers evidence for Hamilton's rule. *Nature communications* 3:939.
35. Michod RE (1982) The theory of kin selection. *Ann. Rev. Ecol. Syst.* 13:23-55.
36. Charnov EL (1977) An elementary treatment of the genetical theory of kin-selection. *J. Theor. Biol.* 66:541-550.
37. Harpending (1979) The population genetics of interactions. *Am. Nat.* 113:622-630.
38. Queller DC (1992) Quantitative genetics, inclusive fitness, and group selection. *Am. Nat.* 139(3):540-558.

39. McGlothlin JW, Moore AJ, Wolf JB, & Brodie ED (2010) interacting phenotypes and the evolutionary process. III. Social evolution. *Evolution* 64(9):2558-2574.
40. Bleakley BH, Wolf JB, & Moore AJ (2010) The quantitative genetics of social behavior. *Social Behaviour: Genes, Ecology and Evolution*, eds Székely T, Moore AJ, & Komdeur J (Cambridge University Press, Cambridge), pp 29-54.
41. Taylor PD & Frank SA (1996) How to make a kin selection model. *Journal of Theoretical Biology* 180(1):27-37.
42. Frank SA (1995) Mutual policing and repression of competition in the evolution of cooperative groups. *Nature* 377:520-522.
43. Hughes W, Oldroyd B, Beekman M, & Ratnieks F (2008) Ancestral monogamy shows kin selection is key to the evolution of eusociality. *Science* 320:1213-1216.
44. Kuzdzal-Fick JJ, Fox SA, Strassmann JE, & Queller DC (2011) High relatedness is necessary and sufficient to maintain multicellularity in *Dictyostelium*. *Science* 334:1548-1551.
45. Trivers RL (1974) Parent-offspring conflict. *Am. Zool.* 14:249-264.

2.7. Figure legends

Figure 2.1 Eusocial evolution when the offspring decision rules is “stay if colony size is below w ”.

The solitary birthrate $b_0=0.5$ gets raised to b in colonies of size m or larger. Other parameters are as in Nowak et al. Fig. 4 ($d_0 = 0.1$, $d = 0.01$, $\alpha = 0.1$, $\eta = 0.01$).

Eusociality evolves (filled circles) more easily (lower b) than under the decision rule (2) “stay with probability q ”, for example at $b=3$ instead of $b=4$ when $m=3$.

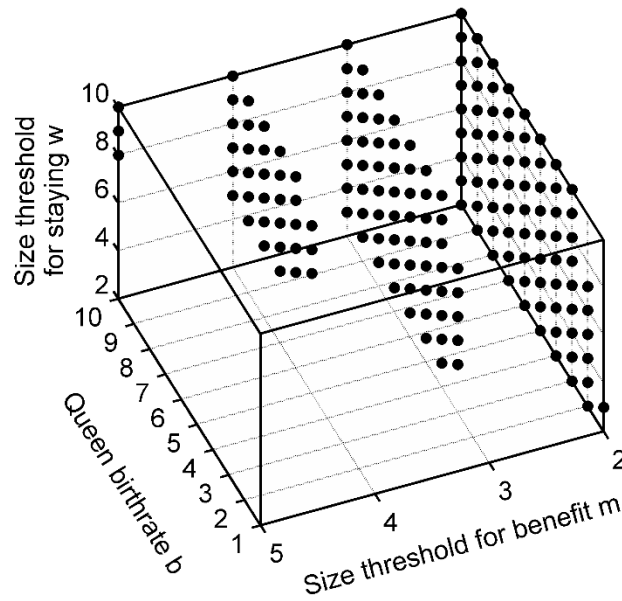


Figure 2.2 Eusocial evolution under threshold fitness functions with no benefits to working below colony size 3 (two workers) (2) (solid circles) versus stepped functions where the one worker contributes half the benefit of two workers (open and solid circles).

Eusociality evolves under a much broader range of maximum queen birthrates with the step function. Other parameters are as in Fig. 4 of Nowak et al. (2) ($m=3$, $b_0 = 0.5$, $d_0 = 0.1$, $d = 0.01$, $\alpha = 0.1$, $\eta = 0.01$).

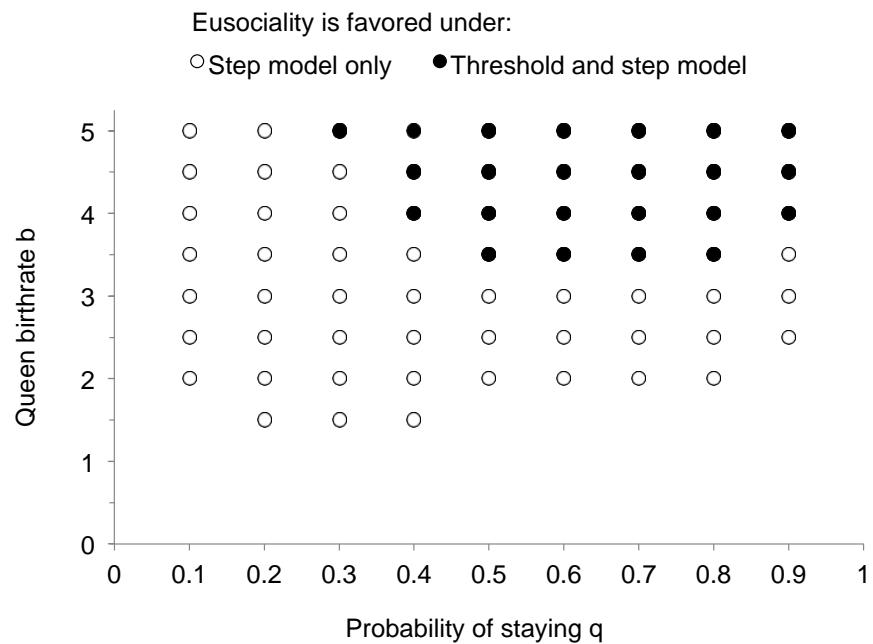


Figure 2.3 Eusocial evolution in the Nowak et al. (2) haploid model, but adding variable relatedness r (equations 3).

Other parameters are as in Fig. 4 of Nowak et al. (2) ($m=3$, $b_0 = 0.5$, $d_0 = 0.1$, $d = 0.01$, $\alpha = 0.1$, $\eta = 0.01$). For any worker-assisted queen birthrate b , reducing relatedness decreases the range of q (offspring probability of staying) supporting eusocial evolution (filled circles). As relatedness declines further, no eusociality does not evolve for any value of q .

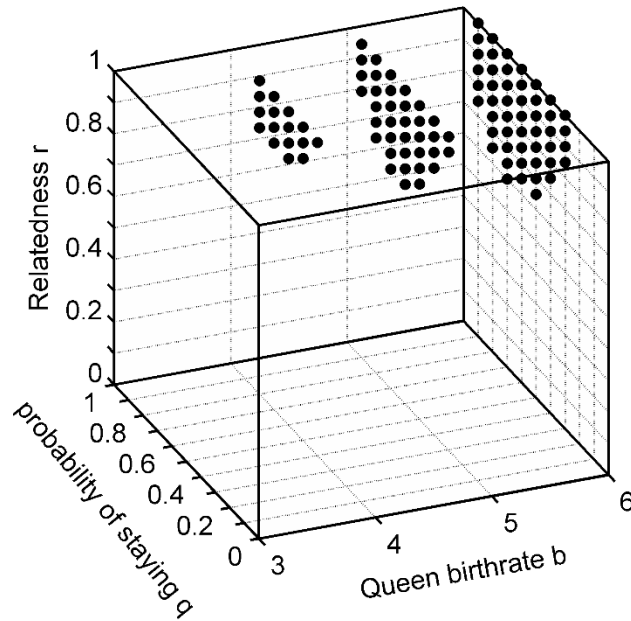
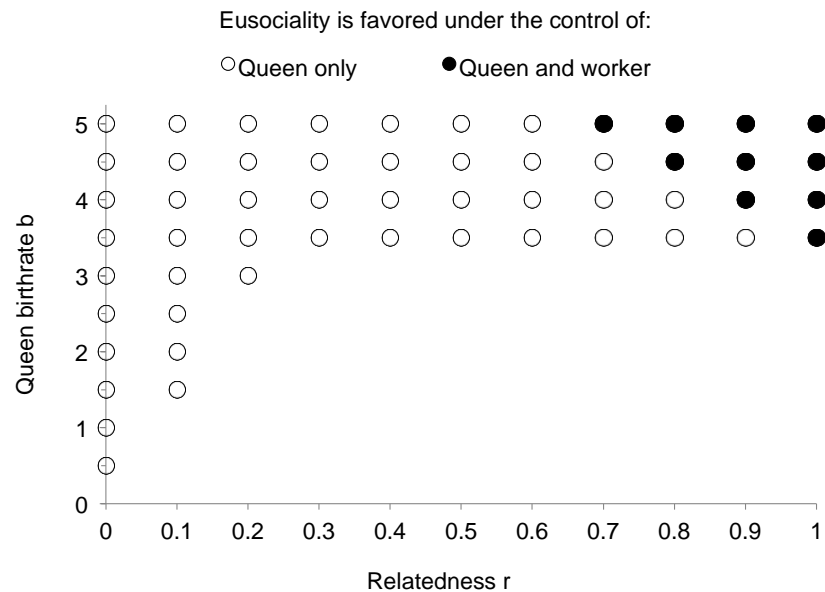


Figure 2.4 Eusocial evolution under offspring versus maternal control.

Filled circles show values of relatedness r and worker assisted queen birthrate b that select for eusociality if the decision is made by offspring (equations 4). When the decision is made by genes acting in mothers (equations 5), eusociality evolves under much broader conditions (open and filled circles). The open circles represent the zone of conflict, when mothers but not offspring favor eusociality. Other parameters are as in Fig. 4 of Nowak et al. (2) ($m=3$, $b_0 = 0.5$, $d_0 = 0.1$, $d = 0.01$, $\alpha = 0.1$, $\eta = 0.01$).



2.8. Supplemental Materials

Inclusive fitness theory predicts that altruistic behavior, such as that shown by workers in eusocial insects, can evolve when sufficient fitness benefits are given to relatives. A different modeling approach has led to a challenge to this theory. The modelers claim that relatedness is not causal, that eusocial behavior is very hard to evolve, and that there is no conflict involved.

2.8.1. Altruism is Easy to Evolve under High Relatedness

Where Nowak *et al.* claim that relatedness is not causal but merely a consequence of eusociality, inclusive fitness theory predicts that relatedness is essential to evolve true altruism. However, Nowak *et al.* did not test the role of relatedness r , as they kept it constant and high in their simulations. Here we revise their asexual model, keeping everything the same except that with probability q helpers join a reproductive at random ($r = 0$) instead of staying with their mother ($r = 1$). If we choose the same parameters that favor eusociality in Nowak *et al.*'s supplemental Figure 4, the unrelated model fails to favor eusociality for any value of q (supplement Figure s1). Even if we increase the work contribution to queen fertility to 50, giving a 100-fold advantage eusociality does not evolve. The reason is obvious: helping of random reproductive gives no benefit to helper genes to compensate for the fitness cost of helping. When r is intermediate with a fraction r

of helpers staying with their own mother, and $1-r$ joining at random, eusociality can be favored, with the minimum queen fertility decreasing as r increases, exactly as predicted from inclusive fitness.

Two strategies: being solitary female or eusocial queen (Figure 2.8.1). The strategy with the faster growth rate wins eventually. Evolutionary dynamics of the two strategies are described by the system of linear differential equations.

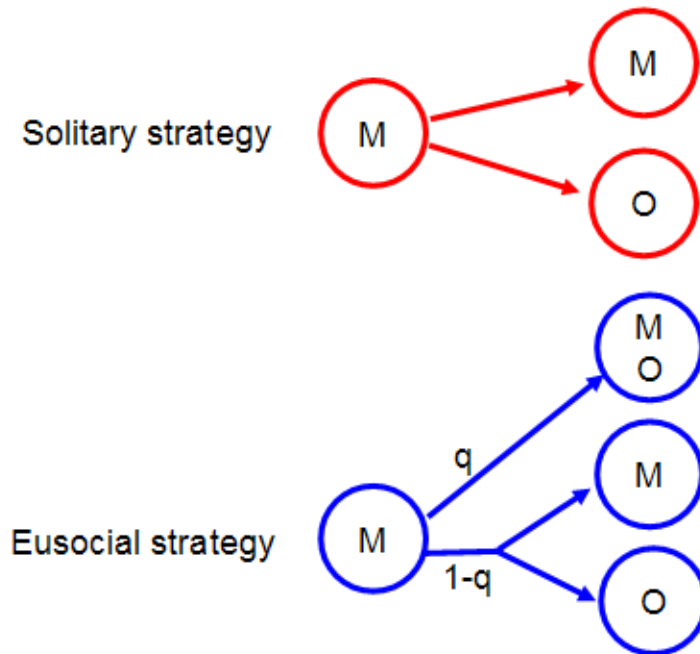


Figure 2.8.1.1 Solitary and Eusocial Strategies

The change in the abundance of solitary females, x_0 and eusocial colonies, x_i with size $i, i = 1, 2, 3, 4, \dots$

$$\dot{x}_0 = (\phi b_0 - d_0)x_0$$

$$\dot{x}_1 = \sum_{i=1}^{\infty} (1-q)\phi b_i x_i - q\phi b_1 x_1 - d_1 x_1 + \alpha x_2$$

$$\dot{x}_i = q\phi b_{i-1} x_{i-1} - q\phi b_i x_i - d_i x_i - \alpha(i-1)x_i + \alpha i x_{i+1} \quad i = 2, 3, \dots$$

Equation 2.8.1 Evolutionary Dynamics under High Relatedness

q Is the probability that the offspring of the eusocial queen stay with the nest; b_0, d_0 is reproductive rate and death rate of solitary females; b_i, d_i is reproductive rate and death rate of a eusocial queen in a nest of size i ; and α is worker's death rate.

We can also write these equations as following. $X_e = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ \vdots \end{bmatrix}$,

$$M_e = \begin{bmatrix} (1-q)b_1 - qb_1 - d_1 & (1-q)b_2 + \alpha & (1-q)b_3 & (1-q)b_4 & \cdots \\ qb_1 & -qb_2 - d_2 - \alpha & 2\alpha & 0 & \cdots \\ 0 & qb_2 & -qb_3 - d_3 - 2\alpha & 3\alpha & \cdots \\ 0 & 0 & qb_3 & -qb_4 - d_4 - 3\alpha & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix};$$

$$\dot{X}_e = M_e X_e .$$

Multiply each birth term with a factor of density limitation $\phi = \frac{1}{1 + \eta X}$, and

the population size $X = x_0 + \sum_i i x_i$.

$$B = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ \vdots \end{bmatrix} ; D = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \\ \vdots \end{bmatrix} .$$

Whether or not eusociality is selected depends on how the demographic parameters of the queen change with colony size. One possibility is to consider a simple step function with a critical colony size, m . For small colonies, $i < m$, the key parameters of the eusocial queen are the same as those of solitary females:

$b_i = b_0, d_i = d_0$. For large colonies, $i \geq m$, the eusocial queen has an increased

fecundity and a reduced death rate: $b_i = b > b_0, d_i = d < d_0$.

Note that intermediate q values ($0.36 < q < 0.9$) are needed for eusociality to evolve (Figures 2.8.1.2 and 2.8.1.3). The intuitive explanation is as follows.

For low q there are too few colonies reach the critical colony size, m , where the advantage of eusociality begins, and too many workers sacrifice for nothing. Since a eusocial queen in colony of size $m-1$ has the same birth rate and death rate as a solitary female, the first $m-2$ workers don't contribute to the advantage of

eusociality until the $(m-1)th$ worker joins in and the queen has increased fecundity and reduced mortality.

On the other hand, if the value of q is too large then the colonies produce too few new queens and too many extra workers. Extra workers than the first $m-1$ ones joining in colonies do not contribute to the advantage of eusociality because queens' fecundity and mortality don't change any more.

Of course, the disadvantage of eusociality is that some of the offspring (workers) do not reproduce; they are subject to worker mortality and they die when the queen dies. Therefore, intermediate values of q allow the evolution of eusociality. There exists relatedness between eusocial queens and workers, and the relatedness is one.

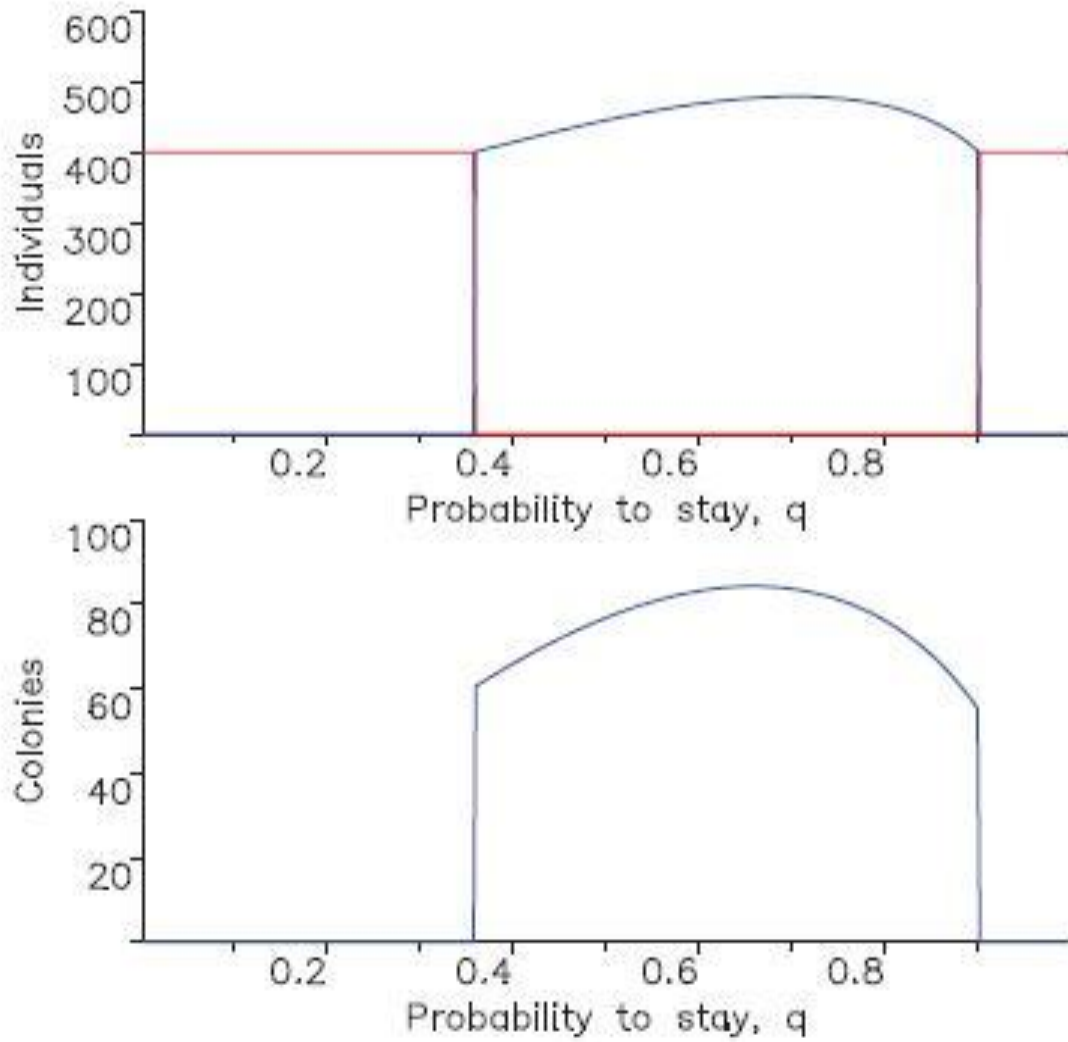


Figure 2.8.1.2 Eusociality Evolves for Intermediate Values of q

$$b_0 = 0.5, d_0 = 0.1, m = 3, b = 4, d = 0.01, \alpha = 0.1, \eta = 0.01.$$

Red, x_0 ; blue, $\sum_i ix_i$; colonies, $\sum_i x_i, i = 2, 3, 4, \dots$

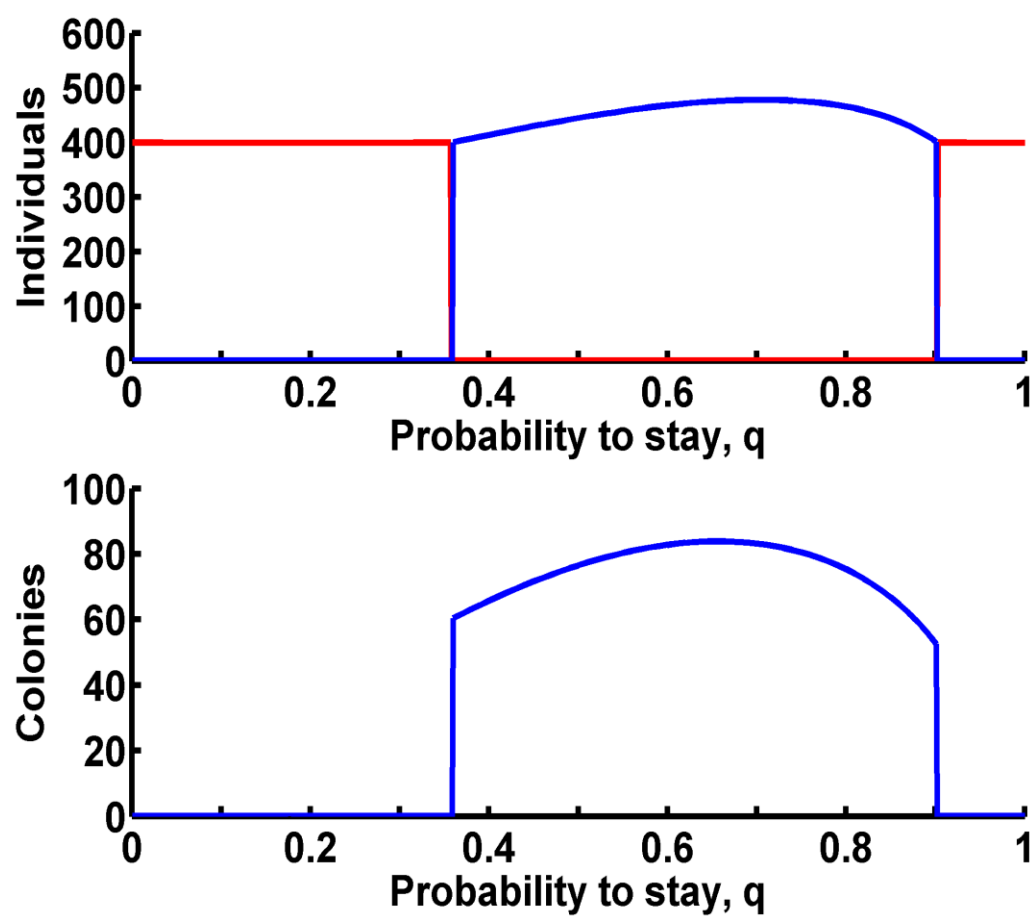


Figure 2.8.1.3 Eusociality Evolves under High Genetic Relatedness

$$b_0 = 0.5, d_0 = 0.1, m = 3, b = 4, d = 0.01, \alpha = 0.1, \eta = 0.01.$$

Red, x_0 ; blue, $\sum_i ix_i$; colonies, $\sum_i x_i, i = 2, 3, 4, \dots$

2.8.2. Altruism cannot Evolve without Relatedness

The offspring of the eusocial queens migrate randomly among all nests (including eusocial and solitary ones) with probability q ; they leave the colony to build their own eusocial ones with probability $1 - q$.

Solitary females can and only can acquire workers from eusocial colonies, and thus they have increased fecundity and reduced mortality in the presence of workers, just the same as eusocial queens. We assume a solitary female has the same reproductive rate and death rate as a eusocial queen in a colony of the same size. Since they can't acquire workers from solitary nests, eusocial queens pay all the cost of producing workers, but every one reaps the gains of the workers randomly. Therefore, since there is no longer any correlation between mother and helper genes, the relatedness in average is zero between queens and workers (Figure 2.8.2).

Just like the definition of x_i , let s_i denote the abundance of solitary nest with size i . Here $i = 1, 2 \dots$ represents the number of individuals in the colony including the solitary female. Thus, s_1 denotes nests with single solitary females, while s_2 denotes nests where a solitary female has one worker, and so on. A solitary female in a nest of size i has the same reproductive rate b_i and death rate d_i as a eusocial queen in a colony of the same size. Denote

$$\bar{b} = \frac{\sum_{i=1}^{\infty} q b_i x_i}{\sum_{i=1}^{\infty} s_i + \sum_{i=1}^{\infty} x_i}$$

Equation 2.8.2 Average Birth Rate Among Colonies Without Genetic Relatedness

The numerator is the total number of migrating workers, and the denominator is the total number of nests. So \bar{b} is the proportion of migrating workers that each nest acquires. Those workers which join in the eusocial and solitary nests of size i are $\bar{b}x_i$, $\bar{b}s_i$, respectively. Those offspring who leave the eusocial and solitary nests of size i to build their own are $(1-q)b_i x_i$, $b_i s_i$.

The equations (58) change to

$$\dot{x}_1 = \sum_{i=1}^{\infty} (1-q)\phi b_i x_i - \phi \bar{b} x_1 - d_1 x_1 + \alpha x_2$$

$$\dot{x}_i = \phi \bar{b} x_{i-1} - \phi \bar{b} x_i - d_i x_i - \alpha(i-1)x_i + \alpha i x_{i+1} \quad i = 2, 3, \dots$$

$$\dot{s}_1 = \sum_{i=1}^{\infty} \phi b_i s_i - \phi \bar{b} s_1 - d_1 s_1 + \alpha s_2$$

$$\dot{s}_i = \phi \bar{b} s_{i-1} - \phi \bar{b} s_i - d_i s_i - \alpha(i-1)s_i + \alpha i s_{i+1} \quad i = 2, 3, \dots$$

Equation 2.8.3 Evolutionary Dynamics without Genetic Relatedness

We can also write these equations as following.

$$X_e = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ \vdots \end{bmatrix}; X_s = \begin{bmatrix} s_1 \\ s_2 \\ s_3 \\ s_4 \\ \vdots \end{bmatrix};$$

$$M_e = \begin{bmatrix} (1-q)b_1 - \bar{b} - d_1 & (1-q)b_2 + \alpha & (1-q)b_3 & (1-q)b_4 & \cdots \\ \bar{b} & -\bar{b} - d_2 - \alpha & 2\alpha & 0 & \cdots \\ 0 & \bar{b} & -\bar{b} - d_3 - 2\alpha & 3\alpha & \cdots \\ 0 & 0 & \bar{b} & -\bar{b} - d_4 - 3\alpha & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}.$$

$$M_s = \begin{bmatrix} b_1 - \bar{b} - d_1 & b_2 + \alpha & b_3 & b_4 & \cdots \\ \bar{b} & -\bar{b} - d_2 - \alpha & 2\alpha & 0 & \cdots \\ 0 & \bar{b} & -\bar{b} - d_3 - 2\alpha & 3\alpha & \cdots \\ 0 & 0 & \bar{b} & -\bar{b} - d_4 - 3\alpha & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix};$$

$\dot{X}_e = M_e X_e$; $\dot{X}_s = M_s X_s$. Multiply each birth term with a factor of density

limitation $\phi = \frac{1}{1 + \eta X}$, and the population size $X = \sum_i (ix_i + is_i)$.

$$B = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ \vdots \end{bmatrix}; D = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \\ \vdots \end{bmatrix}; \bar{B} = \begin{bmatrix} \bar{b} \\ \bar{b} \\ \bar{b} \\ \bar{b} \\ \vdots \end{bmatrix}.$$

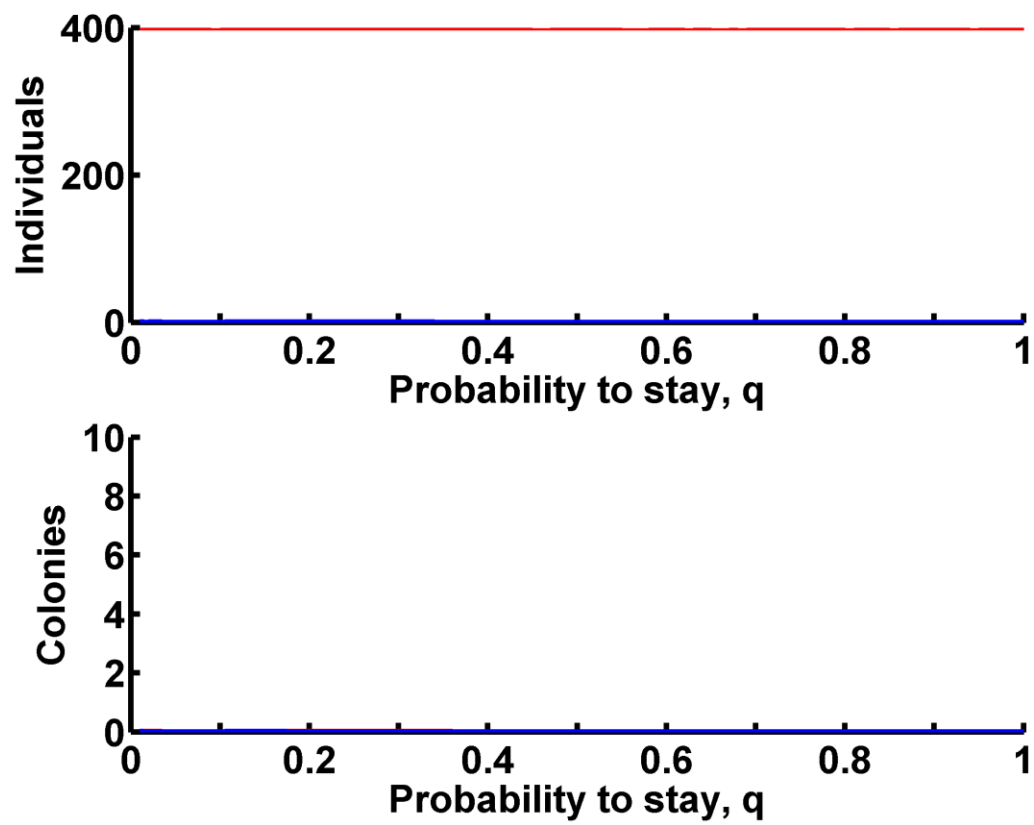


Figure 2.8.2 Eusociality can't evolve without genetic relatedness

2.8.3. The Higher the Relatedness, the Easier Altruism to Evolve

The offspring of the eusocial queens stay with the nest with probability pq ; they migrate randomly among all nests (including eusocial and solitary ones) with probability $(1-p)q$; they leave the colony to build their own eusocial ones with probability $1-q$. Therefore, the relatedness in average is in range of zero and one between queens and workers, $r = p$. Denote

$$\bar{b} = \frac{\sum_{i=1}^{\infty} (1-p)qb_i x_i}{\sum_{i=1}^{\infty} x_i + \sum_{i=1}^{\infty} s_i}$$

Equation 2.8.4 Average Birth Rate Among Colonies With Various Genetic Relatedness

The numerator is the total number of migrating workers, and the denominator is the total number of nests. So \bar{b} is the proportion of migrating workers that each nest acquires randomly.

Those workers which stay in the eusocial nests of size i are $pqb_i x_i$. Those workers which join randomly in the eusocial queen nests of size i are $\bar{b}x_i$. Those workers from eusocial colonies which join randomly in the solitary nests of size i are $\bar{b}s_i$. Those offspring who leave the eusocial colonies and solitary nests of size i to build their own ones are $(1-q)b_i x_i$, $b_i s_i$, respectively.

The equations (58) change to

$$\dot{x}_1 = \sum_{i=1}^{\infty} (1-q)\phi b_i x_i - pq\phi b_1 x_1 - \phi \bar{b} x_1 - d_1 x_1 + \alpha x_2$$

$$\dot{x}_i = pq\phi b_{i-1} x_{i-1} + \phi \bar{b} x_{i-1} - pq\phi b_i x_i - \phi \bar{b} x_i - d_i x_i - \alpha(i-1)x_i + \alpha x_{i+1} \quad i = 2, 3, \dots$$

$$\dot{s}_1 = \sum_{i=1}^{\infty} \phi b_i s_i - \phi \bar{b} s_1 - d_1 s_1 + \alpha s_2$$

$$\dot{s}_i = \phi \bar{b} s_{i-1} - \phi \bar{b} s_i - d_i s_i - \alpha(i-1)s_i + \alpha s_{i+1} \quad i = 2, 3, \dots$$

Equation 2.8.5 Evolutionary Dynamics under Various Genetic Relatedness

We can also write these equations as following.

$$X_e = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ \vdots \end{bmatrix}; X_s = \begin{bmatrix} s_1 \\ s_2 \\ s_3 \\ s_4 \\ \vdots \end{bmatrix}; B = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ \vdots \end{bmatrix}; D = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \\ \vdots \end{bmatrix}; \bar{B} = \begin{bmatrix} \bar{b} \\ \bar{b} \\ \bar{b} \\ \bar{b} \\ \vdots \end{bmatrix}.$$

$$M_e =$$

$$\begin{bmatrix} (1-q)b_1 - pqb_1 - \bar{b} - d_1 & (1-q)b_2 + \alpha & (1-q)b_3 & (1-q)b_4 & \dots \\ pqb_1 + \bar{b} & -pqb_2 - \bar{b} - d_2 - \alpha & 2\alpha & 0 & \dots \\ 0 & pqb_2 + \bar{b} & -pqb_3 - \bar{b} - d_3 - 2\alpha & 3\alpha & \dots \\ 0 & 0 & pqb_3 + \bar{b} & -pqb_4 - \bar{b} - d_4 - 3\alpha & \dots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

$$M_s = \begin{bmatrix} b_1 - \bar{b} + d_1 & b_2 + \alpha & b_3 & b_4 & \cdots \\ \bar{b} & -\bar{b} - d_2 + \alpha & 2\alpha & 0 & \cdots \\ 0 & \bar{b} & -\bar{b} - d_3 + 2\alpha & 3\alpha & \cdots \\ 0 & 0 & \bar{b} & -\bar{b} - d_4 - 3\alpha & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix};$$

$$\dot{X}_e = M_e X_e; \quad \dot{X}_s = M_s X_s.$$

Multiply each birth term with a factor of density limitation $\phi = \frac{1}{1 + \eta X}$, and

the population size $X = \sum_i (ix_i + is_i)$.

The Figure 2.8.3 shows that the higher the relatedness, the easier altruism to evolve.

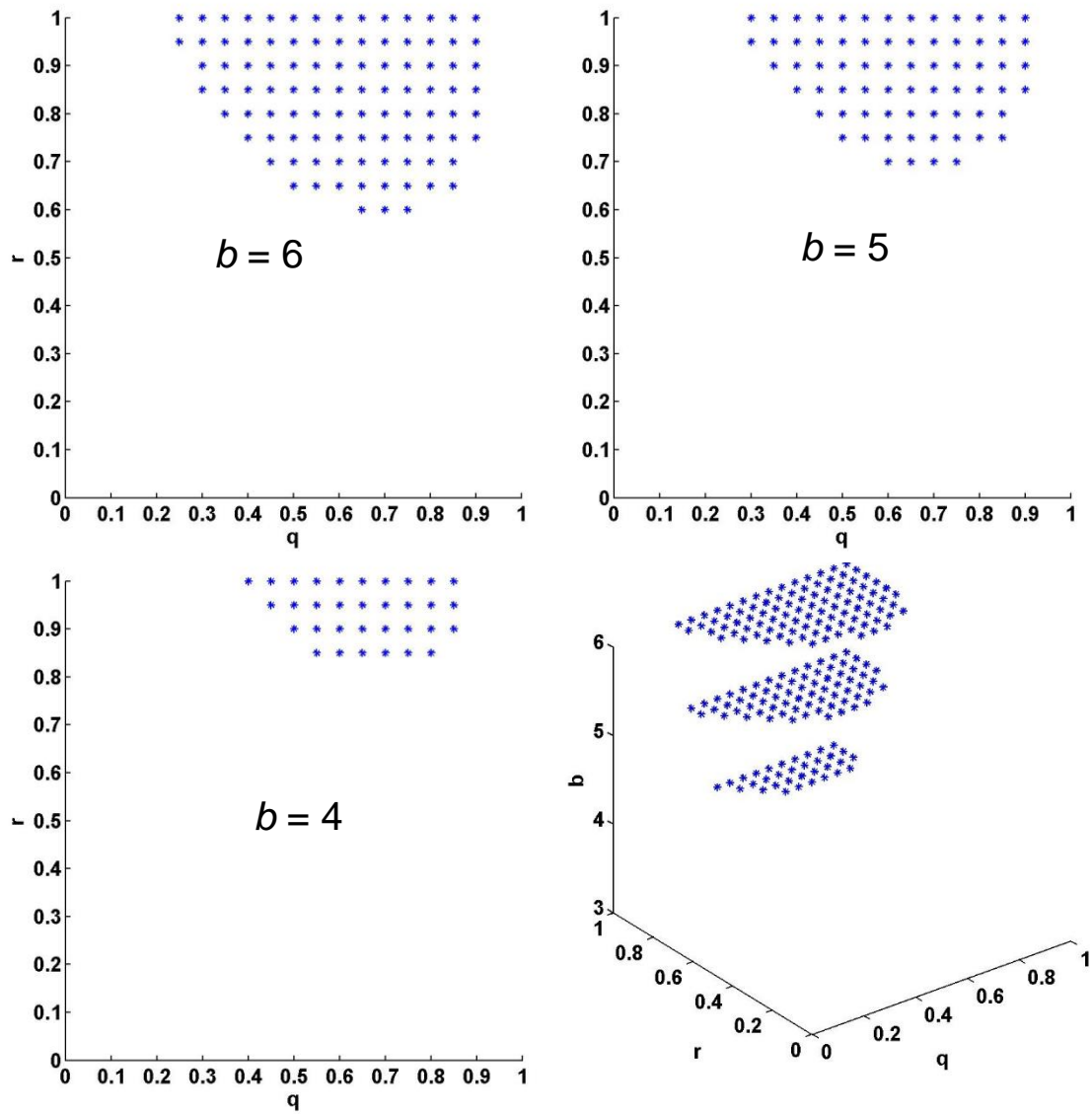


Figure 2.8.3 The Higher the Relatedness, the Easier Altruism to Evolve

2.8.4. Eusociality is Easy to Evolve under Flexible Strategies of Workers

Another surprising claim of Nowak *et al.* was how difficult eusociality was to evolve; it required a very high increase in queen birth rate, and no decrease in queen death rate was sufficient in itself. However, these results are due largely to assumptions of the particular models that are neither biologically realistic nor necessary to the modeling approach. The particular assumptions responsible are that queen fitness is a step function with a single upward step at a critical colony size m (usually $m = 3$ in their examples) and that worker behavior is inflexible – a fraction q of them stay, independent of circumstance. One solution is to remove the threshold from the model – there is little empirical evidence that benefits kick in only after surpassing a critical worker number – but even if thresholds reflect reality, the worker behavior does not. If we use inclusive fitness think that about worker options under a step function, we see two problems associated with the inflexible strategy. First, when a worker joins below the threshold, it contributes nothing until enough other workers join, and it may have to wait some time as a fraction $1-q$ of its sibling leave. Second, after the threshold has been surpassed, the fraction of workers who later join contributes nothing further (except for staying above the threshold when previous workers die). A simple strategy that would minimize these problems is for workers to always join when the colony is below the cap colony size m' and always leave when the colony has reached that size, where the optimum m' will likely be near the critical colony size m . We have implemented this strategy for the asexual model with a new derived transition matrix. Running

the model shows that it does indeed make eusociality pay at much lower queen fecundities (b values) than inflexible strategy (Figure 2.8.4). It also becomes possible to select for eusociality purely by decreasing the queen's death rate.

With the same death rate d we expect lower birth rate b for eusociality to evolve in the modified model than that in Nowak *et al.*

Offspring of colony size, i , stay with $p_i q_i$, migrate with $(1 - p_i) q_i$, and leave with $1 - q_i$.

Assume low and upper threshold of colony size, $m_s, m_l = 1, 2, 3, 4, \dots, m_s \leq m_l$.

When $i < m_s$, then offspring always stay ($p_i = 1, q_i = 1$); when $i \geq m_l$, then offspring always leave ($p_i = 0, q_i = 0$); when $m_s \leq i < m_l$, $p_i = p, q_i = q$. If $m_s = m_l = m$, then offspring always stay when $i < m$ and leave when $i \geq m$.

So the first $m_s - 1$ workers always stay, and the extra workers over $m_l - 1$ always leave; the intermediate workers either stay or migrate. This will make eusociality easier because (1) they will reach the critical colony size m_s quicker, and (2) they won't have useless additional workers above the upper threshold of colony size. We could also combine it with the step-forward birth and death rates for the first $m_s - 1$ workers. Denote

$$\bar{b}_m = \frac{\sum_{i=1}^{\infty} (1-p_i)q_i b_i x_i}{\sum_{i=1}^{\infty} x_i + \sum_{i=1}^{\infty} s_i}$$

Equation 2.8.6 Average Birth Rate Among Colonies With Flexible Strategies of Workers

The equations (58) change to

$$\dot{x}_1 = \sum_{i=1}^{\infty} (1-q_i)\phi b_i x_i - p_1 q_1 \phi b_1 x_1 - \phi \bar{b} x_1 - d_1 x_1 + \alpha x_2$$

$$\dot{x}_i = p_i q_i \phi b_{i-1} x_{i-1} + \phi \bar{b} x_{i-1} - p_i q_i \phi b_i x_i - \phi \bar{b} x_i - d_i x_i - \alpha(i-1)x_i + \alpha i x_{i+1} \quad i = 2, 3, \dots$$

$$\dot{s}_1 = \sum_{i=1}^{\infty} \phi b_i s_i - \phi \bar{b} s_1 - d_1 s_1 + \alpha s_2$$

$$\dot{s}_i = \phi \bar{b} s_{i-1} - \phi \bar{b} s_i - d_i s_i - \alpha(i-1)s_i + \alpha i s_{i+1} \quad i = 2, 3, \dots$$

We can also write these equations as following.

$$X_e = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ \vdots \end{bmatrix}; \quad X_s = \begin{bmatrix} s_1 \\ s_2 \\ s_3 \\ s_4 \\ \vdots \end{bmatrix};$$

$$M_e =$$

$$\begin{bmatrix} (1-q_1)b_1 - p_1q_1b_1 - \bar{b} - d_1 & (1-q_2)b_2 + \alpha & (1-q_3)b_3 & (1-q_4)b_4 & \dots \\ p_1q_1b_1 + \bar{b} & -p_2q_2b_2 - \bar{b} - d_2 - \alpha & 2\alpha & 0 & \dots \\ 0 & p_2q_2b_2 + \bar{b} & -p_3q_3b_3 - \bar{b} - d_3 - 2\alpha & 3\alpha & \dots \\ 0 & 0 & p_3q_3b_3 + \bar{b} & -p_4q_4b_4 - \bar{b} - d_4 - 3\alpha & \dots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

$$M_s = \begin{bmatrix} b_1 - \bar{b} - d_1 & b_2 + \alpha & b_3 & b_4 & \dots \\ \bar{b} & -\bar{b} - d_2 - \alpha & 2\alpha & 0 & \dots \\ 0 & \bar{b} & -\bar{b} - d_3 - 2\alpha & 3\alpha & \dots \\ 0 & 0 & \bar{b} & -\bar{b} - d_4 - 3\alpha & \dots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix};$$

$$\dot{X}_e = M_e X_e;$$

$$\dot{X}_s = M_s X_s.$$

Multiply each birth term with a factor of density limitation, $\phi = \frac{1}{1 + \eta X}$, and

the population size $X = \sum_i (ix_i + is_i)$.

$$B = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ \vdots \end{bmatrix}; D = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \\ \vdots \end{bmatrix}; P = \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \\ \vdots \end{bmatrix}; Q = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \\ \vdots \end{bmatrix}$$

Assume threshold of work number $w-1$, $w = 1, 2, 3, \dots$. When colony size $i < w$, then offspring always stay; when $i \geq w$, then offspring always leave.

Denote the abundance of solitary females, x_0 , and eusocial colonies, x_i , with size $i, i = 1, 2, 3, 4, \dots$. And b_0, d_0 is reproductive rate and death rate of solitary females; b_i, d_i is reproductive rate and death rate of a eusocial queen in a nest of size i ; and α is worker's death rate. Multiply each birth term with a factor of density

limitation $\phi = \frac{1}{1 + \eta X}$, and the population size $X = x_0 + \sum_i i x_i$.

$$\dot{x}_0 = (\phi b_0 - d_0)x_0$$

$$\dot{x}_1 = \sum_{i=w}^{\infty} \phi b_i x_i - \phi b_1 x_1 - d_1 x_1 + \alpha x_2$$

$$\dot{x}_i = \phi b_{i-1} x_{i-1} - \phi b_i x_i - d_i x_i - \alpha(i-1)x_i + \alpha i x_{i+1} \quad i = 2, 3, \dots, w-1$$

$$\dot{x}_i = \phi b_{i-1} x_{i-1} - d_i x_i - \alpha(i-1)x_i + \alpha i x_{i+1} \quad i = w$$

$$\dot{x}_i = -d_i x_i - \alpha(i-1)x_i + \alpha i x_{i+1} \quad i = w+1, w+2, \dots$$

Equation 2.8.7 Evolutionary Dynamics With Flexible Strategies of Workers

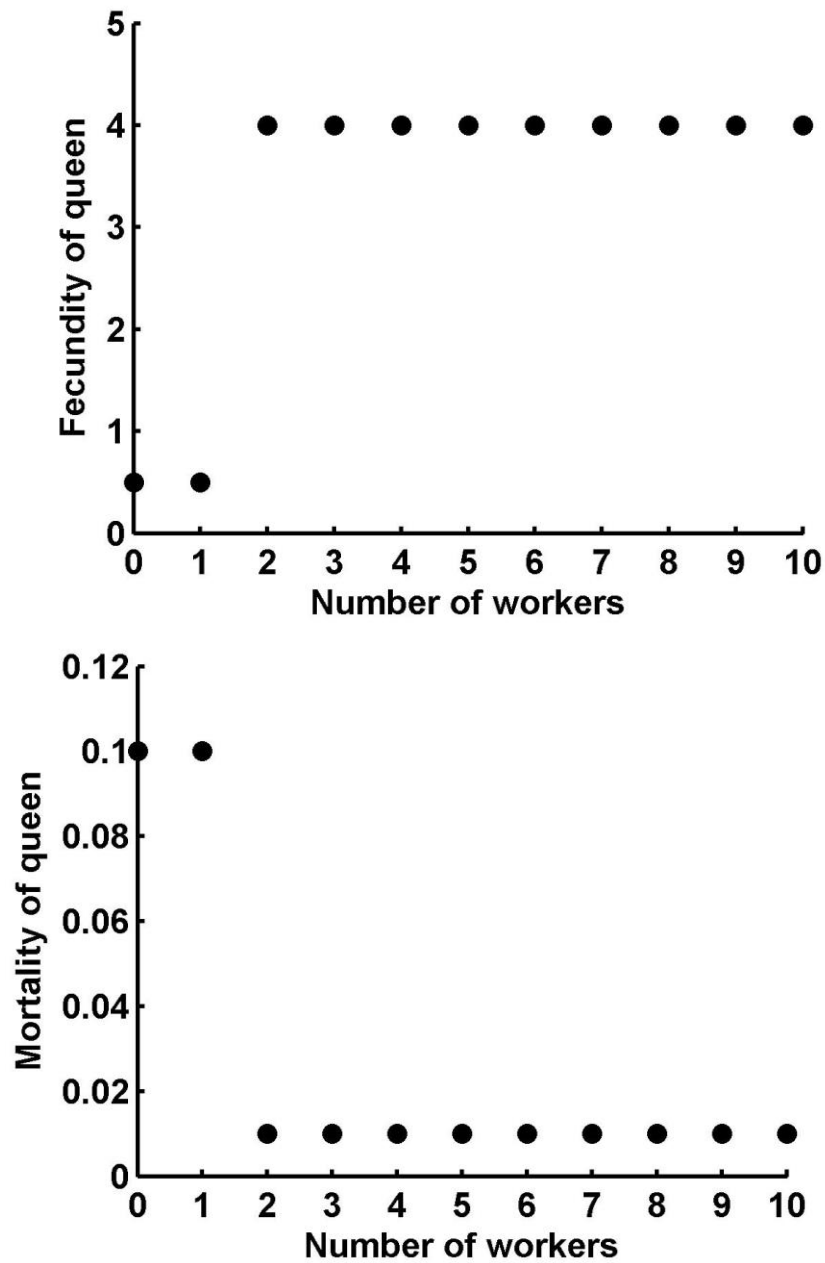


Figure 2.8.4 Flexible Strategies with Threshold of Workers Number

2.8.5. Conclusions

Here I showed that, even within the terms of this modeling framework, inclusive fitness thinking leads to insights that completely change these conclusions. I showed that relatedness is causal, that eusociality is not so difficult to evolve, and that there can be conflict between queens and workers. I concluded that multiple modeling approaches are useful and that efforts to synthesize them are better than asserting that one is universally better than the other.

Evolutionary Dynamics of Genetic Kin Recognition: a General Model

3.1. Abstract

Altruism is a behavior that benefits others at a cost to one's own ability of survival and/or reproduction. Either greenbeard effects or genetic kin recognition requires genetic polymorphisms as cues on which recognition is based. Previous models show that rare cue alleles get eliminated by selection and a common allele gets fixed, which ruins the altruism system. So it is unclear how genetic recognition for altruism persists. Here, I designed a novel model with three types of genetic components, production, perception, and action. Our recognition model suggested the stability of recognition for altruism that altruism can maintain multiple recognition cues and be evolutionarily stable.

3.2. Introduction

In 1964 W. D. Hamilton produced an elegant formal theory that provided a potential solution to this problem (Hamilton 1964). Hamilton argues that altruistic acts to relatives can be favored by natural selection, because relatives share the same gene as helpers. Hamilton expanded the definition of fitness in terms of inclusive fitness which is the sum of a direct benefit through producing offspring and an indirect benefit through aiding genetic relatives. Hamilton made these two components additive by devaluing each offspring or relative by the genetic relatedness to them.

From this Hamilton predicted that altruism will be favored by natural selection when the inequality $rB - C > 0$ is satisfied, where B is the benefit of the act of altruism to the recipient, C the cost of the act to the actor and r the genetic relatedness between the actor and the recipient (Hamilton 1964). Inclusive fitness is applicable not only to helping but also to any behavior (West et al. 2007b). This inequality has now come to be known as Hamilton's rule. Hamilton's theory is also frequently referred to as the inclusive fitness theory or kin selection.

However, testing Hamilton's rule without measuring the cost and benefit of eusociality is an inadequate test. Focusing only on relatedness and neglecting the cost and benefit terms usually takes the form of assuming implicitly that $B = C$. In my thesis I plan to design a novel model of genetic kin recognition and to explore

the range of fitness costs and benefits with the model to explain how eusociality could evolve and persist.

Kin selection depends on the identification of relatives. Recognition is often accomplished through genetic systems that use variable genetic cues (the production component), detection and evaluation of those cues (the perception component), and the social behavior (the action component) (Table 3.1).

In the models below, I consider selection on a haploid organism, with cue loci and perception loci that are unlinked. For simplicity disequilibria generated by selection are ignored (though they will not be ignored in the computer models). Each individual is faced with the choice of helping a partner. The partner will be a relative related by r a fraction p of the time; the remaining $1-p$ of the time the partner will be unrelated. Altruism is determined by matching at the cue locus and evaluation at the perception locus. When a match at the cue locus occurs, and when the perception locus perceives and passes on the match, the individual loses $-c$ units of fitness but gives its partner b units of fitness (Table 3.2).

3.3. Negative Feedbacks from *Judge* to *Cue*

In our model, actor recognizes and helps partner if the following two requirements of recognition are fulfilled (Table 3.1 and 3.2). The first requirement is that *Cue* alleles of actor and partner match each other. So the *Cue* locus is a greenbeard locus. (The effect of matches at the *Cue* locus is to help others who

actually have the allele.) . When $c < b$, *Cue* alleles can be favored by altruism, and common *Cue* alleles are more likely to get benefits than rare *Cue* alleles. The second requirement is that actor's *Judge* allele accepts the matches of *Cue* alleles. It doesn't matter what partner's *Judge* allele is. In our model, actor's J_1 accepts the matches of C_1 and rejects all other *Cue* allele matches. J_2 rejects all *Cue* allele matches except C_2 . So C_1 is favored by J_1 and disfavored by J_2 . C_2 is favored by J_2 and disfavored by J_1 . Where there are no relatives being aided, the *Judge* locus does not benefit easily from altruism since it's unlinked to the greenbeard *Cue* locus. It benefits only when altruism helps copies of the *Judge* allele in partner. So the best thing for *Judge* alleles is not to do any altruism at all. When altruism has a chance to be performed to relatives, and $c < rb$, both *Cue* and *Judge* alleles can be favored since relatives are likely to share the same allele at each locus. We expect that when a common *Cue* allele, let's say C_1 , causes too much altruism from the view of *Judge* alleles, J_1 that accept the matches of C_1 will be disfavored much more than other *Judge* alleles. This in turn will cause the frequency of common C_1 to come down. In contrast, when rare C_1 causes too little altruism, J_1 will be disfavored much less than other *Judge* alleles. This in turn will cause the frequency of rare C_1 to go up. Clearly, *Cue* alleles have negative feedbacks from *Judge* alleles, which could lead allele frequencies oscillate over time, and thereby maintain all alleles.

3.4. Genetic Relatedness at Each of Three Loci

The genetic relatedness for each of three kinds of loci takes into account the helping of partners with the same allele, above random levels of that allele (Table 3.3). To the extent that they help individuals with random levels of the allele, that benefit won't change gene frequencies and therefore don't contribute to relatedness. In each case the denominator is all matches at the cue locus, and the numerator is the proportion of those matches that are identical-above-random at the locus in question (*Cue*, *Judge*, or *Act*).

For the cue allele, the genetic relatedness is 1 because the numerator is the same as the denominator, which consists of two proportions: 1) the proportion of clonemate partners, p ; and 2) the proportion of non-relative partners bearing the matched cue allele, $(1 - p)f_{K_j}$ (Table 3.3, Figure 3.1). For all matches at the cue locus, the beneficiary will share the altruist's cue allele above random levels. Even though some of the partners are random non-relatives, the ones who get aided are the non-random set that carries the matched cue allele. Usually it is relatedness that causes non-random identities, but here it is the matching process.

For genetic relatedness of the judge allele, when there are matches at the cue locus (denominator), the fraction which gives matches at the judge locus above random level is only the proportion of clonemates, p (Table 3.3, Figure 3.1). There will be some additional matches at the judge locus for non-relatives, but there are

random matches and will be exactly canceled out by non-matches among the non-relatives. Among the non-relatives, judge alleles are being helped at their population frequencies, so they don't contribute to selection and therefore don't contribute to the relatedness.

For the action allele, when there are matches at the cue locus, the fraction which gives matches at the judge locus above random level is only the proportion of clonemates, p (Table 3.3, Figure 3.1).

The relatedness of the cue takes into account all matches but relatedness of the judge and the action only counts some of them because none of the matches are random for the cue locus but some of them are random for the judge locus and the action locus.

The genetic relatedness for each of three kinds of loci is the same as the maximum c/b value that will favor altruism for that locus (Figure 3.1).

3.5. Mathematical Analysis of the Model

Mathematical Analysis of the Model for one production locus with multiple cue alleles, one perception locus with multiple judge alleles, and one action locus with one act allele (Table 3.2)

Actor's recognition of its partner is based on their genetic loci: actor's cue allele k_x and judge allele j_x , and partner's cue allele k_y and judge allele j_y . Actor

performs altruistic behavior to its partner only when both of two following requirements are fulfilled: 1) their cue alleles match each other, denoted by $k_x = k_y$; and then 2) actor's judge allele accepts matches of their cues, denoted by $k_x = k_y = j_x$. Partner's judge allele doesn't involve in the recognition process. We assume each judge allele only accepts matches of the certain cue. Let cue alleles in the population be k_1, k_2, \dots , judge alleles be j_1, j_2, \dots , and j_1 accept matches of k_1 , j_2 accept matches of k_2 ..., and so on.

Let g_{k_x, j_x} is actor's genotype with cue allele k_x and judge allele j_x , and g_{k_y, j_y} is partner's genotype with cue allele k_y and judge allele j_y . Assume each individual has a fraction, p , that partner is its clonemate. Then frequency of altruism in the population f_a is

$$f_a = p \sum_{k_x = j_x} g_{k_x, j_x} + (1 - p) \sum_{k_x = k_y = j_x, j_y} g_{k_x, j_x} g_{k_y, j_y}$$

Equation 3.1 Frequency of Altruism in the Population

where $k_x, k_y = k_1, k_2, \dots$; $j_x, j_y = j_1, j_2, \dots$. The former fraction, $p \sum_{k_x = j_x} g_{k_x, j_x}$, is frequency of altruisms performed between clonemates. Since each individual has a clonemate and they share the same alleles, altruistic behavior will be performed when judge allele accepts matches of cues. The latter fraction, $(1 - p) \sum_{k_x = k_y = j_x, j_y} g_{k_x, j_x} g_{k_y, j_y}$, is

frequency of altruisms performed between nonrelatives. It not only requires matches of their cues, but also those matches must be accepted by actor's judge allele.

The frequency of altruism by descent is

$$f_d = p \sum_{j=k} g_{j,k}$$

Equation 3.2 The Frequency of Altruism by Descent

Let the cost of altruistic behavior be c , the benefit be b , and initialized fitness of each individual be w_0 . Then, the mean fitness of the population after altruism is

$$\bar{w} = w_0 + (b - c)f_d$$

Equation 3.3 The Mean Fitness of the Population after Altruism

The mean fitness of individuals bearing the cue allele k , $k = k_1, k_2, \dots$, and the judge allele j , $j = j_1, j_2, \dots$, is

$$w_{k,j} = \begin{cases} w_0 + p(b - c) - (1 - p)c \sum_{k=k_y=j, j_y} g_{k_y, j_y} + (1 - p)b \sum_{k_x=k=j_x} g_{k_x, j_x}, & k = j \\ w_0 + (1 - p)b \sum_{k_x=k=j_x} g_{k_x, j_x}, & k \neq j \end{cases}$$

Equation 3.4 The Mean Fitness of Individuals

where $k_x, k_y = k_1, k_2 \dots$; $j_x, j_y = j_1, j_2 \dots$. This is because when its judge allele could accept matches of cues ($k = j$), the fitness of the individual having cue k and judge j allele will be influenced in three situations. 1) With a fraction p the individual encounters its clonemate. Since the clonemates share the same cue allele and its judge allele accepts the matches of these cues, the actor helps the clonemate. The actor pays the cost of c and its clonemate receives the benefit of b . Thus, it increases their mean fitness by $p(b - c)$. 2) With a fraction $1 - p$ the individual encounters nonrelatives. It helps those who share the same cue allele with it and pays the cost of c each time. Thus, it decreases its fitness by $(1 - p)c \sum_{k=k_y=j_y} g_{k_y, j_y}$. 3) In reverse it may obtain benefits from nonrelatives who provide help to it and thus increase its fitness by $(1 - p)b \sum_{k_x=k=j_x} g_{k_x, j_x}$. When its judge allele can never accept matches of cues ($k \neq j$), the individual will never help others but can receive help from nonrelatives and get benefits of $(1 - p)b \sum_{k_x=k=j_x} g_{k_x, j_x}$.

Let frequencies of cue and judge alleles be $f_{k_1}, f_{k_2} \dots$ (or $p_1, p_2 \dots$) and $f_{j_1}, f_{j_2} \dots$ (or $q_1, q_2 \dots$), respectively. Offspring of the next non-overlapping generation are sexually reproduced by random parents chosen proportionally to their fitness. Let the recombination rate between the cue and judge loci be h . We assume no mutation. Frequencies of genotypes become

$$g'_{k,j} = (1-h)g_{k,j} \frac{w_{k,j}}{w} + hf_k \frac{w_k}{w} f_j \frac{w_j}{w}$$

Equation 3.5 The Frequencies of Genotypes

where w_k and w_j are the mean fitness of cue allele k and judge allele j , respectively.

This follows because a fraction $(1-h)$ of the haplotypes in the offspring have not recombined, and are thus copies of a random haplotype in their parents. A fraction

$g_{j,k} \frac{w_{j,k}}{w}$ of those are the probability of the haplotypes chosen as parents, which is

the production of the haplotype frequency and its proportional fitness. A fraction h of the haplotypes have recombined the cue and judge loci. Since the parents result from random mating, the probability of the copy at cue locus having allele k is $f_k \frac{w_k}{w}$

and the probability of the copy at judge locus having allele j is $f_j \frac{w_j}{w}$, and as these

copies are initially on different haplotypes, these are independent events so that the probabilities can be multiplied.

The frequencies of cue and judge alleles become

$$f'_k = f_k \frac{w_k}{w} = \frac{1}{w} \sum_j g_{k,j} w_{k,j}, k = k_1, k_2 \dots$$

Equation 3.6 The Frequencies of Cue Alleles

$$f'_j = f_j \frac{w_j}{w} = \frac{1}{w} \sum_k g_{k,j} w_{k,j}, j = j_1, j_2 \dots$$

Equation 3.7 The Frequencies of Judge Alleles

So frequencies of genotypes also become

$$g'_{k,j} = (1-h)g_{k,j} \frac{w_{k,j}}{w} + h \frac{1}{w} \left(\sum_j g_{k,j} w_{k,j} \right) \left(\sum_k g_{k,j} w_{k,j} \right), k = k_1, k_2 \dots, j = j_1, j_2 \dots$$

Equation 3.8 The Frequencies of Genotypes

3.6. Prediction of Stability of Evolution of Altruism and Eusociality

What it takes into account is the helping of partners with the same allele, above random levels of that allele. To the extent that they help individuals with random levels of the allele, that benefit won't change gene frequencies). In each case the denominator is all matches at the cue locus, and the numerator is the proportion of those matches that are also identical-above-random at the locus in question (cue, judge, or act) (Figures 3.1, 3.2, and 3.3).

So for the cue allele, $r=1$ because the numerator is the same as the denominator (same denominator as for judge r) (Figures 3.1, 3.2, and 3.3). For all matches at the cue locus, the beneficiary will share the altruists cue allele above random levels. Even though some of the partners are random non relatives, the

ones who get aided are the non-random set that carry the matched cue allele. Usually it is relatedness that causes non-random identities, but here it is the matching process.

For the judge r , when there are matches at the cue locus (denominator), the fraction give matches at the judge locus above random levels is p , the proportion of clonemates (Figures 3.1, 3.2, and 3.3). There will be some additional matches at the judge allele for non-relatives, but these are random matches and will be exactly canceled out by non-matches among the non-relatives. Among the non-relatives, judge alleles are being helped at their population frequencies, so they do not contribute to selection (and therefore don't contribute to r). So what is confusing is that counts all matches for the cue r , but only some of them for the judge r . It's because none of the matches are random for the cue locus but some of them are random for the judge locus.

When partners are more likely to be your clonemates, that is high p , it is easier to reach the stability of altruism since your partners share the same genes (Figures 3.4).

When there are more cues in the population, it is more accurate to distinguish kins from non-kins. So high diversity of recognition cues favors the evolutionary dynamics of altruism (Figures 3.5).

Plots of Shannon's diversity index for cue and judge alleles. The main advantage of that is that you will be able to quickly screen results of many

simulations to see if there are some conditions that maintain diversity (Figures 3.6, 3.7, 3.8 and 3.9).

$$H = -\sum_{i=1}^n f_i \ln f_i, H \in [0, \ln n]$$

Equation 3.9 Shannon's diversity index for cue and judge alleles

Some of the assumptions in the theoretical model of genetic kin recognition are: genetically unlinked cue and judge loci (recombination rate = 0.5); no selection except kin recognition; non-random mating (sexually reproduction proportionally to fitness); non-structure population (encounter each other randomly); no mutation; no genetic drift.

3.7. Conclusions

Our recognition model suggests the stability of recognition for altruism that altruism can maintain multiple recognition cues and be evolutionarily stable. I found that all the frequencies of alleles can oscillate synchronously over time and eventually converge to stable equilibrium states in the model for one production locus with multiple cue alleles, one perception locus with multiple judge alleles, (and one action locus with one act allele). The altruistic acts still happen at the equilibrium states since none of the genotypes gets fixed.

3.8. References

Axelrod, R. (1997). *The complexity of cooperation* (Princeton, New Jersey: Princeton University Press).

Axelrod, R., and Hamilton, W. D. (1981). The Evolution of Cooperation. *Science* 211, 1390-1396.

Axelrod, R., Hammond, R. A., and Grafen, A. (2004). Altruism via kin-selection strategies that rely on arbitrary tags with which they coevolve. *Evolution* 58, 1833-1838.

Barton, N. H., and Turelli, M. (1991). Natural and sexual selection on many loci. *Genetics* 127, 229-255.

Bshary, R., and Bergmueller, R. (2008). Distinguishing four fundamental approaches to the evolution of helping. *Journal of Evolutionary Biology* 21, 405-420.

Crozier, R. H. (1986). Genetic clonal recognition abilities in marine invertebrates must be maintained by selection for something else. *Evolution* 40, 1100-1101.

Crozier, R. H. (1987). *Genetic aspects of kin recognition: concepts, models, and synthesis* (New York: Wiley).

Darwin, C. (1859). *On the Origin of Species* (Cambridge, Massachusetts: Harvard University Press).

Dawkins, R. (1976). *The Selfish Gene* (Oxford: Oxford Univ. Press).

Dreber, A., Rand, D. G., Fudenberg, D., and Nowak, M. A. (2008). Winners don't punish. *Nature* 452, 348-351.

Fletcher, D. J. C., and Michener, C. D. (1987). *Kin Recognition in Animals*: John Wiley & Sons).

Frank, S. A. (1998). *Foundations of Social Evolution* (Princeton, New Jersey: Princeton University Press).

Fudenberg, D., Nowak, M. A., Taylor, C., and Imhof, L. A. (2006). Evolutionary game dynamics in finite populations with strong selection and weak mutation. *Theoretical Population Biology* 70, 352-363.

Gardner, A., and West, S. A. (2010). Greenbeards. *Evolution* 64, 25-38.

Gardner, A., West, S. A., and Barton, N. H. (2007). The relation between multilocus population genetics and social evolution theory. *Am Nat* 169, 207-226.

Gigord, L. D. B., Macnair, M. R., and Smithson, A. (2001). Negative frequency-dependent selection maintains a dramatic flower color polymorphism in the rewardless orchid *Dactylorhiza sambucina* (L.) Soo. *Proceedings of the National Academy of Sciences of the United States of America* 98, 6253-6255.

Grafen, A. (1985). A geometric view of relatedness, In *Oxford Surveys in Evolutionary Biology*, R. Darwins, and M. Ridley, eds. (Oxford University Press).

Grafen, A. (1990). Do animals really recognize kin? *Anim Behav* 39, 42–54.

Griffin, A. S., West, S. A., and Buckling, A. (2004). Cooperation and competition in pathogenic bacteria. *Nature* 430, 1024-1027.

Grosberg, R. K., and Quinn, J. F. (1989). The evolution of selective aggression conditioned on allorecognition specificity. *Evolution* 43, 504–515.

Hamilton, W. D. (1964). The genetic evolution of social behaviour, I & II. *J Theor Biol* 7, 1-52.

Hamilton, W. D. (1970). Selfish and Spiteful Behaviour in an Evolutionary Model. *Nature* 228, 1218-&.

Hamilton, W. D. (1987). *Discriminating nepotism: expectable, common, overlooked* (New York: Wiley).

Hauert, C., Michor, F., Nowak, M. A., and Doebeli, M. (2006). Synergy and discounting of cooperation in social dilemmas. *Journal of Theoretical Biology* 239, 195-202.

Jansen, V. A., and van Baalen, M. (2006). Altruism through beard chromodynamics. *Nature* 440, 663-666.

Keller, L., and Ross, K. G. (1998). Selfish genes: a green beard in the red fire ant. *Nature* 394, 573–575.

Kirkpatrick, M., Johnson, T., and Barton, N. (2002). General models of multilocus evolution. *Genetics* 161, 1727-1750.

Koeslag, J. H. (1997). Sex, the prisoner's dilemma game, and the evolutionary inevitability of cooperation. *Journal of Theoretical Biology* 189, 53-61.

Koeslag, J. H., and Terblanche, E. (2003). Evolution of cooperation: cooperation defeats defection in the cornfield model. *Journal of Theoretical Biology* 224, 399-410.

Langer, P., Nowak, M. A., and Hauert, C. (2008). Spatial invasion of cooperation. *Journal of Theoretical Biology* 250, 636-643.

Lehmann, L., and Keller, L. (2006). The evolution of cooperation and altruism - a general framework and a classification of models. *Journal of Evolutionary Biology* 19, 1365-1376.

Lieberman, E., Hauert, C., and Nowak, M. A. (2005). Evolutionary dynamics on graphs. *Nature* 433, 312-316.

Mateo, J. M. (2004). Recognition systems and biological organization: The perception component of social recognition. *Annales Zoologici Fennici* 41, 729-745.

McElreath, R., and Boyd, R. (2007). *Mathematical Models of Social Evolution* (Chicago, Illinois: University Of Chicago Press).

Nowak, M. A. (2006). *Evolutionary Dynamics: Exploring the Equations of Life* (Cambridge, Massachusetts: Belknap Press).

Nowak, M. A. (2006). Five rules for the evolution of cooperation. *Science* 314, 1560-1563.

Nowak, M. A., and Sigmund, K. (2005). Evolution of indirect reciprocity. *Nature* 437, 1291-1298.

Nowak, M. A., Tarnita, C. E., and Wilson, E. O. (2010). The evolution of eusociality. *Nature* 466, 1057-1062.

Pacheco, J. M., Traulsen, A., and Nowak, M. A. (2006). Active linking in evolutionary games. *Journal of Theoretical Biology* 243, 437-443.

Price, G. R. (1970). Selection and Covariance. *Nature* 227, 520-&.

Queller, D. C. (1985). Kinship, Reciprocity and Synergism in the Evolution of Social-Behavior. *Nature* 318, 366-367.

Queller, D. C., Ponte, E., Bozzaro, S., and Strassmann, J. E. (2003). Single-gene greenbeard effects in the social amoeba *Dictyostelium discoideum*. *Science* 299, 105-106.

Ratnieks, F. L. W. (1991). The evolution of genetic odor-cue diversity in social Hymenoptera. *Am Nat* 137, 202-226.

Ross, K. G., and Keller, L. (1998). Genetic control of social organization in an ant. *Proc Natl Acad Sci U S A* 95, 14232-14237.

Rousset, F., and Roze, D. (2007). Constraints on the origin and maintenance of genetic kin recognition. *Evolution* 61, 2320-2330.

Roze, D., and Rousset, F. (2005). Inbreeding depression and the evolution of dispersal rates: a multilocus model. *Am Nat* 166, 708-721.

Sachs, J. L., Mueller, U. G., Wilcox, T. P., and Bull, J. J. (2004). The evolution of cooperation. *Quarterly Review of Biology* 79, 135-160.

Summers, K., and Crespi, B. (2005). Cadherins in maternal-foetal interactions: red queen with a green beard? *Proc Biol Sci* 272, 643-649.

Taylor, C., Iwasa, Y., and Nowak, M. A. (2006). A symmetry of fixation times in evolutionary dynamics. *Journal of Theoretical Biology* 243, 245-251.

Taylor, C., and Nowak, M. A. (2006). Evolutionary game dynamics with non-uniform interaction rates. *Theoretical Population Biology* 69, 243-252.

Traulsen, A., Nowak, M. A., and Pacheco, J. M. (2006). Stochastic dynamics of invasion and fixation. *Physical Review E* 74, -.

Trivers, R. L. (1971). Evolution of Reciprocal Altruism. *Quarterly Review of Biology* 46, 35-&.

West, S. A., Griffin, A. S., and Gardner, A. (2007). Evolutionary explanations for cooperation. *Curr Biol* 17, R661-672.

West, S. A., Griffin, A. S., and Gardner, A. (2007). Social semantics: altruism, cooperation, mutualism, strong reciprocity and group selection. *Journal of Evolutionary Biology* 20, 415-432.

Wright, S. (1922). Coefficients of inbreeding and relationship. *American Naturalist* 56, 330-338.

Wynne-Edwards, V. C. (1962). *Animal Dispersion in Relation to Social Behavior* (London: Oliver & Boyd).

3.9. Table legends

Index	A_i	C_j	J_k
1	1	1	1
2	1	1	2
3	1	2	1
4	1	2	2
5	2	1	1
6	2	1	2
7	2	2	1
8	2	2	2

Table 3.1 Alleles Indexes at Three Loci

Payoff	1	2	3	4	5	6	7	8
1	b-c	-c	0	0	-c	-c	0	0
2	b	0	0	0	0	0	0	0
3	0	0	0	b	0	0	0	0
4	0	0	-c	b-c	0	0	-c	-c
5	b	0	0	0	0	0	0	0
6	b	0	0	0	0	0	0	0
7	0	0	0	b	0	0	0	0
8	0	0	0	b	0	0	0	0

Table 3.2 Payoff matrix of altruism

Locus	Genetic relatedness, r	Actor's cost, C	Recipient's benefit, B
<i>Cue</i>	$r(Cue) = 1$	c	b
<i>Judge</i>	$r(Judge) = \frac{p}{p + (1 - p)f_{C_i}}$	c	b
<i>Action</i>	$r(Act) = \frac{p}{p + (1 - p)(f_{C_1}^2 f_{J_1} + f_{C_2}^2 f_{J_2}) / (f_{C_1} f_{J_1} + f_{C_2} f_{J_2})}$	c	b

Table 3.3 Genetic Relatedness at Each of Three loci

3.10. Figure legends

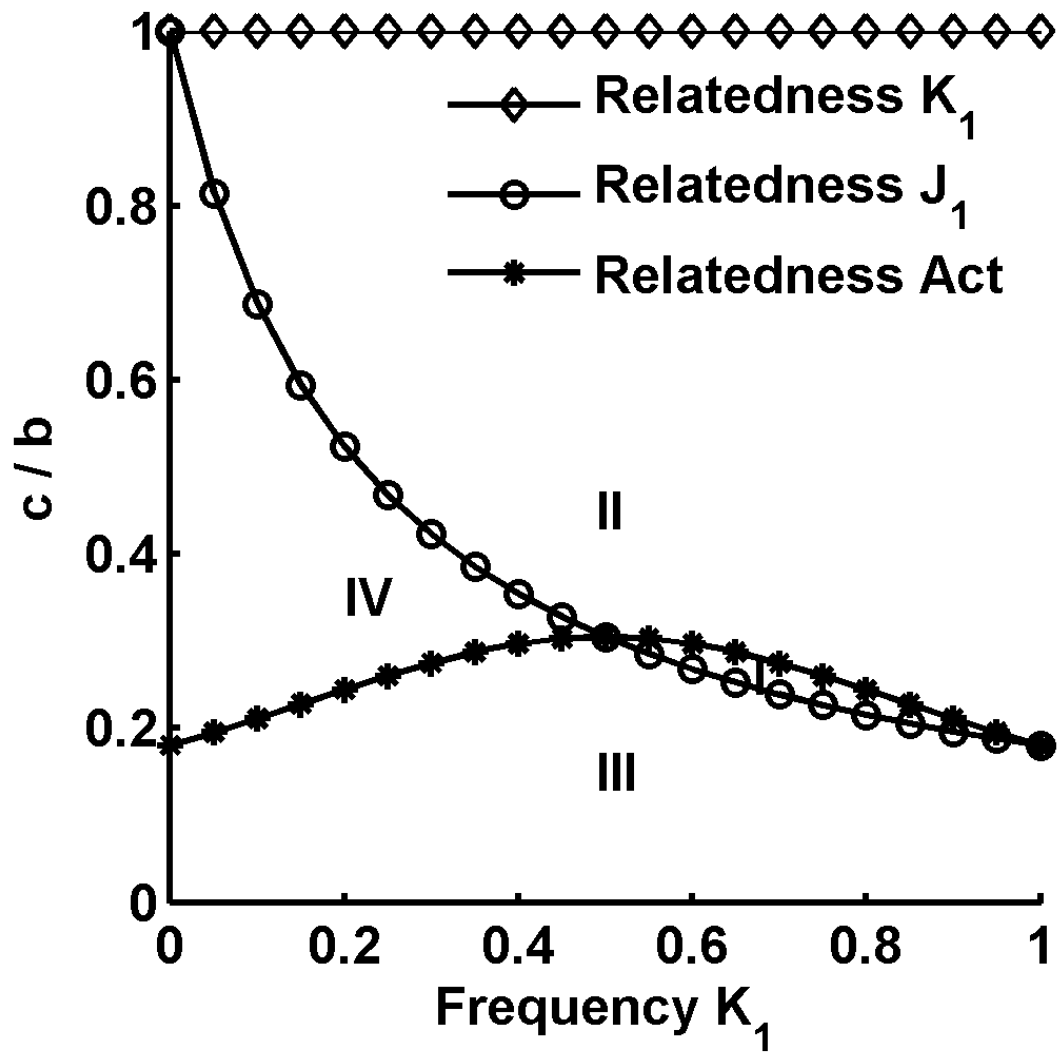


Figure 3.1 Genetic Relatedness at Each of Three loci

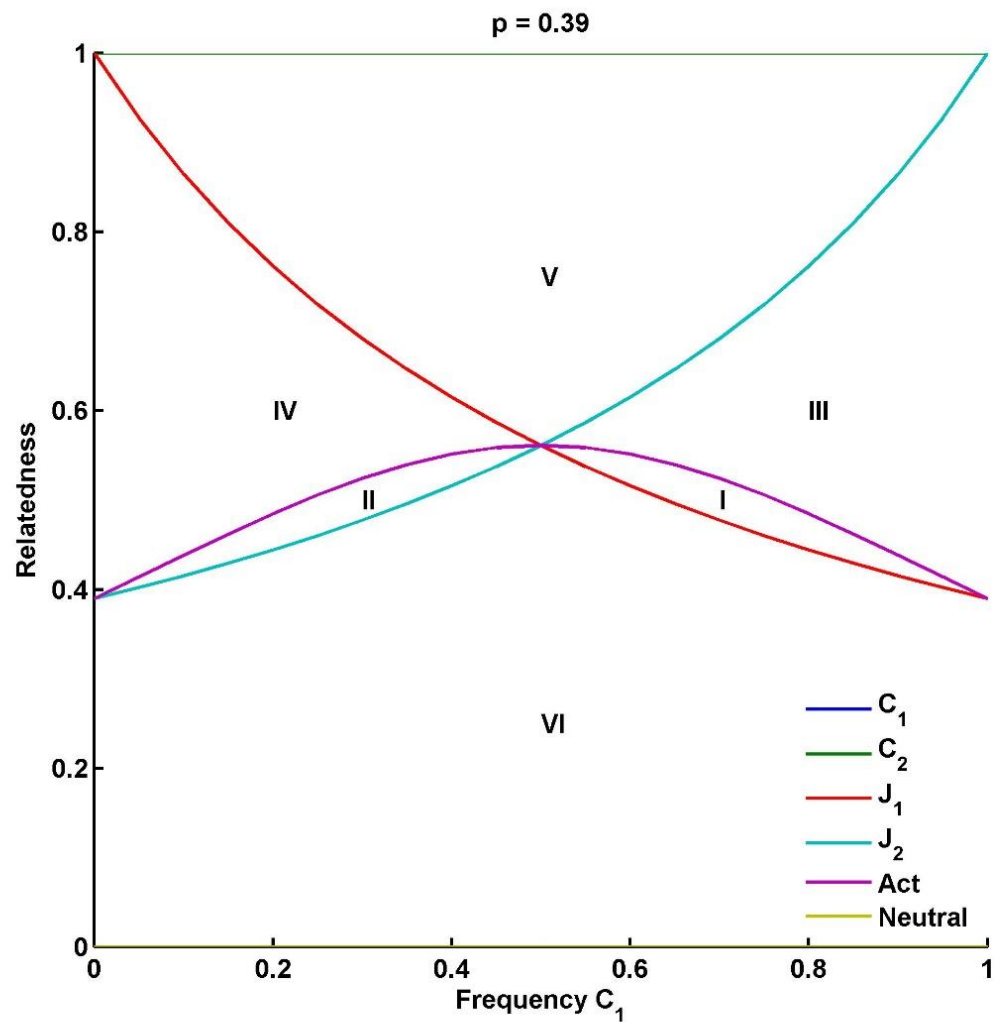


Figure 3.2 Genetic Relatedness for all alleles at three loci

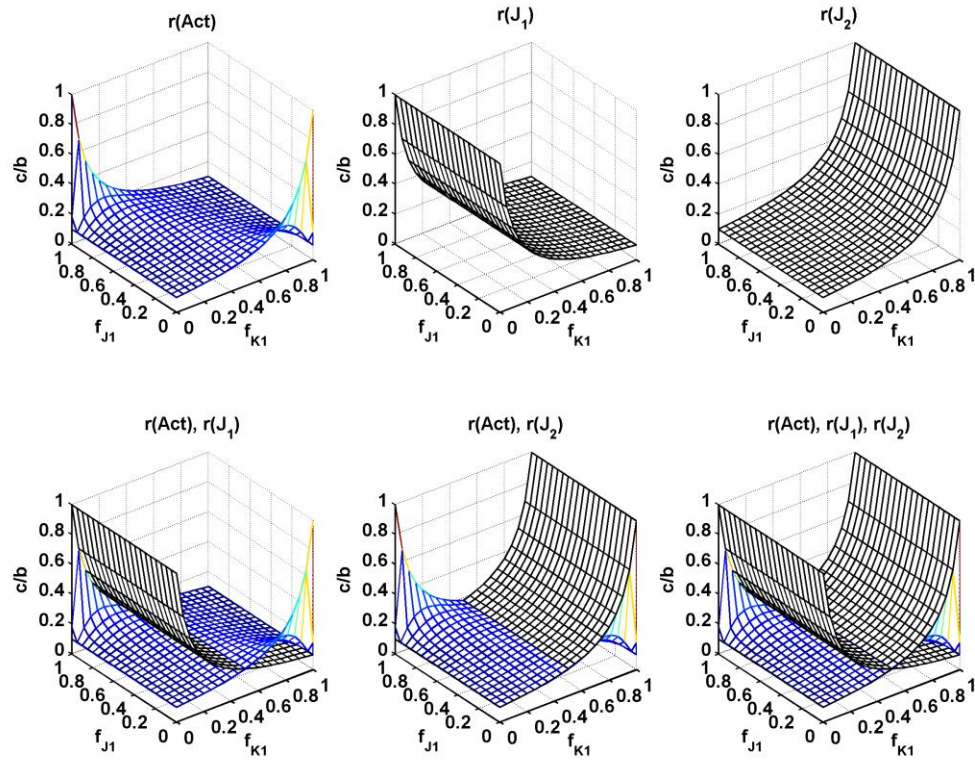


Figure 3.3 Prediction of Stability of Evolution of Altruism and Eusociality

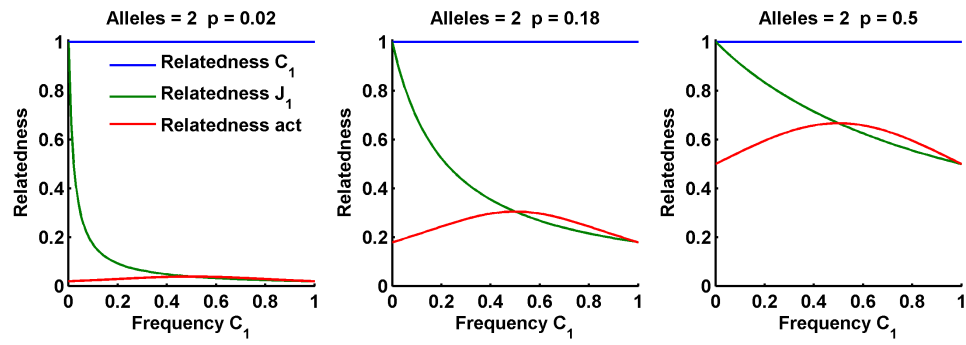


Figure 3.4 Genetic Relatedness with Two Cue Alleles

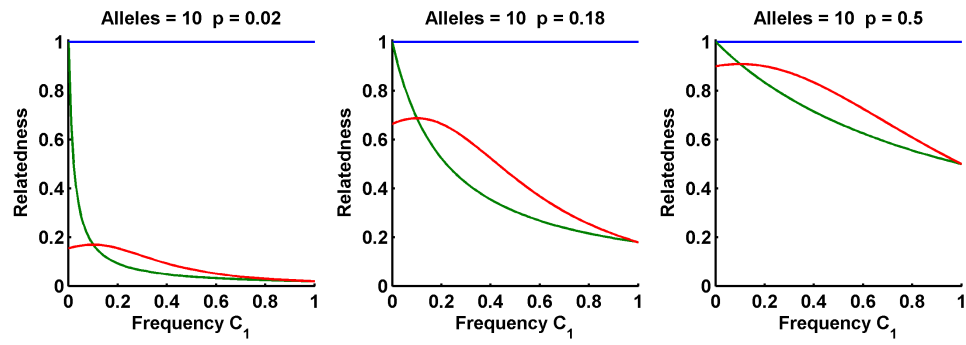


Figure 3.5 Genetic Relatedness with Ten Cue Alleles

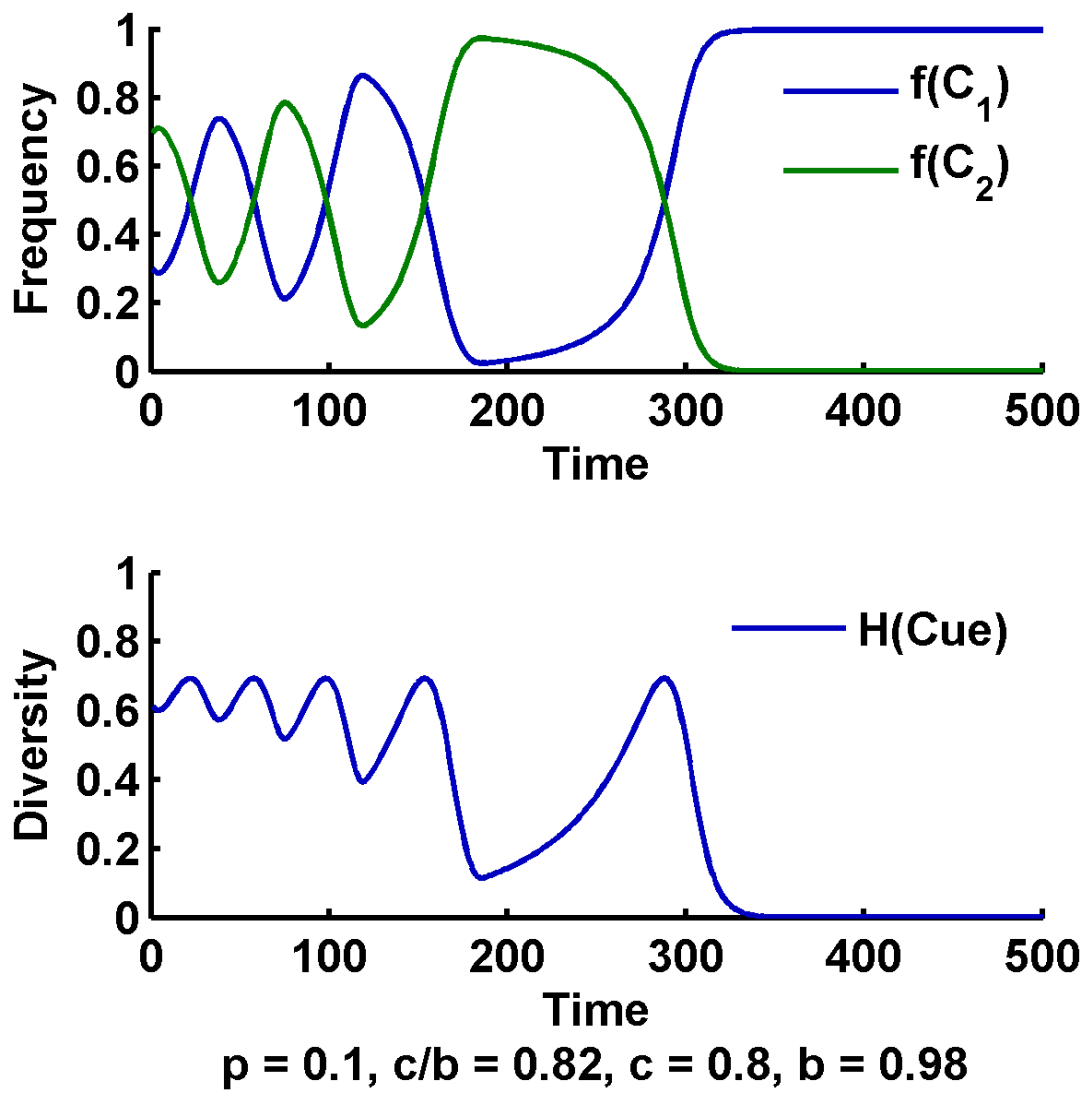


Figure 3.6 Plots of Shannon's diversity index for two cue and judge alleles ($c/b=0.82$).

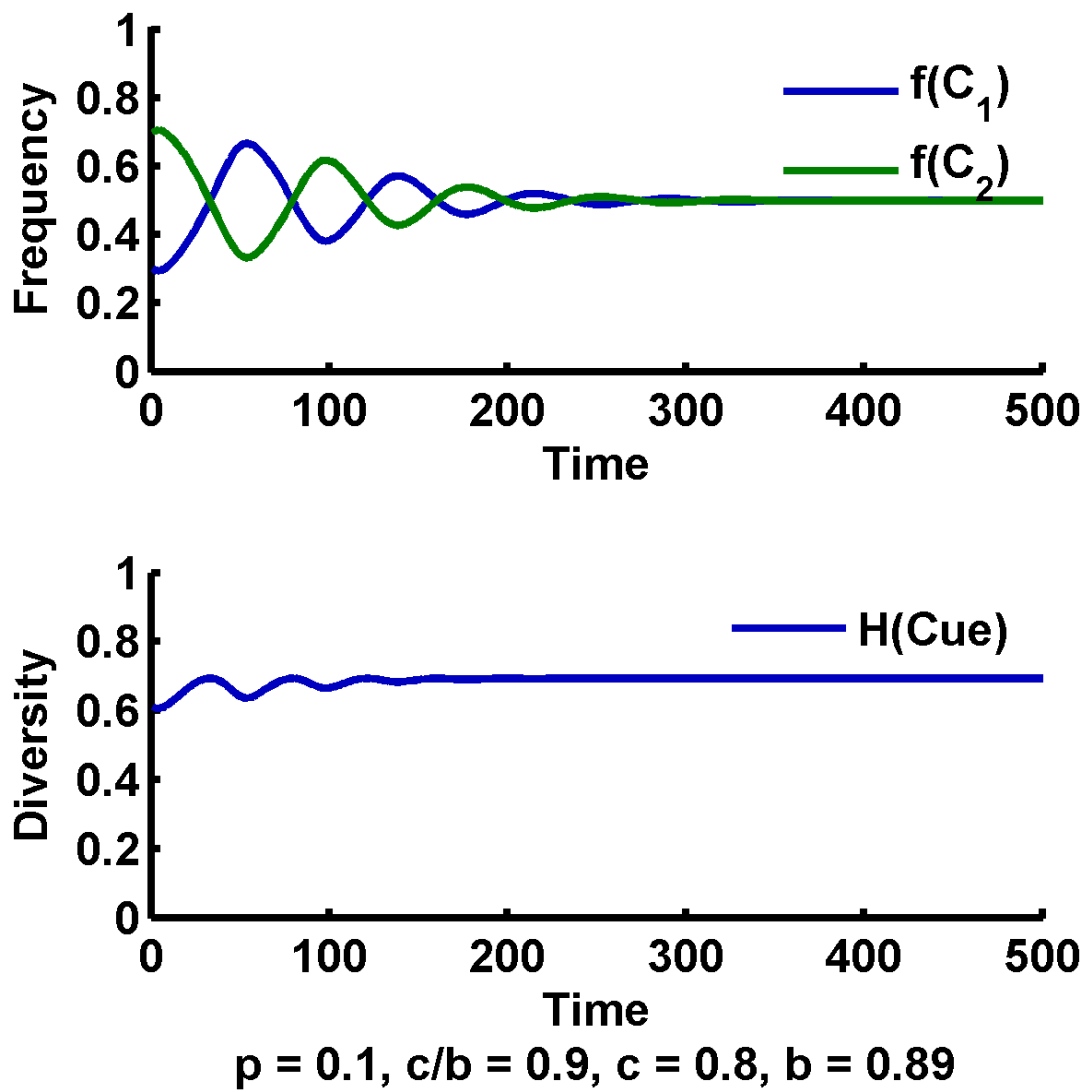


Figure 3.7 Plots of Shannon's diversity index for two cue and judge alleles ($c/b=0.9$).

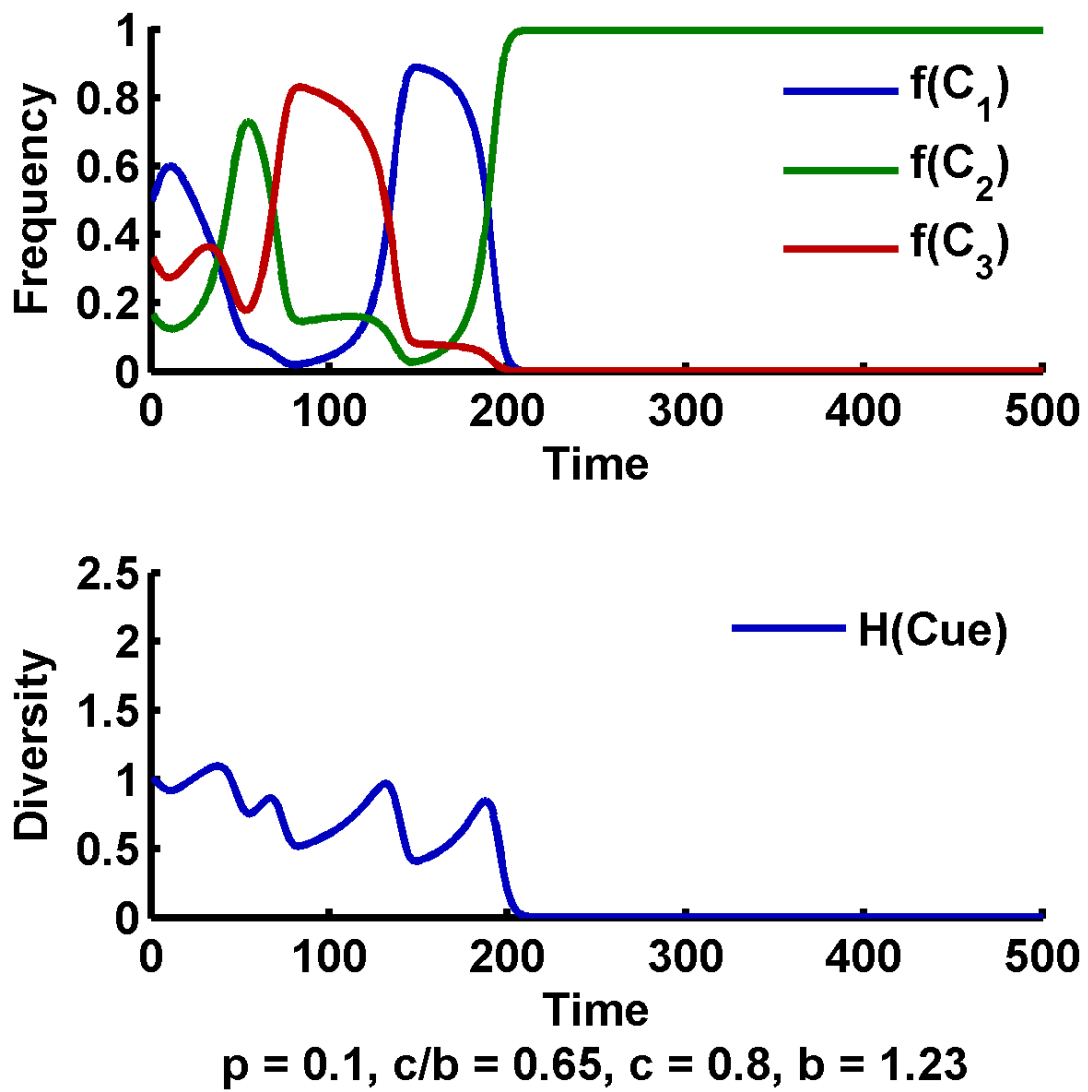


Figure 3.8 Plots of Shannon's diversity index for three cue and judge alleles ($c/b = 0.65$).

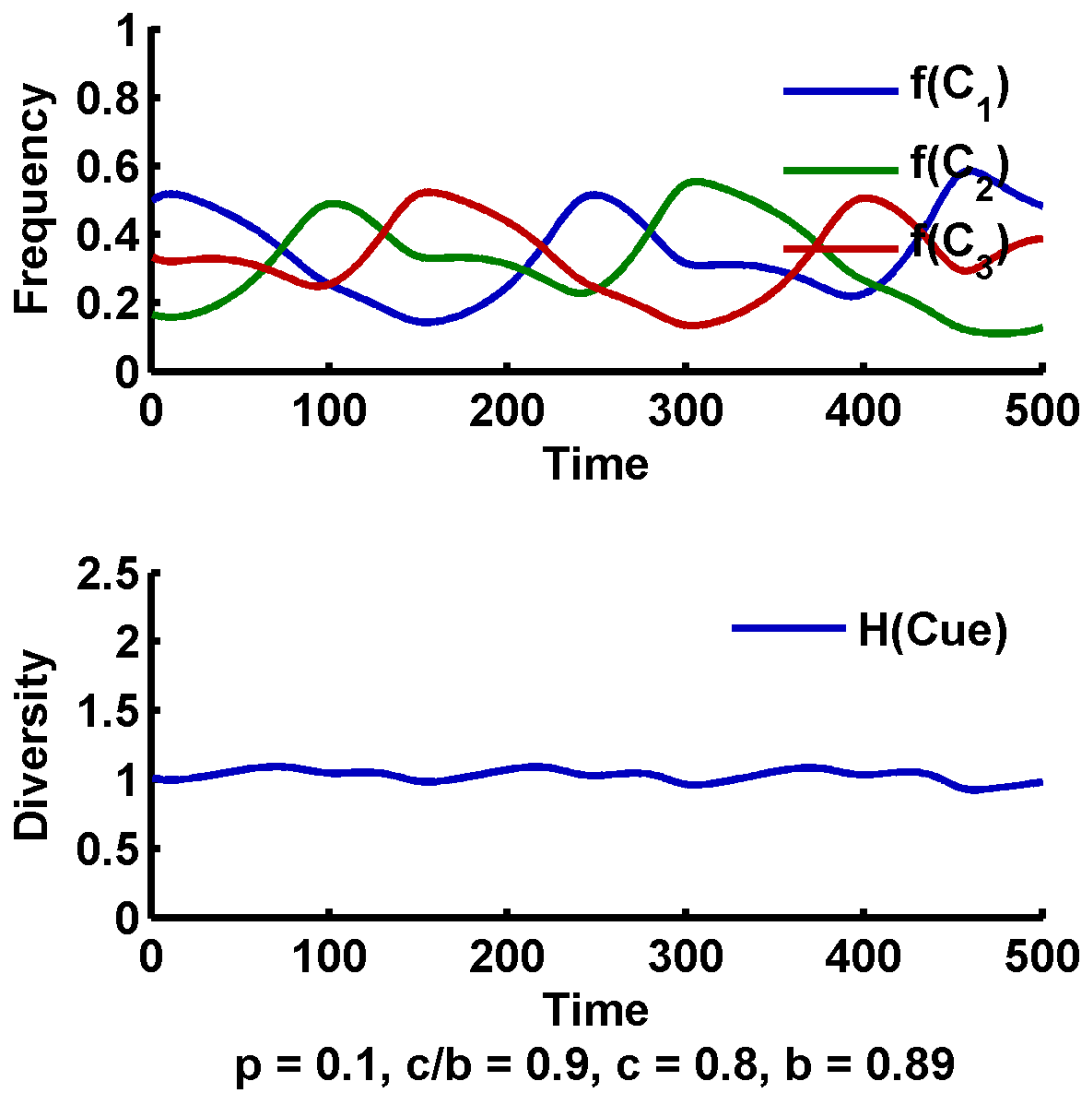


Figure 3.9 Plots of Shannon's diversity index for three cue and judge alleles ($c/b = 0.9$).

Exploring Fitness Cost/Benefit to Solve the Crozier's Paradox

4.1. Abstract

Previous models show that rare cue alleles get eliminated by selection and a common allele gets fixed, which ruins the altruism system. So it is unclear how genetic recognition for altruism persists. Here, I designed a novel model with three types of genetic components, production, perception and action. I tested whether interactions between perception and production loci could evolve to reject common cue alleles, allowing multiple cue alleles persist. I found that all the frequencies of the genotypes, alleles and altruistic acts can oscillate over time and eventually converge to stable equilibrium states.

4.2. Introduction

In 1964 W. D. Hamilton produced an elegant formal theory that provided a potential solution to this problem (Hamilton 1964). Hamilton argues that altruistic

acts to relatives can be favored by natural selection, because relatives share the same gene as helpers. Hamilton expanded the definition of fitness in terms of inclusive fitness which is the sum of a direct benefit through producing offspring and an indirect benefit through aiding genetic relatives. Hamilton made these two components additive by devaluing each offspring or relative by the genetic relatedness to them.

From this Hamilton predicted that altruism will be favored by natural selection when the inequality $rB - C > 0$ is satisfied, where B is the benefit of the act of altruism to the recipient, C the cost of the act to the actor and r the genetic relatedness between the actor and the recipient (Hamilton 1964). Inclusive fitness is applicable not only to helping but also to any behavior (West et al. 2007b). This inequality has now come to be known as Hamilton's rule. Hamilton's theory is also frequently referred to as the inclusive fitness theory or kin selection.

However, testing Hamilton's rule without measuring the cost and benefit of eusociality is an inadequate test. Focusing only on relatedness and neglecting the cost and benefit terms usually takes the form of assuming implicitly that $B = C$. In my thesis I plan to design a novel model of genetic kin recognition and to explore the range of fitness costs and benefits with the model to explain how eusociality could evolve and persist.

The genetic cues of the production component are shown to be greenbeard genes. There are alleles that, in effect, recognize copies of themselves in others,

regardless of relatedness at other loci. The greenbeard nature of these alleles is responsible for what is known as Crozier's paradox, the observation that selection favors common cue alleles and thereby removes the variation that is required for discrimination (Crozier RH 1986, Crozier RH 1987, Crozier RH & Pamilo P 1996). Altruistic greenbeard alleles are outlaw genes because, by causing altruism towards others who are not relatives, they act against the interest of other genes in the genome. This can lead to intragenomic conflict, with other genes being selected to eliminate the extra altruism, if they can do so without also eliminating themselves as targets of altruism.

We reconceptualize the components of kin recognition in terms intragenomic conflict. The genetic cues of the production component are shown to be greenbeard genes. There are alleles that, in effect, recognize copies of themselves in others, regardless of relatedness at other loci. The greenbeard nature of these alleles is responsible for what is known as Crozier's paradox, the observation that selection favors common cue alleles and thereby removes the variation that is required for discrimination. Altruistic greenbeard alleles are outlaw genes because, by causing altruism towards others who are not relatives, they act against the interest of other genes in the genome. This can lead to intragenomic conflict, with other genes being selected to eliminate the extra altruism, if they can do so without also eliminating themselves as targets of altruism.

In recognition systems this is easily accomplished by genes in the perception component; they will be selected to ignore high-frequency cue alleles in their decisions to give altruism. This in turn can decrease or reverse selection for common alleles, solving Crozier's paradox. In this view, greenbeard genes are far from rare, but are at the foundation of all genetic cue mechanisms. Their inappropriate levels of altruism need to be tamed by loci in the perception or action components before such recognition systems can be effective in implementing inclusive fitness.

In the models below, I consider selection on a haploid organism, with cue loci and perception loci that are unlinked. For simplicity disequilibria generated by selection are ignored (though they will not be ignored in the computer models). Each individual is faced with the choice of helping a partner. The partner will be a relative related by r a fraction p of the time; the remaining $1-p$ of the time the partner will be unrelated. Altruism is determined by matching at the cue locus and evaluation at the perception locus. When a match at the cue locus occurs, and when the perception locus perceives and passes on the match, the individual loses $-c$ units of fitness but gives its partner b units of fitness.

Imagine a recognition system that causes altruism whenever the actor and its partner have the same alleles at the matching locus. In the background of the perception and action genes generating altruism by this kind of rule, the effect at the matching locus is a simple greenbeard effect: it causes altruism to occur regardless

of whether the partner is a relative who shares the allele identical by descent or a random individual who shares the allele identical by state. The objection might be raised that the matching allele does not mechanistically cause the altruism, because other genes are also required for that. But this objection would apply to all genes for complex traits. A gene for long tails does not create a long tail by itself; many other genes are required. But we say it is a gene for long tails because (on average) it makes the difference between a longer and a shorter tail. Our matching locus in a greenbeard gene in this sense –it makes the difference between altruism and no altruism and it does so in a way that pays no attention to kinship.

Here, I design a novel model with three types of genetic components, production, perception and action. I test whether interactions between perception and production loci could evolve to reject common cue alleles, allowing multiple cue alleles persist. I found that all the frequencies of the genotypes, alleles and altruistic acts can oscillate over time and eventually converge to stable equilibrium states.

Our recognition model suggests the stability of recognition for altruism that altruism can maintain multiple recognition cues and be evolutionarily stable. I found that all the frequencies of alleles can oscillate synchronously over time and eventually converge to stable equilibrium states in the model for one production locus with multiple cue alleles, one perception locus with multiple judge alleles, and one action locus with one act allele. The altruistic acts still happen at the

equilibrium states since none of the genotypes gets fixed. And the frequency of altruistic acts remains constant since the genotype frequencies remain constant.

Furthermore, both rare *Cue* and *Judge* alleles can invade into the system though they can't be stabilized. Since the invasion of rare *Cue* alleles and the stabilization of recognition are not under the same conditions, it is still unclear how rare cues could be maintained by altruism.

4.3. Recognition Bases on Multiple Cues

Altruism is a common behavior in nature which benefits others at a cost to one own fitness. Hamilton's rule tells us if genetic relatedness r in a population equals to 1, such as in clonemates, altruism can evolve when fitness benefit b is larger than cost c . The kin recognition mechanism argues that according to a complex system, which in general consists of three genetic components. Production, perception, and action, individuals can recognize their relatives based on their production cues, and favor each other. Since relatives are likely to share same genes, genes of those who perform altruism could be inherited through copies in their relatives' offspring. That's how altruism can evolve. The recognition for greenbeard mechanism is simple. It's only based on a single gene. Those who share a same greenbeard gene can discriminate and help each other. In both mechanisms, recognition is based on multiple cues. If multiple cues persist in a population, recognition could be stable.

Recognition is unstable. However, previous models eventually lead to a fixed cue. In greenbeard, those who have common greenbeard alleles, such as the red one in this diagram, are more likely to favor each other than those of rare ones, such as the blue one. And thereby common greenbeard alleles spread. If there are no other forces to check this spread, a common allele will eventually fix in a population. In the diagram, the red fixes and the blue goes extinct. In kin recognition, it leads to the same consequence. A common production cue fixes. When there is only one cue left, all individuals are same and can't be discriminated. So recognition in previous models is unstable. Recognition could be stable only when multiple cues, rather than a fixed cue, are maintained in populations by altruism. So, how could multiple cues be maintained?

Stable recognition model. To answer this important question, I designed a novel model in which recognition for altruism is based on two genetic components. production and perception. The *Cue* locus in the production component has two alleles. C_1 and C_2 . The *Judge* locus in the perception component also has two alleles. J_1 and J_2 . The two loci are unlinked. Our hypothesis is that common *Cue* alleles could be disfavored and rare ones favored by negative feedbacks from *Judge* alleles, and thereby multiple cues could be maintained. How do negative feedbacks occur in the model?

4.4. Simulation of Pure Greenbeard

To test whether our model is stable or not, we do computational simulations in two steps. We first simulate in a pure greenbeard system without kin recognition; and then simulate in a combined system of greenbeard and kin recognition. In simulation of pure greenbeard, the population consists of haploid adults. A genetically random partner is present to each individual. Offspring's population is sexually reproduced, proportionally to parents' fitness after altruism. In the ancestor, the four allele frequencies are initialized by these four parameters.

4.5. Negative Feedbacks can Maintain Multiple Cue with Intermediate Frequencies

We test our model with a constant population size 3,000. In the figures 4.3, 4.4 and 4.5, the X axis is generation, and the Y axis are frequencies of genotype and allele. As we see, when all alleles start intermediate frequencies and the cost and cost-to-benefit ratio are high, all allele frequencies oscillate cross generations. Let us plot Judge allele frequencies versus Cue allele frequencies. For example, in the figures at the bottom, J_1 against C_1 . If frequency curve reaches the X axis, that means, C_1 goes extinct and C_2 gets fixed. If curve reaches the top border, C_1 fixes. If curve reaches the Y axis, that means, J_1 goes extinct and J_2 gets fixed. If curve reaches the right border, J_1 fixes. If curve doesn't reach any axis or border, that means, multiple Cue and Judge alleles persist. These three figures show that after starting

at the black point, allele frequency curves cycle in a certain central space cross generations. It implies that multiple alleles can persist, and none of alleles goes extinct or gets fixed. These simulations suggest that negative feedbacks can maintain multiple alleles with intermediate frequencies, and therefore our recognition model can be stable.

But negative feedbacks can't always maintain intermediate frequencies. Simulations at low cost and cost-to-benefit ratio show that one of Cue and Judge alleles eventually gets fixed and the other goes extinct (Figures 4.1 and 4.2).

4.6. Stable Recognition Model under High Fitness Cost/Benefit and High Cost

Now, an interesting question is in which space of cost and benefit negative feedbacks can maintain multiple alleles. I did simulations with different costs and cost-to-benefit ratios. In these plots, from left to right, c goes higher; from bottom to up, c/b becomes higher (Figures 4.6 and 4.7). These plots clearly show that multiple intermediate alleles can persist under high c/b and high c . It suggests that our recognition system is stable with intermediate allele frequencies under high c/b and high c . The greenbeard Cue alleles are favored when $c < b$, that is $c/b < 1$. When c/b is low, they are favored too strongly, so a Cue allele fixes before negative feedback can control it. When c/b is high, negative feedback has a chance to bring common

Cue allele frequencies down and raise rare Cue allele frequencies up. When $c = b$, Cue alleles are not favored.

The Judge alleles are disfavored when $c > 0$ (Figure 4.7). When c is high, they are disfavored strongly to oppose the greenbeard Cue alleles. When c is low, they are disfavored weakly to provide enough feedbacks. So there is strong selection on the Judge locus and weak selection on the Cue locus.

4.7. The Recognition is Unstable When Cue Alleles Start Rare

When Cue alleles start rare, for example, C_0 starts at twenty percent, simulations show that rare cue alleles sometimes can invade into the system, but the recognition system is unstable (Figure 4.8). Our model is sometimes stable with rare *Judge* alleles. When Judge alleles starts rare, for example, J_0 starts at one percent, simulations show that our recognition model is sometimes stable.

Combined greenbeard and kin recognition. Now we test our model in a recombined system of greenbeard and kin recognition. The population changes into pairs of clonemates. In the previous pure greenbeard, Judge alleles can't be favored. Now, they can be favored in kin recognition by helping clonemate because they share same Judge alleles. Cue alleles can still be favored by greenbeard effects. The probability that partner is clonemate is Pr , and the probability that partner is random is $1 - Pr$. When $Pr = 0$, that is, all partners are random, the system stays as a pure greenbeard system.

Combined system of greenbeard and kin recognition. As Pr goes higher, such as 0.5, that means, the probabilities of clonemate partner and random partner are equal (Figure 4.9). In such combined system of greenbeard and kin recognition, the recognition is sometimes stable. The recognition is unstable in pure kin recognition system. When all partners are clonemates, that is $Pr = 1$, the system switches to a pure kin recognition system. The recognition in such system is always unstable.

We found the recognition can be stable under low h , that is when the cue and judge loci tend to link together (Figure 4.10 and 4.11).

4.8. Conclusions

The data also show that the altruism is still performed during the equilibrium status. The time to reach equilibrium status is critical for the evolution of genetic kin recognition. It evolves quickly to reach the equilibrium status if there is strong selection, and evolves slowly if there exists weak selection.

According to Crozier's paradox, common alleles will eventually get fixed in the population since they get much more benefits from help of relatives compared to rare alleles. Rare alleles will eventually go extinct in the population since they receive much fewer benefits. To maintain multiple alleles, there must be some kinds of negative feedbacks to bring down common alleles and raise up rare alleles. Most of previous models only focus on the cue alleles and try to figure out the mechanism of kin recognition system which is incomprehensive. It is impossible that a single

gene perform the complex duties both recognizing relatives and exercising altruistic behaviors toward them. Adding another gene responsible for acting is still not enough which leads to Crozier's paradox.

Here we show that models with three components (recognition, detection, and action) show highlight on the mechanism of evolution of kin recognition.

We analyzed the results from simulations and tracked down to the conditions which require the altruistic behavior to evolve. So the kin recognition can be favored by the dynamics among three components. Even though the rare cue alleles invade into the system, intermediate cues always can be maintained by the system. This is the first time that a model show equilibrium status of evolution of genetic kin recognition. The model is useful to the field which highlights the possibility of mechanism of evolution of altruism. It is unclear how the diversity of cues is maintained in the system, how common cues are brought down and rare cues invade. Here our model shows that it is the negative feedback from other genetic components to balance the evolution of cues. There are strong selections on detective components and weak selections. It is suggested that the models are showing the conditions under which genetic kin recognition system can evolve over time.

So far, simulation results suggest that our recognition model for altruism can maintain two cues with intermediate frequencies in a population, and thereby the

recognition can be stable. The model is unstable in some conditions, especially with relatives. It's not clear how rare cues invade.

4.9. References

Axelrod, R. (1997). *The complexity of cooperation* (Princeton, New Jersey: Princeton University Press).

Axelrod, R., and Hamilton, W. D. (1981). The Evolution of Cooperation. *Science* 211, 1390-1396.

Axelrod, R., Hammond, R. A., and Grafen, A. (2004). Altruism via kin-selection strategies that rely on arbitrary tags with which they coevolve. *Evolution* 58, 1833-1838.

Barton, N. H., and Turelli, M. (1991). Natural and sexual selection on many loci. *Genetics* 127, 229-255.

Bshary, R., and Bergmueller, R. (2008). Distinguishing four fundamental approaches to the evolution of helping. *Journal of Evolutionary Biology* 21, 405-420.

Crozier, R. H. (1986). Genetic clonal recognition abilities in marine invertebrates must be maintained by selection for something else. *Evolution* 40, 1100-1101.

Crozier, R. H. (1987). Genetic aspects of kin recognition: concepts, models, and synthesis (New York: Wiley).

Darwin, C. (1859). On the Origin of Species (Cambridge, Massachusetts: Harvard University Press).

Dawkins, R. (1976). The Selfish Gene (Oxford: Oxford Univ. Press).

Dreber, A., Rand, D. G., Fudenberg, D., and Nowak, M. A. (2008). Winners don't punish. *Nature* 452, 348-351.

Fletcher, D. J. C., and Michener, C. D. (1987). Kin Recognition in Animals: John Wiley & Sons).

Frank, S. A. (1998). Foundations of Social Evolution (Princeton, New Jersey: Princeton University Press).

Fudenberg, D., Nowak, M. A., Taylor, C., and Imhof, L. A. (2006). Evolutionary game dynamics in finite populations with strong selection and weak mutation. *Theoretical Population Biology* 70, 352-363.

Gardner, A., and West, S. A. (2010). Greenbeards. *Evolution* 64, 25-38.

Gardner, A., West, S. A., and Barton, N. H. (2007). The relation between multilocus population genetics and social evolution theory. *Am Nat* 169, 207-226.

Gigord, L. D. B., Macnair, M. R., and Smithson, A. (2001). Negative frequency-dependent selection maintains a dramatic flower color polymorphism in the

rewardless orchid *Dactylorhiza sambucina* (L.) Soo. *Proceedings of the National Academy of Sciences of the United States of America* 98, 6253-6255.

Grafen, A. (1985). A geometric view of relatedness, In *Oxford Surveys in Evolutionary Biology*, R. Darwins, and M. Ridley, eds. (Oxford University Press).

Grafen, A. (1990). Do animals really recognize kin? *Anim Behav* 39, 42–54.

Griffin, A. S., West, S. A., and Buckling, A. (2004). Cooperation and competition in pathogenic bacteria. *Nature* 430, 1024-1027.

Grosberg, R. K., and Quinn, J. F. (1989). The evolution of selective aggression conditioned on allorecognition specificity. *Evolution* 43, 504–515.

Hamilton, W. D. (1964). The genetic evolution of social behaviour, I & II. *J Theor Biol* 7, 1-52.

Hamilton, W. D. (1970). Selfish and Spiteful Behaviour in an Evolutionary Model. *Nature* 228, 1218-&.

Hamilton, W. D. (1987). *Discriminating nepotism: expectable, common, overlooked* (New York: Wiley).

Hauert, C., Michor, F., Nowak, M. A., and Doebeli, M. (2006). Synergy and discounting of cooperation in social dilemmas. *Journal of Theoretical Biology* 239, 195-202.

Jansen, V. A., and van Baalen, M. (2006). Altruism through beard chromodynamics. *Nature* 440, 663-666.

Keller, L., and Ross, K. G. (1998). Selfish genes: a green beard in the red fire ant. *Nature* 394, 573-575.

Kirkpatrick, M., Johnson, T., and Barton, N. (2002). General models of multilocus evolution. *Genetics* 161, 1727-1750.

Koeslag, J. H. (1997). Sex, the prisoner's dilemma game, and the evolutionary inevitability of cooperation. *Journal of Theoretical Biology* 189, 53-61.

Koeslag, J. H., and Terblanche, E. (2003). Evolution of cooperation: cooperation defeats defection in the cornfield model. *Journal of Theoretical Biology* 224, 399-410.

Langer, P., Nowak, M. A., and Hauert, C. (2008). Spatial invasion of cooperation. *Journal of Theoretical Biology* 250, 636-643.

Lehmann, L., and Keller, L. (2006). The evolution of cooperation and altruism - a general framework and a classification of models. *Journal of Evolutionary Biology* 19, 1365-1376.

Lieberman, E., Hauert, C., and Nowak, M. A. (2005). Evolutionary dynamics on graphs. *Nature* 433, 312-316.

Mateo, J. M. (2004). Recognition systems and biological organization: The perception component of social recognition. *Annales Zoologici Fennici* 41, 729-745.

McElreath, R., and Boyd, R. (2007). *Mathematical Models of Social Evolution* (Chicago, Illinois: University Of Chicago Press).

Nowak, M. A. (2006). *Evolutionary Dynamics: Exploring the Equations of Life* (Cambridge, Massachusetts: Belknap Press).

Nowak, M. A. (2006). Five rules for the evolution of cooperation. *Science* 314, 1560-1563.

Nowak, M. A., and Sigmund, K. (2005). Evolution of indirect reciprocity. *Nature* 437, 1291-1298.

Nowak, M. A., Tarnita, C. E., and Wilson, E. O. (2010). The evolution of eusociality. *Nature* 466, 1057-1062.

Pacheco, J. M., Traulsen, A., and Nowak, M. A. (2006). Active linking in evolutionary games. *Journal of Theoretical Biology* 243, 437-443.

Price, G. R. (1970). Selection and Covariance. *Nature* 227, 520-&.

Queller, D. C. (1985). Kinship, Reciprocity and Synergism in the Evolution of Social-Behavior. *Nature* 318, 366-367.

Queller, D. C., Ponte, E., Bozzaro, S., and Strassmann, J. E. (2003). Single-gene greenbeard effects in the social amoeba *Dictyostelium discoideum*. *Science* 299, 105-106.

Ratnieks, F. L. W. (1991). The evolution of genetic odor-cue diversity in social Hymenoptera. *Am Nat* 137, 202-226.

Ross, K. G., and Keller, L. (1998). Genetic control of social organization in an ant. *Proc Natl Acad Sci U S A* 95, 14232-14237.

Rousset, F., and Roze, D. (2007). Constraints on the origin and maintenance of genetic kin recognition. *Evolution* 61, 2320-2330.

Roze, D., and Rousset, F. (2005). Inbreeding depression and the evolution of dispersal rates: a multilocus model. *Am Nat* 166, 708-721.

Sachs, J. L., Mueller, U. G., Wilcox, T. P., and Bull, J. J. (2004). The evolution of cooperation. *Quarterly Review of Biology* 79, 135-160.

Summers, K., and Crespi, B. (2005). Cadherins in maternal-foetal interactions: red queen with a green beard? *Proc Biol Sci* 272, 643-649.

Taylor, C., Iwasa, Y., and Nowak, M. A. (2006). A symmetry of fixation times in evolutionary dynamics. *Journal of Theoretical Biology* 243, 245-251.

Taylor, C., and Nowak, M. A. (2006). Evolutionary game dynamics with non-uniform interaction rates. *Theoretical Population Biology* 69, 243-252.

Traulsen, A., Nowak, M. A., and Pacheco, J. M. (2006). Stochastic dynamics of invasion and fixation. *Physical Review E* 74, -.

Trivers, R. L. (1971). Evolution of Reciprocal Altruism. *Quarterly Review of Biology* 46, 35-&.

West, S. A., Griffin, A. S., and Gardner, A. (2007). Evolutionary explanations for cooperation. *Curr Biol* 17, R661-672.

West, S. A., Griffin, A. S., and Gardner, A. (2007). Social semantics: altruism, cooperation, mutualism, strong reciprocity and group selection. *Journal of Evolutionary Biology* 20, 415-432.

Wright, S. (1922). Coefficients of inbreeding and relationship. *American Naturalist* 56, 330-338.

Wynne-Edwards, V. C. (1962). *Animal Dispersion in Relation to Social Behavior* (London: Oliver & Boyd).

4.10. Figure legends

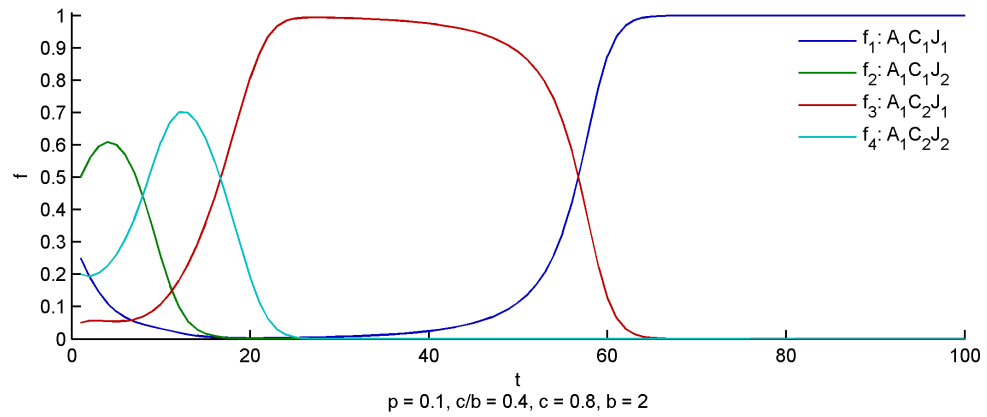


Figure 4.1 Evolutionary dynamics of Genotype Frequencies in Unstable Kin Recognition

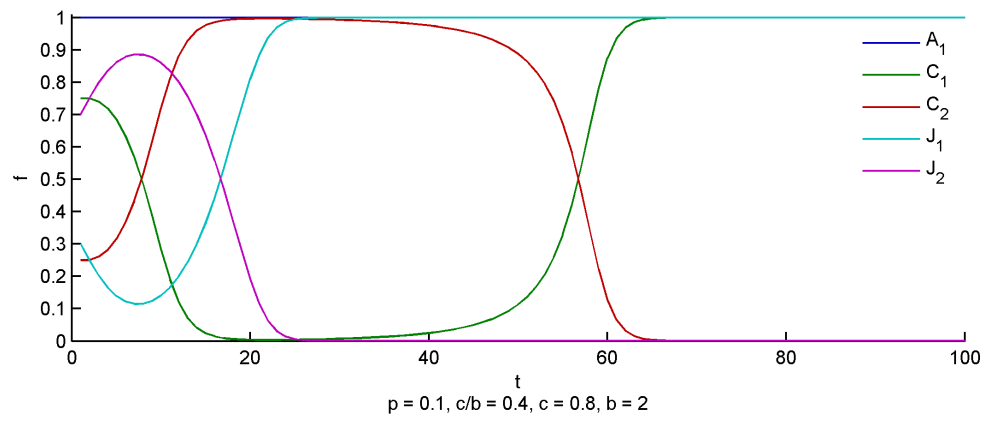


Figure 4.2 Evolutionary dynamics of Allele Frequencies in Unstable Kin Recognition

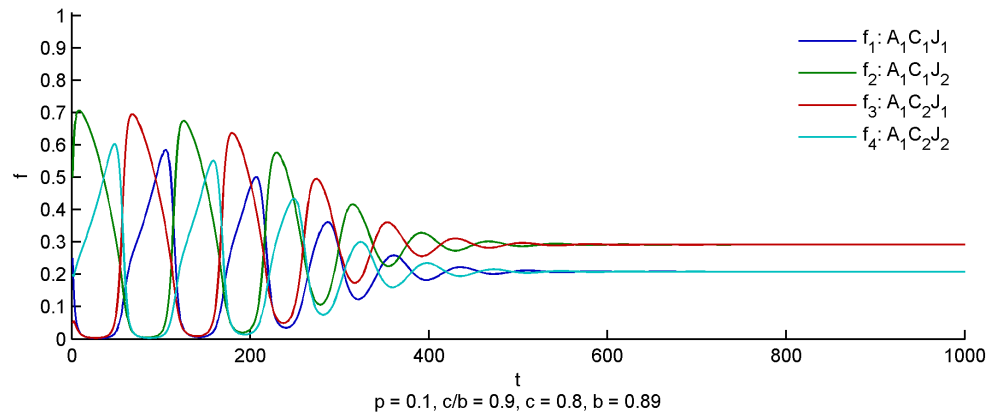


Figure 4.3 Evolutionary dynamics of Genotype Frequencies in Stable Kin Recognition

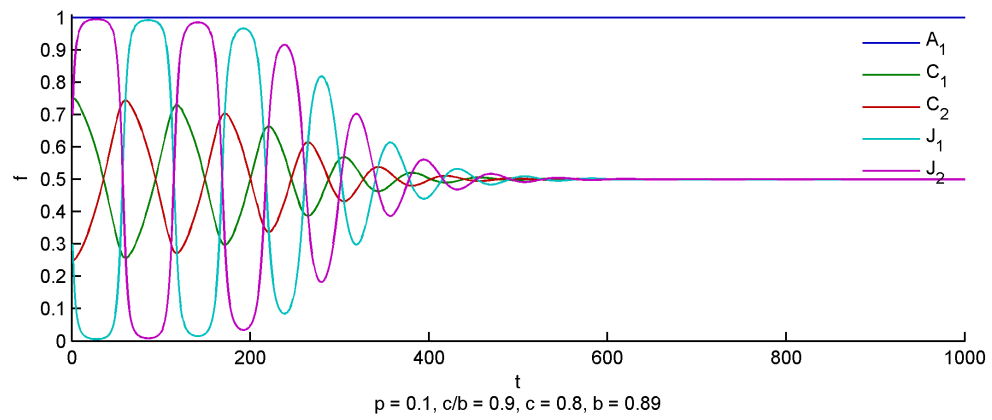


Figure 4.4 Evolutionary dynamics of Allele Frequencies in Stable Kin Recognition

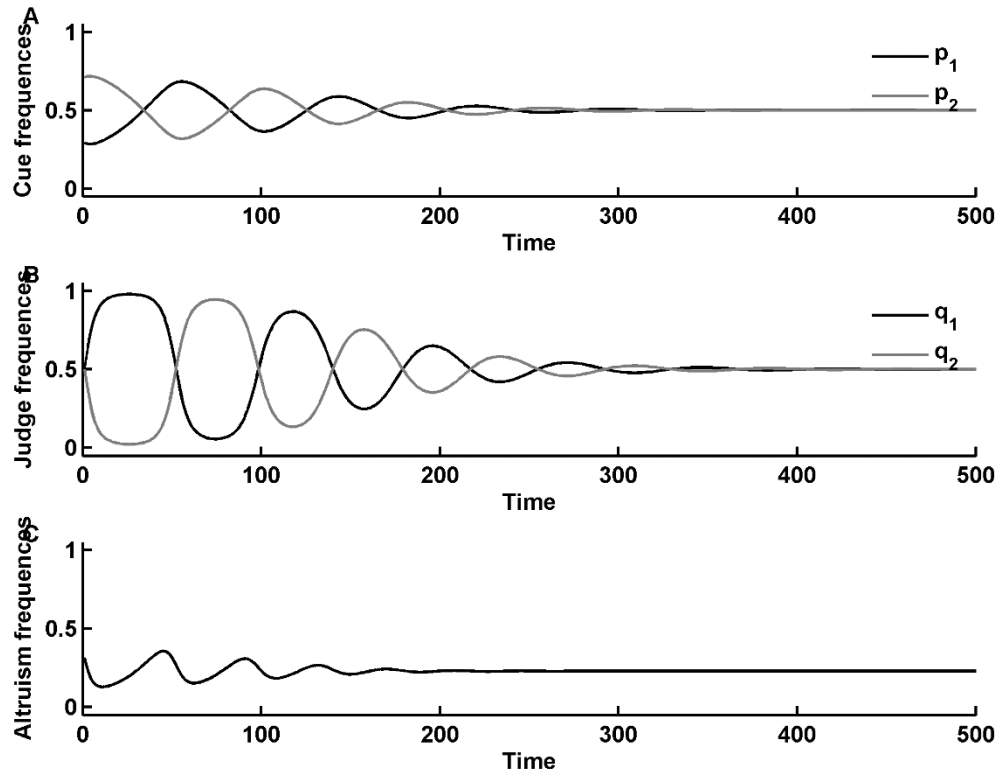


Figure 4.5 The stability of recognition.

(A) Change of frequencies of cue alleles. (B) Change of frequencies of judge alleles. (C) Change of frequencies of altruistic behavior. per individual. $c = 0.8$, $c/b = 0.9$, $p = 0.1$, $h = 0.5$, $p_1 = 0.29$, $p_2 = 0.71$, $q_1 = q_2 = 0.5$, $T = 500$. Blank dot, start of simulation; solid dot, end of simulation.

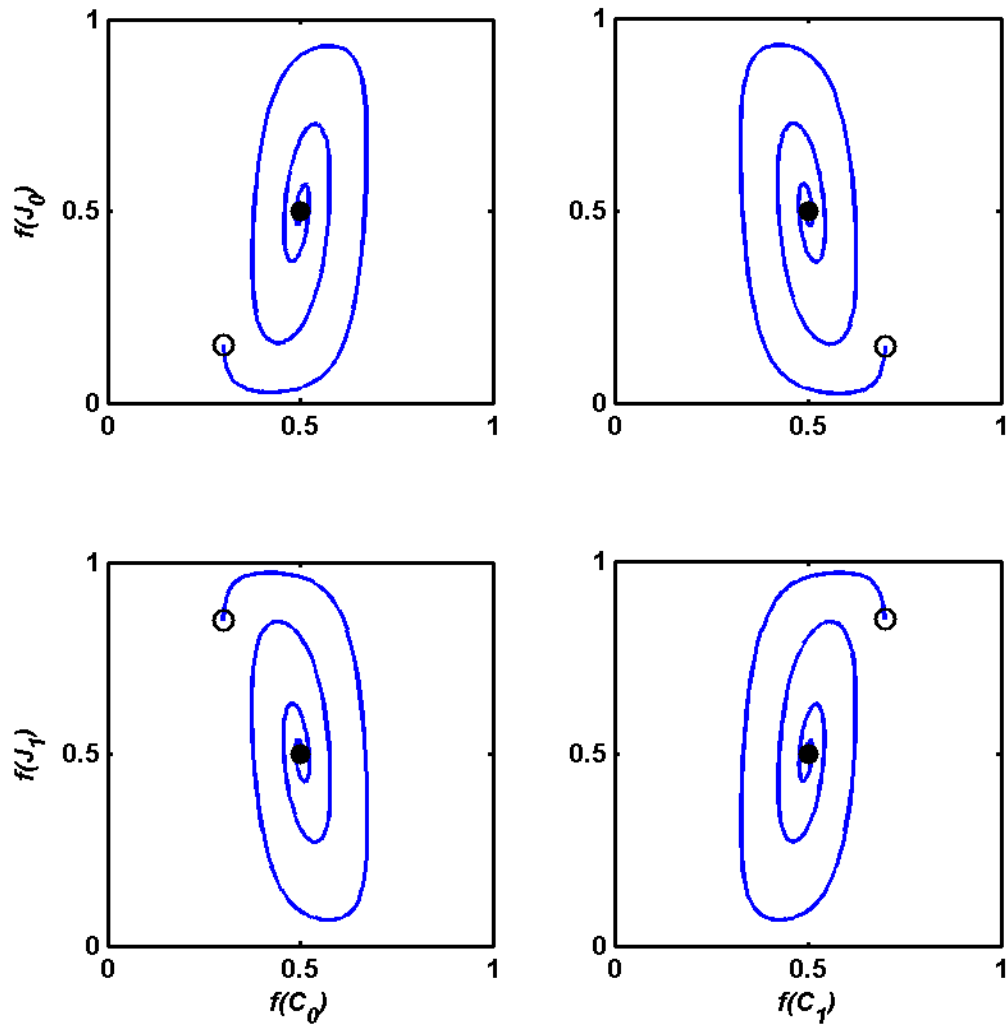


Figure 4.6 The stability of recognition in space of frequencies.
 Blank dot, start of simulation; solid dot, end of simulation. $c = 0.8$, $c/b = 0.9$, $p = 0.1$, $h = 0.5$, $p_1 = 0.29$, $p_2 = 0.71$, $q_1 = q_2 = 0.5$, $T = 500$.

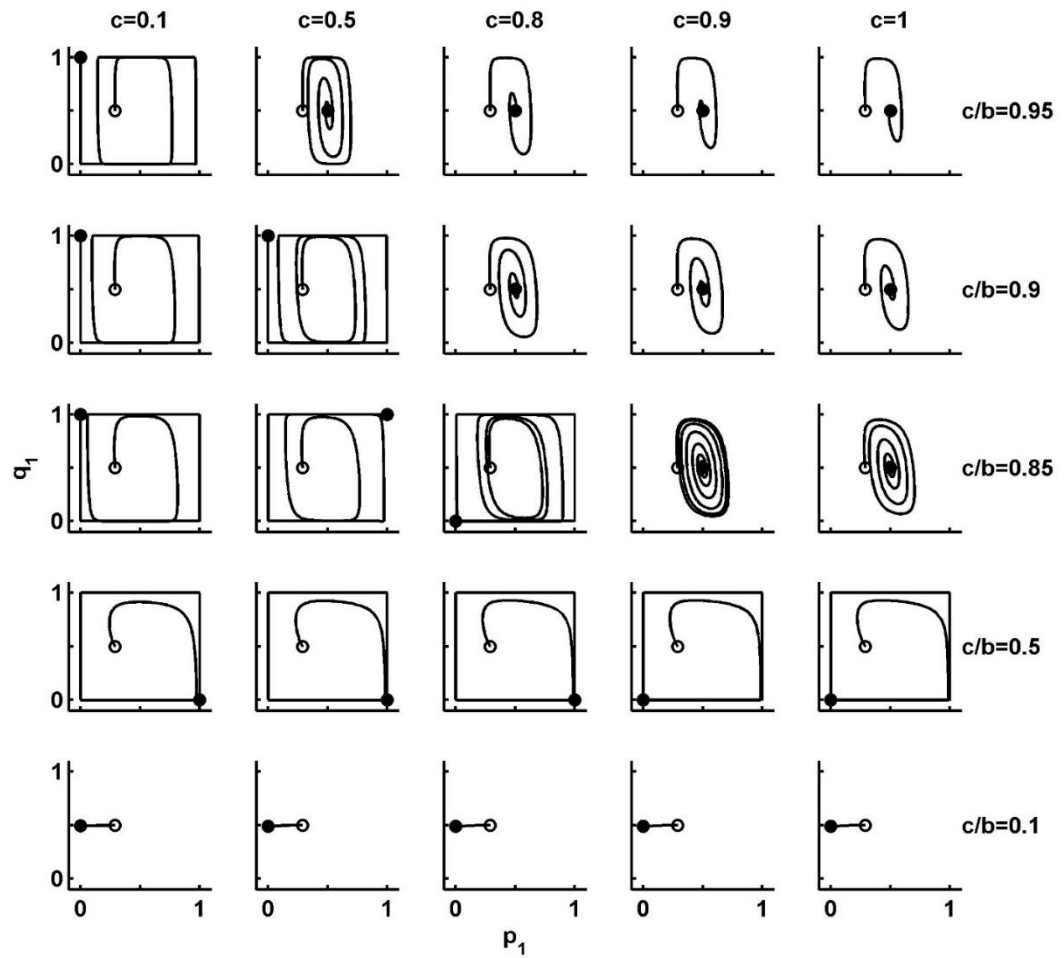


Figure 4.7 Limited regions of stability: high c/b and high c .
**The space of frequency spaces under various initialized allele. $h = 0.5$, $p = 0.1$,
 $p_1 = 0.29$, $p_2 = 0.71$, $q_1 = q_2 = 0.5$, $T = 10,000$. Blank dot, start of simulation;
solid dot, end of simulation.**

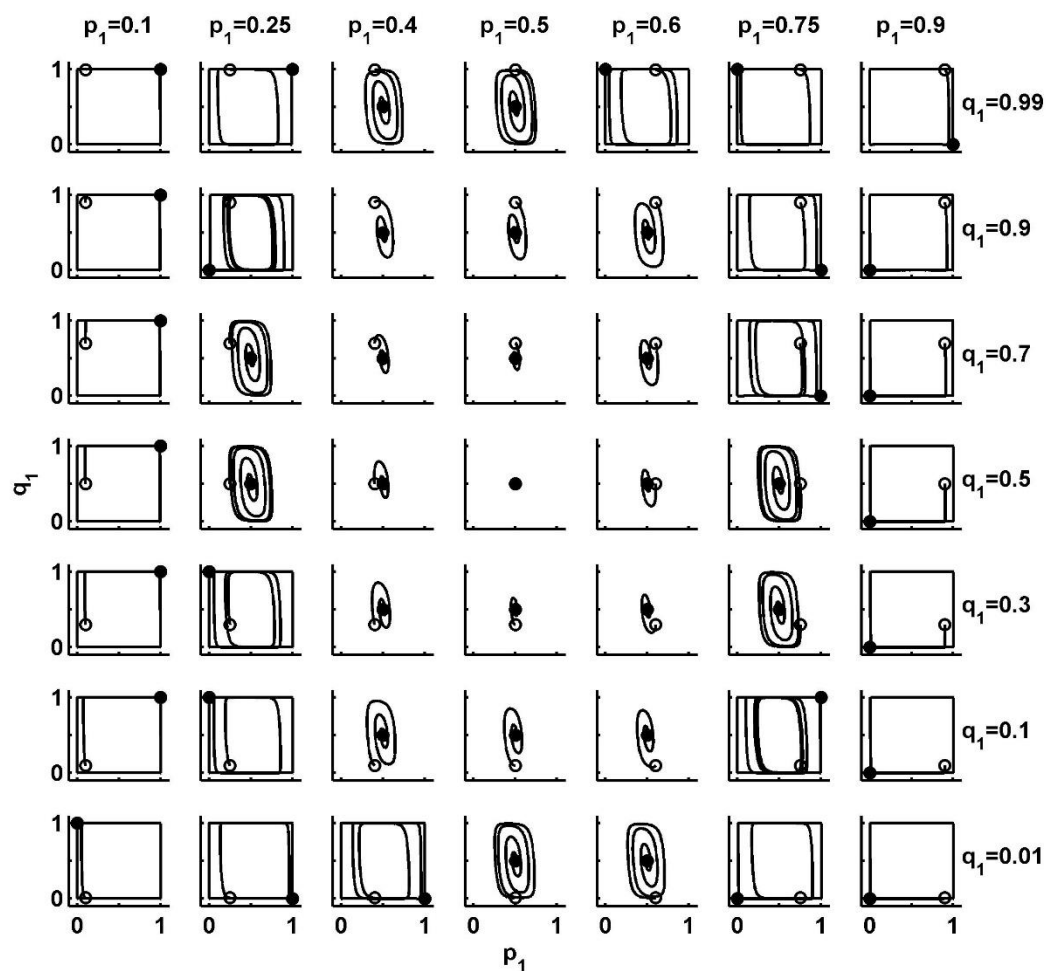


Figure 4.8 Limited regions of stability: intermediate frequencies.
The space of frequency spaces under various costs and cost-to-benefit ratios.
 $h = 0.5$, $p = 0.1$, $p_1 = 0.29$, $p_2 = 0.71$, $q_1 = q_2 = 0.5$, $T = 10,000$. Blank dot, start of simulation; solid dot, end of simulation.

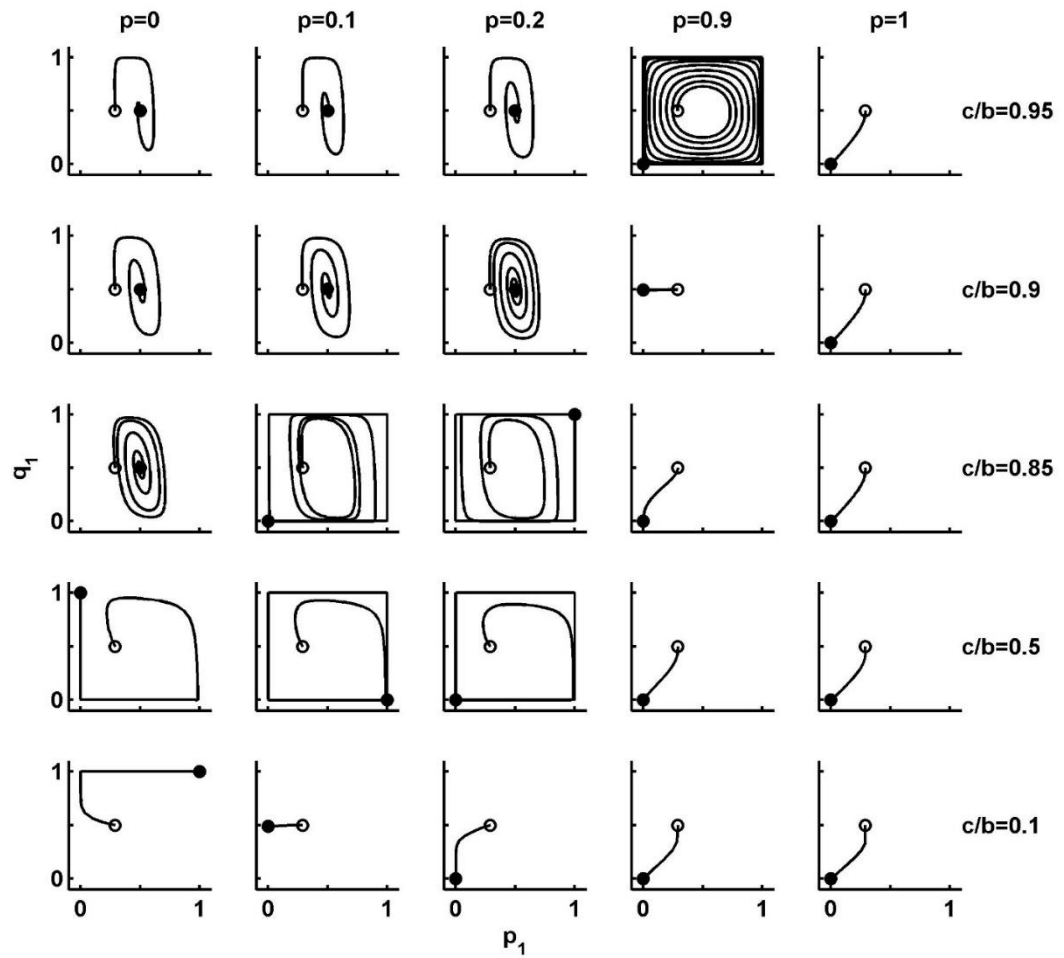


Figure 4.7 Limited regions of stability: low p .

The space of frequency spaces under various probabilities of clonemate partner and cost-to-benefit ratios. $c = 0.8$, $h = 0.5$, $p_1 = 0.29$, $p_2 = 0.71$, $q_1 = q_2 = 0.5$, $T = 10,000$. Blank dot, start of simulation; solid dot, end of simulation.

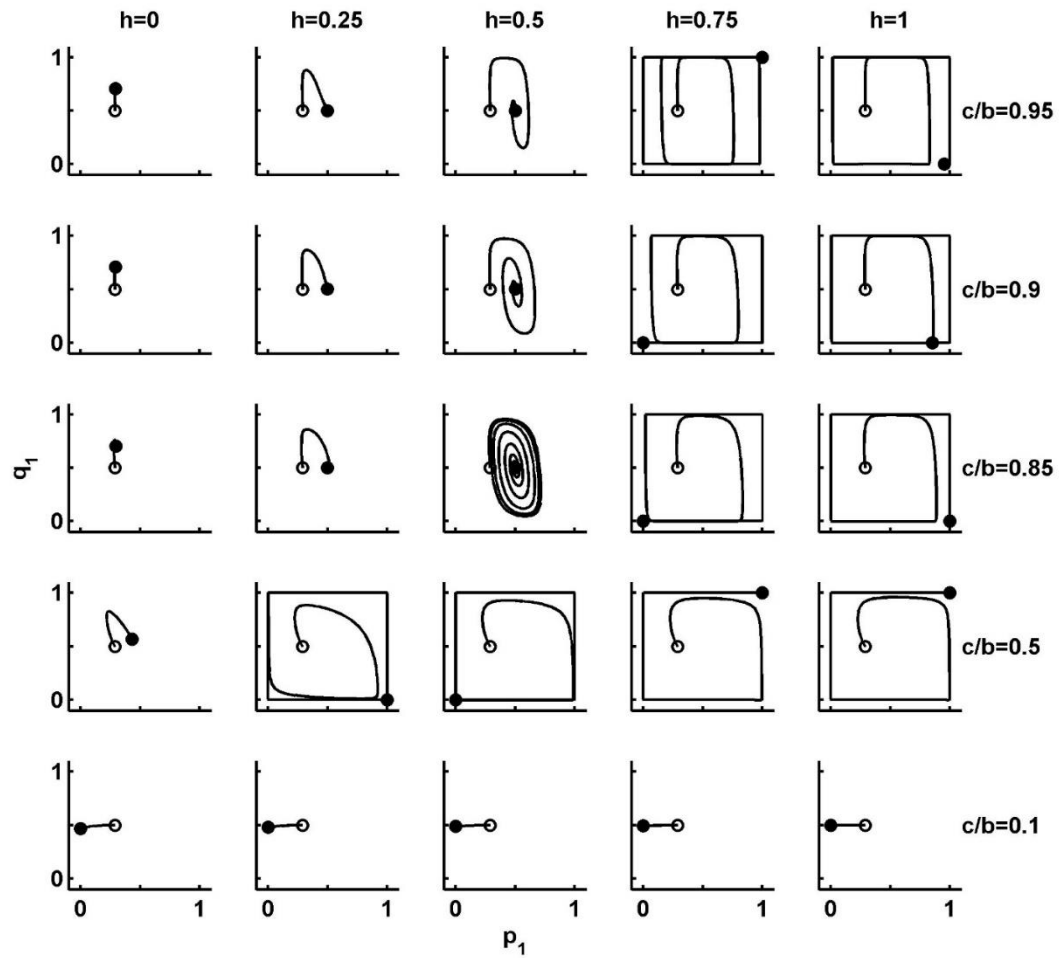


Figure 4.8 The space of frequency spaces under various recombination rates and cost-to-benefit ratios.

$c = 0.8$, $p = 0.1$, $p_1 = 0.29$, $p_2 = 0.71$, $q_1 = q_2 = 0.5$, $T = 10,000$. Blank dot, start of simulation; solid dot, end of simulation.

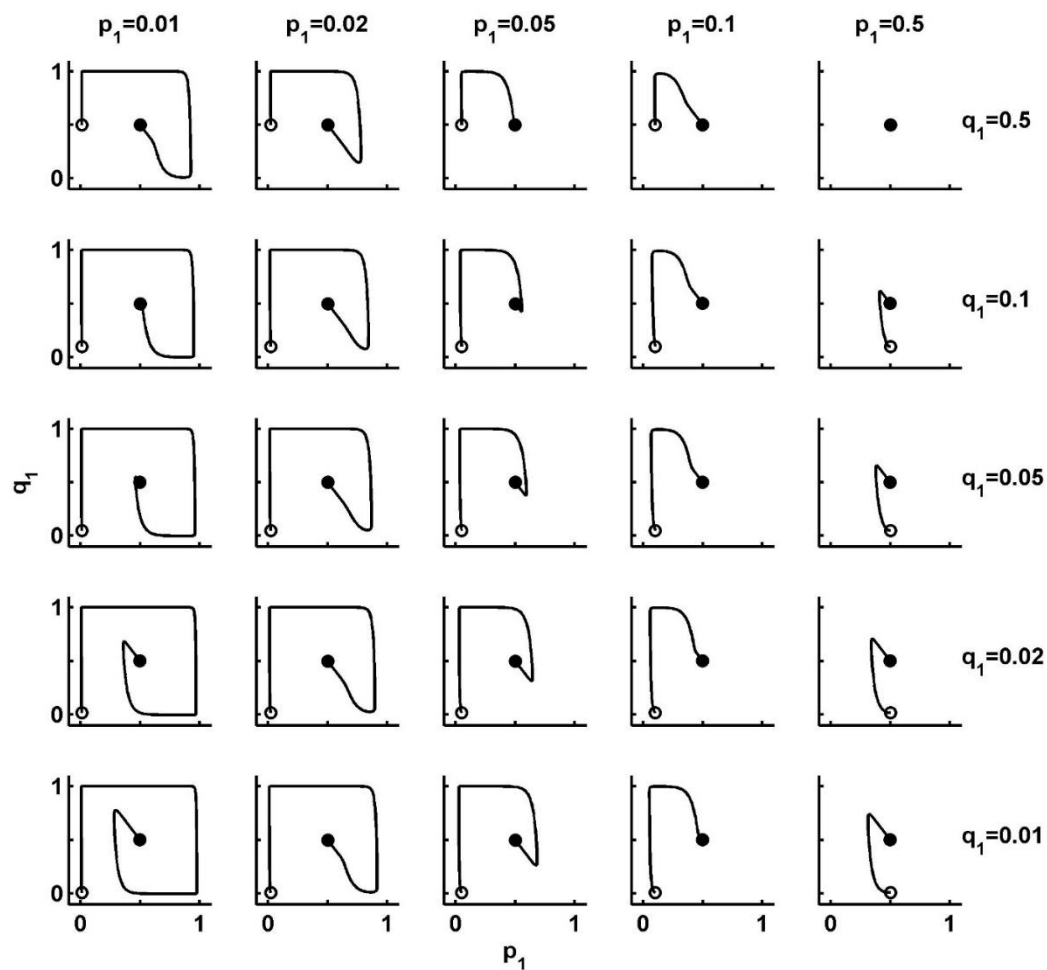


Figure 4.9 The space of frequency spaces under various initialized allele with highly linked loci.

$c = 0.8$, $c/b = 0.9$, $p = 0.1$, $h = 0.1$, $T = 10,000$. Blank dot, start of simulation; solid dot, end of simulation.

Applications of New Perspective of Hamilton's rule: $r > C/B$

5.1. Abstract

Altruism is a behavior that benefits others at a cost to one's own ability of survival and/or reproduction, which is one of the paradoxes in Darwin's theory of evolution. Altruistic behaviors are commonly performed in eusocial animals, such as nearly all hymenoptera (including bees, wasps, and ants), termites, ambrosia beetles, and so on. Inclusive fitness theory predicts that altruistic behavior can evolve when sufficient fitness benefits are given to relatives. A different modeling approach has led to a challenge to this theory. The modelers claim that relatedness is not causal, that eusocial behavior is very hard to evolve requiring more workers before the queen increased fitness, and that there is no conflict involved. I showed that, even within the terms of this modeling framework, inclusive fitness thinking leads to insights that completely change these conclusions. I showed that relatedness is causal, that eusociality does evolve more readily being favored under

a lower benefits threshold. I concluded that multiple modeling approaches are useful and that efforts to synthesize them are better than asserting that one is universally better than the other. Moreover, either greenbeard effects or genetic kin recognition requires genetic polymorphisms as cues on which recognition is based. Previous models show that rare cue alleles get eliminated by selection and a common allele gets fixed, which ruins the altruism system. So it is unclear how genetic recognition for altruism persists. I designed a novel model with two types of genetic components, production and perception. I analyzed my recognition model theoretically toward a cost/benefit analysis of fitness and genetic relatedness. I predicted the stability of recognition for altruism based on my model. Furthermore I tested my recognition model through various computational and biological simulations. My simulation results consistently show altruism can maintain multiple recognition cues and be evolutionarily stable. I concluded that cost/benefit of fitness and genetic relatedness play a critical role in the evolution of altruism and eusociality, and therefore can maintain the stability of recognition for altruism.

5.2. Introduction

Altruistic behaviors benefit others at a cost to one's own ability of survival and/or reproduction, which is one of the paradoxes in Darwin's theory of evolution. Altruistic behaviors are commonly performed in eusocial animals, such as nearly all hymenoptera (including bees, wasps, and ants), termites, ambrosia beetles, and so on.

5.3. The genetic relatedness and evolution of eusociality

The controversy over the Nowak et al paper has mostly been conducted at rather abstract levels; different researchers favor different modeling strategies and interpret the evidence differently. I take a different and more concrete approach by investigating their model for the evolution of eusociality more deeply. I have therefore followed the recommendation of Nowak et al. for modeling social evolution, and in particular eusociality, using deterministic evolutionary dynamics described by ordinary differential equations. However, stimulated by inclusive fitness thinking, I have sought to understand apparent differences between their results compared to previous models. In every case, I find that their rejection of accepted results is premature, and that in fact the insights known from inclusive fitness theory also emerge using their method.

5.4. The general model for genetic kin recognition

Inclusive fitness theory predicts that altruistic behavior can evolve when sufficient fitness benefits are given to relatives. Moreover, either greenbeard effects or genetic kin recognition requires genetic polymorphisms as cues on which recognition is based. Previous models show that rare cue alleles get eliminated by selection and a common allele gets fixed, which ruins the altruism system. I designed a novel model with two types of genetic components, production and perception. I analyzed my recognition model theoretically toward a cost/benefit

analysis of fitness and genetic relatedness. I predicted the stability of recognition for altruism based on my model. Furthermore I tested my recognition model through various computational and biological simulations.

5.5. The fitness cost/benefit and the Crozier's Paradox

My simulation results consistently show altruism can maintain multiple recognition cues and be evolutionarily stable. I concluded that cost/benefit of fitness and genetic relatedness play a critical role in the evolution of altruism and eusociality, and therefore can maintain the stability of recognition for altruism.

5.6. Conclusions

I showed that relatedness is causal, that eusociality does evolve more readily being favored under a lower benefits threshold. I concluded that multiple modeling approaches are useful and that efforts to synthesize them are better than asserting that one is universally better than the other. I designed a novel model with two types of genetic components, production and perception. I analyzed my recognition model theoretically toward a cost/benefit analysis of fitness and genetic relatedness. I predicted the stability of recognition for altruism based on my model. Furthermore I tested my recognition model through various computational and biological simulations. My simulation results consistently show altruism can maintain multiple recognition cues and be evolutionarily stable. I concluded that

cost/benefit of fitness and genetic relatedness play a critical role in the evolution of altruism and eusociality, and therefore can maintain the stability of recognition for altruism.

5.7. References

Abbot, P. et al. Inclusive fitness theory and eusociality. *Nature* 471, E1-E4, doi:10.1038/nature09831 (2011).

Alexander, R. D. *The biology of moral systems*. (Aldine de Gruyter, 1987).

Boomsma, J. J. et al. Only full-sibling families evolved eusociality. *Nature* 471, E4-E5 (2011).

Bourke, A. F. G. & Franks, N. R. *Social Evolution in Ants*. (Princeton University Press, 1995).

Bourke, A. F. G. *Principles of social evolution*. (Oxford University Press, 2011).

Bourke, A. F. G. The validity and value of inclusive fitness theory. *Proceedings of the Royal Society B: Biological Sciences* 278, 3313-3320 (2011).

Brand, N. & Chapuisat, M. Impact of helpers on colony productivity in a primitively eusocial bee. *Behav. Ecol. Sociobiol.* 68, 291-298 (2014).

Crozier, R. H. & Pamilo, P. *Evolution of Social Insect Colonies: Sex Allocation and Kin Selection*. (Oxford University Press, 1996).

Ferriere, R. & Michod, R. E. Inclusive fitness in evolution. *Nature* 471, E6-E8 (2011).

Foster, K. F. in *Social behaviour: genes, ecology and evolution* (eds T Szekely, AJ Moore, & J Komdeur) 331-356 (Cambridge University Press, 2011).

Haig, D. *Genomic imprinting and kinship*. (Rutgers University Press, 2002).

Hamilton, W. D. Altruism and related phenomena, mainly in the social insects. *Annu. Rev. Ecol. Syst.* 3, 193-232 (1972).

Hamilton, W. D. The genetical evolution of social behaviour. I-II. *J. Theor. Biol.* 7, 1-52 (1964).

Herre, E. A. & Wcislo, W. T. In defence of inclusive fitness theory. *Nature* 471, E8-E9 (2011).

Hughes, W., Oldroyd, B., Beekman, M. & Ratnieks, F. Ancestral monogamy shows kin selection is key to the evolution of eusociality. *Science* 320, 1213-1216 (2008).

Kuzdzal-Fick, J. J., Fox, S. A., Strassmann, J. E. & Queller, D. C. High relatedness is necessary and sufficient to maintain multicellularity in *Dictyostelium*. *Science* 334, 1548-1551 (2011).

McGlothlin, J. W., Moore, A. J., Wolf, J. B. & Brodie, E. D. interacting phenotypes and the evolutionary process. III. Social evolution. *Evolution* 64, 2558-2574, doi:10.1111/j.1558-5646.2010.01012.x (2010).

Michod, R. E. The theory of kin selection. *Ann. Rev. Ecol. Syst.* 13, 23-55 (1982).

Noonan, K. M. in *Natural Selection and Social Behavior* (eds R. D. Alexander & D. W. Tinkle) 18-44 (Chiron, 1981).

Nowak, M. A., Tarnita, C. E. & Wilson, E. O. Nowak et. al reply. *Nature* 471, E9-E10 (2011).

Nowak, M. A., Tarnita, C. E. & Wilson, E. O. The evolution of eusociality. *Nature* 466, 1057-1062 (2010).

Queller, D. C. & Strassmann, J. E. Kin selection and social insects. *Bioscience* 48, 165-175 (1998).

Queller, D. C. Extended parental care and the origin of eusociality. *Proceedings of the Royal Society of London, Series B* 256, 105-111 (1994).

Queller, D. C. Quantitative genetics, inclusive fitness, and group selection. *Am. Nat.* 139, 540-558 (1992).

Rousset, F. & Lion, S. Much ado about nothing: Nowak et al.'s charge against inclusive fitness theory. *J. Evol. Biol.* 24, 1386-1392 (2011).

Shakarad, M. & Gadagkar, R. Colony founding in the primitively eusocial wasp, *Ropalidia marginata* (Hymenoptera: Vespidae). *Ecological Entomology* 20, 273-282 (1995).

Shreeves, G. & Field, J. Group size and direct fitness in social queues. *The American Naturalist* 159, 81-95 (2002).

Stevens, M. I., Hogendoorn, K. & Schwarz, M. P. Evolution of sociality by natural selection on variances in reproductive fitness: evidence from a social bee. *BMC Evol. Biol.* 7, 153 (2007).

Strassmann, J. E., Page, R. E., Robinson, G. E. & Seeley, T. D. Kin selection and eusociality. *Nature* 471, E5-E6, doi:10.1038/nature09833 (2011).

Taylor, P. D. & Frank, S. A. How to make a kin selection model. *Journal of Theoretical Biology* 180, 27-37 (1996).

Trivers, R. L. & Hare, H. Haplodiploidy and the evolution of the social insects. *Science* 191, 249-263 (1976).

Appendix A

A.1 Scripts for Altruism to Evolve with Genetic Relatedness

```
%-----%

function nowak_figure_4(T, S, diff)
    qs = 0.01;
    qss = 0.001;
    q = [qss:qss:0.005, 0.01:qs:0.35, 0.355:qss:0.365, 0.37:qs:0.89, 0.895:qss:0.905,
0.91:qs:1]';
    %q = [0.2, 0.6, 0.95, 1];
    %q = [0.6];
    L = length(q);

    [ X01, Xs1, b0, d0, m, b, d, alpha, eta ] = init(S);

    X0q = zeros(L, 1);
    Xq = zeros(L, 1);
    Cq = zeros(L, 1);

    Tq = zeros(L, 1);
    X0t = zeros(T, L);
    Xt = zeros(T, L);
    Ct = zeros(T, L);

    for i = 1:L;
        q0 = q(i);
        [X0q(i), Xq(i), Cq(i), Tq(i), X0t(1:T, i), Xt(1:T, i), Ct(1:T, i) ] = nowak(q0, S, T, X01,
Xs1, b0, d0, m, b, d, alpha, eta, diff);
    end

    plotq10( X0q, Xq, Cq, q);

end

%-----%

function [X01, Xs1, b0, d0, m, b, d, alpha, eta] = init(S)
    n = 1;
```

```

X01 = n;
Xs1 = zeros(1, S);
Xs1(1, 1) = n;
b0 = 0.5;
d0 = 0.1;
m = 3;
b = 4;
d = 0.01;
alpha = 0.1;
eta = 0.01;
end

```

```

%-----%

```

```

function [X0q, Xq, Cq, T0, X0, X, C] = nowak(q, S, T, X01, Xs1, b0, d0, m, b, d, alpha,
eta, diff)

```

```

    zerodef = 0.05;

```

```

    X0q = 0;
    Xq = 0;
    Cq = 0;
    T0 = T;

```

```

    X0 = zeros(T, 1);
    Xs = zeros(T, S);
    X = zeros(T, 1);
    C = zeros(T, 1);

```

```

    B = bd(S, m, b0, b);
    D = bd(S, m, d0, d);

```

```

    X0(1) = X01;
    Xs(1, :) = Xs1;

```

```

    X(1, :) = population( Xs(1, 1:S) );
    C(1) = sum( Xs(1, 2:S) );

```

```

    for t = 1:T-1
        % phi = 1 / ( X0(t) + eta * X(t) );
        phi = 1 / ( 1 + eta * (X0(t) + X(t)) );
    end

```

```

X0(t+1) = X0(t) + (b0 * phi - d0) * X0(t);
X0(t+1) = checknegative( X0(t+1) );

Mt = matrix( S, q, B, D, phi, alpha );
Mtdi = Mt / diff;
for di = 1:diff
    Xs(t, :) = ( Xs(t, :)' + Mtdi * Xs(t, :)' )';
end

%Xs(t+1, :) = ( Xs(t, :)' + Mt * Xs(t, :)' )';
Xs(t+1, :) = Xs(t, :);
Xs(t+1, :) = checknegative( Xs(t+1, : ) );

X(t+1) = population( Xs(t+1, : ) );
%X(t+1) = checknegative( X(t+1) );

C(t+1) = sum( Xs(t+1, 2:S) );
%C(t+1) = checknegative( C(t+1) );

if ( X(t+1) < zerodef || X0(t+1) < zerodef )
    T0 = t;
    X0q = X0(t+1);
    Xq = X(t+1);
    Cq = C(t+1);
    break;
end
end
% T0 = T;
% plotPopulation(X0, X, C, q, T0);

plotgroup(Xs, T0, q);

end

%-----%

function X = population( Xs )
[m, S] = size( Xs );
X = 0;
for i = 1:S
    X = X + i * Xs(1, i);
end

```

```

    if (m ~= 1)
        error('wrong!');
    end
end

```

```

%-----%

```

```

function BD = bd(S, m, bd0, bd)
% One possibility is to consider a simple step function with a critical colony size, m.
% For small colonies,  $i < m$ , the key parameters of the eusocial queen are the same
% as those of solitary females:  $b_i = b_0$  and  $d_i = d_0$ .
% For large colonies,  $i \geq m$ , the eusocial queen has an increased fecundity and a
% reduced death rate:  $b_i = b > b_0$  and  $d_i = d < d_0$ .
    BD = ones(S, 1) * bd;
    for i = 1:m-1
        BD(i) = bd0;
    end
end

```

```

%-----%

```

```

function Mt = matrix(S, q, B0, D, phi, alpha)
    B = B0 * phi;
    Mt = zeros(S, S);
    for i = 2:S
        Mt(1, i) = B(i) * (1 - q);
        Mt(i, i-1) = B(i-1) * q;
    end
    for i = 2:S
        if ((i-1) * alpha <= 1)
            Mt(i, i) = - ( B(i) * q + D(i) + (i-1) * alpha );
            Mt(i-1, i) = (i-1) * alpha;
        else
            Mt(i, i) = - ( B(i) * q + D(i) + 1 );
            Mt(i-1, i) = 1;
        end
    end
    Mt(1,1) = B(1) * (1 - q) - (B(1) * q + D(1));
    Mt(1,2) = B(2) * (1 - q) + alpha;

    for i = 1:S

```



```

        for j = 1:S
            if Mt(i, j) < -1
                %Mt(i, j) = -1;
            end
        end
    end
end
end

```

```

%-----%

```

```

function A = checknegative(A)
    [m, n] = size(A);
    for i = 1:m
        for j = 1:n
            if A(i, j) < 0
                A(i, j) = 0;
            end
        end
    end
end
end

```

```

%-----%

```

```

function plotq10( X0q, Xq, Cq, q )
    figh = figure('Position', [0 0 800 600], 'Color', 'w', 'Resize', 'off');

    subplot(2, 1, 1);
    plot(q, X0q, 'r', q, Xq, 'b');
    %annotation(1, 'q', 'Individuals');
    box off;
    xlabel('Probability to stay, q');
    xlim([0, 1]);
    set(gca, 'XTick', 0:0.1:1, 'XTickLabel', {'0', '', '0.2', '', '0.4', '', '0.6', '', '0.8', '', '1'});
    ylabel('Individuals');
    ylim([0, 600]);
    set(gca, 'YTick', 0:100:600);

    subplot(2, 1, 2);
    plot(q, Cq, 'b');
    %annotation(1, 'q', 'Colonies');

```

```

box off;
xlabel('Probability to stay, q');
xlim([0, 1]);
set(gca, 'XTick', 0:0.1:1, 'XTickLabel', {'0','0.2','0.4','0.6','0.8','1'});
ylabel('Colonies');
ylim([0, 100]);
set(gca, 'YTick', 0:20:100);

filename = 'nowak_figure_4';
saveas( figh, filename, 'fig');
end

```

```
%-----%
```

```

function annotation(T, xl, yl)
    box off;
    xlabel(xl);
    set(gca, 'XTick', 0:T/5:T);
    ylabel(yl);
    % ylim([0, 600]);
    % set(gca, 'YTick', 0:100:600);
end

```

```
%-----%
```

```

function plotPopulation(X0, X, C, q, T0)
    figh = figure('Position', [0 0 800 600], 'Color', 'w', 'Resize', 'off');
    [m,n] = size(X0);
    t = 1:T0;
    % [T0, X0(m), X(m)]

    subplot(3, 1, 1);
    plot(t, X0(1:T0), 'r');
    annotation(m, 't', 'Solitary');
    title(['q = ', num2str(q) ]);
    ylim([0, 600]);
    set(gca, 'YTick', 0:100:600); set(gca, 'XTick', 0:2*T0:T0);

    subplot(3, 1, 2);
    plot(t, X(1:T0), 'b');

```

```

    annotation(m, 't', 'Eusocial');
    ylim([0, 600]);
    set(gca, 'YTick', 0:100:600); set(gca, 'XTick', 0:2*T0:T0);

    subplot(3, 1, 3);
    plot(t, C(1:T0), 'b');
    annotation(m, 't', 'Colonies');
    ylim([0, 100]);
    set(gca, 'YTick', 0:20:100); set(gca, 'XTick', 0:2*T0:T0);

    filename = num2str(100*q);
    saveas( figh, filename, 'fig');

end

%-----%

function plotgroup(Xs, T0, q)

    fighg = figure('Position', [0 0 800 700], 'Color', 'w', 'Resize', 'off');
    semilogy(Xs(1:T0, 1:7));
    % legend('1', '2', '3', '4');
    legend('Size 1', 'Size 2', 'Size 3', 'Size 4', 'Size 5', 'Size 6', 'Size 7');
    ylim([1, 1000]);
    box off;
    xlabel('t');
    set(gca, 'XTick', 0:T0/5:T0);
    ylabel('Colonies');

    filename = strcat('nowak_figure_4_p', num2str(100*q));
    saveas( fighg, filename, 'jpg');
    saveas( fighg, filename, 'fig');
    T0
end

%-----%

function bmw = nowak_liao_asexual(b, m, w, T, Td)

```

```

[b0, d0, d, alpha, eta, n] = nowak_liao_asexual_init_constant();
D = nowak_liao_asexual_init_BD(m, d0, d, w);
B = nowak_liao_asexual_init_BD(m, b0, b, w);
[em, Xe, Xs]= nowak_liao_asexual_eusociality(b0, d0, B, D, alpha, eta, n, w, T, Td);

if (Xe > Xs)
    bmw = 1;
else
    bmw = 0;
end

end

%-----%

function nowak_liao_asexual_bmw(bmin, bstep, bmax, mmin, mstep, mmax, wmin,
wstep, wmax, T, Td)

    bset = [bmin:bstep:bmax];
    mset = [mmin:mstep:mmax];
    wset = [wmin:wstep:wmax];
    [bi, bn] = size(bset);
    [mi, mn] = size(mset);
    [wi, wn] = size(wset);

    bmw = zeros(bn, mn, wn);

    for i = 1:mn
        b = mset(i);
        for j = 1:mn
            m = mset(j);
            for k = 1:mn
                w = mset(k);
                bmw(i, j, k) = nowak_liao_asexual(b, m, w, T, Td);
            end
        end
    end

    nowak_liao_asexual_plot_bmw(bset, mset, wset, bmw);

    dlmwrite('bmw.data', bmw, 'delimiter', '\t');

```

end

%-----%

```
function [eqpb, Xetd, Xstd] = nowak_liao_asexual_eusociality(b0, d0, B, D, alpha, eta,
n, w, T, Td)
```

```
    eqpb = 0;
    zerodef = 0.05;
```

```
    Etd = zeros(1, w);
    Etd(1, 1) = n;
    Std = n;
```

```
    for t = 1:T-1
        for tdi = 1:Td
            Xetd = population_ES(Etd, w);
            Xstd = Std;
            if Xstd < zerodef
                eqpb = 1;
                return;
            elseif Xetd < zerodef
                eqpb = 0;
                return;
            end

            phi = 1 / (1 + eta * (Xetd + Xstd));
```

```
            Me = matrix_E(B, D, alpha, phi, w);
            Etd(1, :) = (Etd(1, :) + Me * Etd(1, :) / Td)';
```

```
            Std = Std + (phi * b0 - d0) * Std / Td;
```

```
        end
    end
end
```

%-----%

```
function X = population_ES( ESmm, w)
```

```

X = 0;
for i = 1:w
    X = X + i * ESmm(i);
end
end

```

```

%-----%

```

```

function Me = matrix_E(B0, D, alpha, phi, w)

```

```

    B = B0 * phi;
    Me = zeros(w, w);
    for i = 2:w-1
        Me(i, i-1) = B(i-1) ;
        Me(i, i) = -B(i) - D(i) - (i-1) * alpha;
        Me(i, i+1) = i * alpha;
    end
    Me(1,1) = -B(1) - D(1);
    Me(1,2) = alpha;
    Me(1,w) = B(w);

```

```

    Me(w,w-1) = B(w-1);
    Me(w,w) = -D(w) - (w-1) * alpha;

```

```

end

```

```

%-----%

```

```

function B = nowak_liao_asexual_init_BD(m, b0, b, mm)

```

```

    % One possibility is to consider a simple step function with a critical colony size, m.
    % For small colonies,  $i < m$ , the key parameters of the eusocial queen are the same
    % as those of solitary females:  $b_i = b_0$  and  $d_i = d_0$ .
    % For large colonies,  $i \geq m$ , the eusocial queen has an increased fecundity
    % and a reduced death rate:  $b_i = b > b_0$  and  $d_i = d < d_0$ .

```

```

    B = ones(mm, 1) * b;
    for i = 1:m-1
        B(i) = b0;
    end
end

```

```
%-----%
```

```
function [b0, d0, d, alpha, eta, n] = nowak_liao_asexual_init_constant()
```

```
    b0 = 0.5;
```

```
    d0 = 0.1;
```

```
    d = 0.01;
```

```
    alpha = 0.1;
```

```
    eta = 0.01;
```

```
    n = 100;
```

```
end
```

```
%-----%
```

```
function P = nowak_liao_asexual_init_PQ(ms, ml, p, mm)
```

```
    if (1 <= ms && ms <= ml && ms <= mm) % 1 <= ms <= ml, mm, and should be
integer.
```

```
    else
```

```
        error('Invalid values of ms, ml or mm.\n ');
```

```
    end
```

```
    P = ones(mm, 1) * p;
```

```
    if (ms > 1)
```

```
        P(1:ms-1) = 1;
```

```
    % else % ms == 1
```

```
    end
```

```
    if (ml <= mm)
```

```
        P(ml:mm) = 0;
```

```
    % else % ml > mm
```

```
    end
```

```
end
```

```

%-----%

function nowak_liao_asexual_bmw(bmin, bstep, bmax, mmin, mstep, mmax, wmax,
wstep, wmin, T, Td)
    bset = [bmin:bstep:bmax];
    mset = [mmin:mstep:mmax];
    wset = [wmin:wstep:wmax];
    [bi, bn] = size(bset);
    [mi, mn] = size(mset);
    [wi, wn] = size(wset);

    bmw = zeros(bn, mn, wn);

    for i = 1:mn
        b = mset(i);
        for j = 1:mn
            m = mset(j);
            for k = 1:mn
                w = mset(k);
                nowak_liao_asexual(b, m, w, T, Td);
            end
        end
    end

    plot_bmw(bset, mset, wset, bmw);

end

%-----%

```

```

function nowak_liao_asexual_plot_bmw(bs, ms, ws, bmw)

    [bn, mn, wn] = size(bmw);

    figh = figure('Position', [0 0 600 600], 'Color', 'w', 'Resize', 'off');

    for k = 1:wn
        for j = 1:mn
            for i = 1:bn
                if (bmw(i, j, k) == 1)

```



```

        plot3(bs(i), ms(j), ws(k), '*'); hold on;
    end
end
end
end

xlabel('b'); ylabel('m'); zlabel('w');

axis([bs(1), bs(bn), ms(1), ms(mn), ws(1), ws(wn)]);
%set(gca, 'XTick', 0:0.2:1);
set(gca, 'YTick', ms(1):1: ms(mn));
set(gca, 'YTick', ws(1):1: ws(wn));
box off;

saveas( figh, 'bmw', 'fig');

end

%-----%

function nowak_liao_asexual_plot_m(filename_m_plot, ms, Xe, Xs)

    figh = figure('Position', [0 0 600 600], 'Color', 'w', 'Resize', 'off');

    plot(ms, Xe, '-b*', ms, Xs, '-ro'); hold on;
    xlabel('w'); ylabel('Individuals'); title('m = 3');
    set(gca, 'XTick', ms(1):1:ms(length(ms))); %set(gca, 'YTick', 0:100:600);
    box off;

    saveas( figh, filename_m_plot, 'fig');

end

%-----%

function nowak_liao_asexual_plot_bmw(bs, ms, ws, bmw)

    [bn, mn, wn] = size(bmw);

```

```

figh = figure('Position', [0 0 600 600], 'Color', 'w', 'Resize', 'off');

for k = 1:wn
    for j = 1:mn
        for i = 1:bn
            if (bmw(i, j, k) == 1)
                plot3(bs(i), ms(j), ws(k), '*'); hold on;
            end
        end
    end
end

xlabel('b'); ylabel('m'); zlabel('w');

%axis([0, 1, 0, 1, bs(1), bs(bn)]);
%set(gca, 'XTick', 0:0.2:1); set(gca, 'YTick', 0:0.2:1);
%set(gca, 'ZTick', bs(1):1:bs(bn));
box off;

saveas( figh, 'bmw', 'fig');

end

%-----%

function mw = nowak_liao_asexual_w(wmin, wstep, wmax, m, b, T, Td)

    filename_m_plot = strcat('w', num2str(wmin), '_', num2str(wstep), '_',
num2str(wmax), '_T', num2str(T), '_Td', num2str(Td), '.fig');

    [b0, d0, d, alpha, eta, n] = nowak_liao_asexual_init_constant();
    %b = 4;

    wset = [wmin:wstep:wmax];
    [wi, wn] = size(wset);

    em = zeros(1, wn);
    Xe = zeros(1, wn);

```

```

Xs = zeros(1, wn);

for i = 1:wn
    w = wset(i);

    D = nowak_liao_asexual_init_BD(m, d0, d, w);
    B = nowak_liao_asexual_init_BD(m, b0, b, w);

    [em(i), Xe(i), Xs(i)] = nowak_liao_asexual_eusociality(b0, d0, B, D, alpha, eta, n, w,
T, Td);

    dlmwrite('b.data', B, 'delimiter', '\t', '-append');
    dlmwrite('d.data', D, 'delimiter', '\t', '-append');

end
nowak_liao_asexual_plot_m(filename_m_plot, wset, Xe, Xs);

dlmwrite('w.data', wset, 'delimiter', '\t');
dlmwrite('xe.data', Xe, 'delimiter', '\t');
dlmwrite('xs.data', Xs, 'delimiter', '\t');
dlmwrite('em.data', em, 'delimiter', '\t');

mw = zeros(1, wn);
for i = 1:wn
    if (Xe(i) > Xs(i))
        mw(i) = 1;
    end
end
end
end

%-----%

#PBS -N xl3
#PBS -V
#PBS -q commons
#PBS -l nodes=1,ppn=1,pmem=10m,walltime=2:00:00

```

```
cd /users/xl3/nowak_liao/nowak_liao_asexual_m/nowak_liao_asexual_m_/
matlab -nosplash -nodesktop -r "nowak_liao_asexual_m(2, 1, 10, 20, 5000, 10)"
exit
```

```
%-----%
```

```
function eqpb = nowak_liao_asexual_eusociality(Q, P, B, D, alpha, eta, n, mm, T, Td)
```

```
    eqpb = 0;
```

```
    zerodef = 0.05;
```

```
    Etd = zeros(1, mm);
```

```
    Std = zeros(1, mm);
```

```
    Etd(1, 1) = n;
```

```
    Std(1, 1) = n;
```

```
    for t = 1:T-1
```

```
        for tdi = 1:Td
```

```
            Xetd = population_ES(Etd, mm);
```

```
            Xstd = population_ES(Std, mm);
```

```
            if Xstd < zerodef
```

```
                eqpb = 1;
```

```
                return;
```

```
            elseif Xetd < zerodef
```

```
                eqpb = 0;
```

```
                return;
```

```
            end
```

```
            phi = 1 / (1 + eta * (Xetd + Xstd));
```

```
            bma = birth_migrate_average(P, Q, B, Etd, Std, mm);
```

```
            Me = matrix_E(P, Q, B, D, alpha, phi, bma, mm);
```

```
            Ms = matrix_S(P, Q, B, D, alpha, phi, bma, mm);
```

```
            Etd(1, :) = (Etd(1, :) + Me * Etd(1, :) / Td)';
```

```
            Std(1, :) = (Std(1, :) + Ms * Std(1, :) / Td)';
```

```
        end
```

```
    end
```

```
end
```

```
% ----- %
```

```
function X = population_ES( ESmm, mm)
```

```
    X = 0;
```

```
    for i = 1:mm
```

```
        X = X + i * ESmm(i);
```

```
    end
```

```
end
```

```
%-----%
```

```
function bm = birth_migrate_average(P, Q, B, Smm, Xmm, mm)
```

```
    os = 0;
```

```
    for i = 1:mm
```

```
        os = os + (1 - P(i)) * Q(i) * B(i) * Xmm(i);
```

```
    end
```

```
    bm = os / (sum(Smm) + sum(Xmm));
```

```
end
```

```
%-----%
```

```
function Me = matrix_E(P, Q, B0, D, alpha, phi, bma0, mm)
```

```
    B = B0 * phi;
```

```
    bma = bma0 * phi;
```

```
    Me = zeros(mm, mm);
```

```
    for i = 2:mm
```

```
        Me(1, i) = (1 - Q(i)) * B(i);
```

```
        Me(i, i-1) = P(i-1) * Q(i-1) * B(i-1) + bma ;
```

```
        Me(i, i) = - P(i) * Q(i) * B(i) - bma - D(i) - (i-1) * alpha;
```

```
        Me(i-1, i) = (i-1) * alpha;
```

```
    end
```

```
    Me(1,1) = (1 - Q(1)) * B(1) - P(1) * Q(1) * B(1) - bma - D(1);
```

```
    Me(1,2) = (1 - Q(2)) * B(2) + alpha;
```

```
end
```

```
%-----%
```

```
function Ms = matrix_S(P, Q, B0, D, alpha, phi, bma0, mm)
```

```
    B = B0 * phi;
```

```

bma = bma0 * phi;
Ms = zeros(mm, mm);
for i = 2:mm
    Ms(1, i) = B(i);
    Ms(i, i-1) = bma ;
    Ms(i, i) = - bma - D(i) - (i-1) * alpha;
    Ms(i-1, i) = (i-1) * alpha;
end
Ms(1,1) = B(1) - bma - D(1);
Ms(1,2) = B(2) + alpha;
end

```

```

%-----%

```

```

function B = nowak_liao_asexual_init_BD(m, b0, b, mm)
% One possibility is to consider a simple step function with a critical colony size, m.
% For small colonies,  $i < m$ , the key parameters of the eusocial queen are the same
as those of solitary females:  $b_i = b_0$  and  $d_i = d_0$ .
% For large colonies,  $i \geq m$ , the eusocial queen has an increased fecundity
% and a reduced death rate:  $b_i = b > b_0$  and  $d_i = d < d_0$ .

```

```

    B = ones(mm, 1) * b;
    for i = 1:m-1
        B(i) = b0;
    end
end

```

```

%-----%

```

```

function [b0, d0, d, alpha, eta, n] = nowak_liao_asexual_init_constant()

```

```

b0 = 0.5;
d0 = 0.1;
d = 0.01;

alpha = 0.1;
eta = 0.01;

n = 1;

```

```

end

%-----%

function P = nowak_liao_asexual_init_PQ(ms, ml, p, mm)
    if (1 <= ms && ms <= ml && ms <= mm) % 1 <= ms <= ml, mm, and should be
integer.
        else
            error('Invalid values of ms, ml or mm.\n ');
        end

        P = ones(mm, 1) * p;

        if (ms > 1)
            P(1:ms-1) = 1;
        % else % ms == 1
        end
        if (ml <= mm)
            P(ml:mm) = 0;
        % else % ml > mm
        end
    end
end

%-----%

function nowak_liao_asexual_plot_qp(filename, qs, ps, eqp)

    figh = figure('Position', [0 0 600 600], 'Color', 'w', 'Resize', 'off');

    [qn, pn] = size(eqp);

    for j = 1:pn
        for i = 1:qn
            if (eqp(i, j) == 1)
                plot(qs(i), ps(j), '*'); hold on;
            end
        end
    end

    xlabel('q'); ylabel('r');

```

```

axis([0, 1, 0, 1]);
set(gca, 'XTick', 0:0.1:1); set(gca, 'YTick', 0:0.1:1);
box off;

saveas( figh, filename, 'fig');

end

%-----%

function nowak_liao_asexual_plot_qpb(filename, qs, ps, bs, eqpb)

figh = figure('Position', [0 0 600 600], 'Color', 'w', 'Resize', 'off');

[qn, pn, bn] = size(eqpb);

for k = 1:bn

    for j = 1:pn

        for i = 1:qn

            if (eqpb(i, j, k) == 1)

                plot3(qs(i), ps(j), bs(k), '*'); hold on;

```



```

        end

    end

end

end

xlabel('q'); ylabel('r'); zlabel('b');

axis([0, 1, 0, 1, bs(1), bs(bn)]);

set(gca, 'XTick', 0:0.2:1); set(gca, 'YTick', 0:0.2:1);

set(gca, 'ZTick', bs(1):1:bs(bn));

box off;


saveas( figh, filename, 'fig');


end

%Xiaoyun Liao
%2010-12-31

function nowak_liao_asexual_qpb(qmin, qstep, qmax, pmin, pstep, pmax, bmin,
bstep, bmax, m, ms, ml, mm, T, Td)

    filename = strcat('_m', num2str(m), '_ms', num2str(ms), '_ml', num2str(ml), '_mm',
num2str(mm), '_T', num2str(T), '_Td', num2str(Td));

```

```

filename_qp = strcat('q', num2str(qmin), '_', num2str(qstep), '_', num2str(qmax),
'_p', num2str(pmin), '_', num2str(pstep), '_', num2str(pmax), filename);

[b0, d0, d, alpha, eta, n] = nowak_liao_asexual_init_constant();

qs = [qmin:qstep:qmax];
ps = [pmin:pstep:pmax];
bs = [bmin:bstep:bmax];
dlmwrite('q.data', qs, 'delimiter', '\t');
dlmwrite('p.data', ps, 'delimiter', '\t');
    dlmwrite('b.data', bs, 'delimiter', '\t');

[qi, qn] = size(qs);
    [pj, pn] = size(ps);
[bk, bn] = size(bs);
eqpb = zeros(qn, pn, bn);
D = nowak_liao_asexual_init_BD(m, d0, d, mm);
for k = 1:bn
    b = bs(k);
    B = nowak_liao_asexual_init_BD(m, b0, b, mm);
    for j = 1:pn
        p = ps(j);
        P = nowak_liao_asexual_init_PQ(ms, ml, p, mm);
        for i = 1:qn
            q = qs(i);
            Q = nowak_liao_asexual_init_PQ(ms, ml, q, mm);
            eqpb(i, j, k) = nowak_liao_asexual_eusociality(Q, P, B, D, alpha, eta, n, mm, T,
Td);
        end
    end

end

filename_b_eqp_plot = strcat('b', num2str(b), '_', filename_qp, '.fig' );
nowak_liao_asexual_plot_qp(filename_b_eqp_plot, qs, ps, eqpb(:,:,k));

filename_b_eqp_data = strcat('b', num2str(b), '_eqp.data');
dlmwrite(filename_b_eqp_data, eqpb(:,:,k), 'delimiter', '\t');
end

filename_qpb = strcat('b', num2str(bmin), '_', num2str(bstep), '_', num2str(bmax),
'_', filename_qp, '.fig' );
nowak_liao_asexual_plot_qpb(filename_qpb, qs, ps, bs, eqpb);

```

170

end

#PBS -N xl3

#PBS -V

#PBS -q commons

#PBS -l nodes=1:ppn=1,pmem=10m,walltime=2:00:00

cd

/users/xl3/nowak_liao/nowak_liao_asexual_qpb/nowak_liao_asexual_qpb_2010123
1/nowak_liao_asexual_qpb_201012314/

matlab -nosplash -nodesktop -r "nowak_liao_asexual_qpb(0.05, 0.05, 1, 0.05,
0.05, 1, 2, 2, 10, 3, 1, 21, 20, 20000, 10)"

exit

A.2 Scripts for Evolutionary Dynamics of Genetic Kin Recognition: a

General Model

```

function analysisCL(p, r, c, fJ0, G)
% Xiaoyun Liao xiao@rice.edu 2009-1-11
% Simulate the recognition model for altruism.

W0 = 1.0;
b = c / r;

fC0 = [0.01, 0.05, 0.1, 0.15, 0.2, 0.25, 0.5];
nfC0 = length(fC0);
L0 = [1, 0.9, 0.8, 0.7, 0.6, 0.5, 0.4, 0.3, 0.2, 0.1, 0];
nL0 = length(L0);

fig = figure('Position', [0 0 700 1000], 'Color', 'w', 'Resize', 'off'); hold
iPlot = 0;

for iL0 = 1:nL0;
    L = L0(iL0);
    for ifC0 = 1:nfC0
        fC0t = fC0(ifC0);
        fJ0t = fJ0;
        fG0 = [fC0t * fJ0t, fC0t * (1 - fJ0t), (1 - fC0t) * fJ0t, (1 - fC0t) * (1 - fJ0t)]; %
        Genotypes: C0J0, C0J1, C1J0, C1J1

        FG = zeros(4); % genotype frequencies: C0J0, C0J1, C1J0, C1J1
        WG = zeros(4); % genotype fitness
        for ifG0 = 1:4
            FG(ifG0) = fG0(ifG0);
        end

        FA = zeros(1 + G, 2); % allele frequencies: C0, J0
        FA(1, 1) = fC0t;
        FA(1, 2) = fJ0t;

        for t = 1 : G
            FAA = p * (FG(2) + FG(3)) + (1 - p) * (FG(2) * (FG(1) + FG(2)) + FG(3) *
(FG(3) + FG(4)));
            W = W0 + (b - c) * FAA;

```

```

WG(1) = W0 + (1 - p) * b * FG(2);
WG(2) = W0 + p * (b - c) + (1 - p) * ((b - c) * FG(2) - c * FG(1));
WG(3) = W0 + p * (b - c) + (1 - p) * ((b - c) * FG(3) - c * FG(4));
WG(4) = W0 + (1 - p) * b * FG(3);

FGW1 = FG(1) * WG(1) / W;
FGW2 = FG(2) * WG(2) / W;
FGW3 = FG(3) * WG(3) / W;
FGW4 = FG(4) * WG(4) / W;
FGWD = (FGW1 * FGW4 - FGW2 * FGW3) / 2;

% from time t to t + 1
FG(1) = (2 * (1 - L) - (1 - 2 * L) * FGW1) * FGW1 - (1 - L) * FGWD; % C0J0
FG(2) = (2 * L + (1 - 2 * L) * FGW2) * FGW2 + L * FGWD; % C0J1
FG(3) = (2 * L + (1 - 2 * L) * FGW3) * FGW3 + L * FGWD; % C1J0
FG(4) = (2 * (1 - L) - (1 - 2 * L) * FGW4) * FGW4 - (1 - L) * FGWD; % C1J1

FA(t + 1, 1) = FG(1) + FG(2); % C0 = C0J0 + C0J1
FA(t + 1, 2) = FG(1) + FG(3); % J0 = C0J0 + C1J0

end

% plot the simulation results
iPlot = iPlot + 1;
subplot(nL0, nfC0, iPlot); hold;
plot(FA(:, 1), FA(:, 2));
plot(FA(1, 1), FA(1, 2), 'ok', 'MarkerSize', 4);
plot(FA(G, 1), FA(G, 2), 'k', 'MarkerSize', 16);
xlim([-0.01 1.01]); ylim([-0.01 1.01]); box on; set(gca, 'DataAspectRatio', [1
1 1]);
set(gca, 'XTick', 0:1:1); set(gca, 'YTick', 0:1:1);
if iL0 == nL0 && ifC0 >= nfC0 / 2 && ifC0 < nfC0 / 2 + 1
    %xlabel({'f(C_0)'; '\itp = 0, f(C_0) = 0.25, f(J_0) = 0.2, f(C_0J_0) = 0.05. G =
10^5.'; '\itBlank dot: begin; black dot: end.'});
    xlabel('\itf(C_0)');
end
if ifC0 == 1 && iL0 >= nL0 / 2 && iL0 < nL0 / 2 + 1
    ylabel('\itf(J_0)');
end
if iL0 < nL0
    set(gca, 'XTickLabel', {'', ''});
end
end

```

```

if ifC0 > 1
    set(gca, 'YTickLabel', {'', ''});
end
if iL0 == 1
    switch ifC0
        case 1
            title('f_C_0=0.01');
        case 2
            title('f_C_0=0.05');
        case 3
            title('f_C_0=0.1');
        case 4
            title('f_C_0=0.15');
        case 5
            title('f_C_0=0.2');
        case 6
            title('f_C_0=0.25');
        case 7
            title('f_C_0=0.5');
        otherwise
            end
    end
end

if ifC0 == nfC0
    switch iL0
        case 1
            text(1, 0.5, ' l=1', 'LineStyle', 'none', 'HorizontalAlignment', 'Left');
        case 2
            text(1, 0.5, ' l=0.9', 'LineStyle', 'none', 'HorizontalAlignment', 'Left');
        case 3
            text(1, 0.5, ' l=0.8', 'LineStyle', 'none', 'HorizontalAlignment', 'Left');
        case 4
            text(1, 0.5, ' l=0.7', 'LineStyle', 'none', 'HorizontalAlignment', 'Left');
        case 5
            text(1, 0.5, ' l=0.6', 'LineStyle', 'none', 'HorizontalAlignment', 'Left');
        case 6
            text(1, 0.5, ' l=0.5', 'LineStyle', 'none', 'HorizontalAlignment', 'Left');
        case 7
            text(1, 0.5, ' l=0.4', 'LineStyle', 'none', 'HorizontalAlignment', 'Left');
        case 8
            text(1, 0.5, ' l=0.3', 'LineStyle', 'none', 'HorizontalAlignment', 'Left');
        case 9
            text(1, 0.5, ' l=0.2', 'LineStyle', 'none', 'HorizontalAlignment', 'Left');
    end
end

```

```

        case 10
            text(1, 0.5, ' l=0.1', 'LineStyle', 'none', 'HorizontalAlignment', 'Left');
        case 11
            text(1, 0.5, ' l=0', 'LineStyle', 'none', 'HorizontalAlignment', 'Left');
        otherwise
        end
    end
end
end
end
end

```

```
function analysisPL(r, c, fC0J0, fC0J1, fC1J0, TG)
```

```
% Xiaoyun Liao xiao@rice.edu 2009-1-11
```

```
% Analysis of the recognition model for altruism.
```

```
    fC1J1 = 1 - fC0J0 - fC0J1 - fC1J0;
```

```
    if (fC1J1 < 0)
```

```
        error('f!= 1.0');
```

```
    end
```

```
    W0 = 1.0;
```

```
    fG0 = [fC0J0, fC0J1, fC1J0, fC1J1]; % Genotypes: C0J0, C0J1, C1J0, C1J1
```

```
    %fA0 = [fC0J0 + fC0J1, fC1J0 + fC1J1, fC0J0 + fC1J0, fC0J1 + fC1J1]; % Alleles: C0, C1,  
J0, J1
```

```
    L0 = [1, 0.65, 0.6, 0.58, 0.56, 0.55, 0.54, 0.52, 0.5, 0.48, 0.45, 0.2, 0];
```

```
    nL = length(L0);
```

```
    p0 = [0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1];
```

```
    np = length(p0);
```

```
    fig = figure('Position', [0 0 700 700], 'Color', 'w', 'Resize', 'off'); hold
```

```
    iPlot = 0;
```

```

for iL = 1:nL
    L = L0(iL);

    for ip = 1:np
        p = p0(ip);

        b = c / r;

        FA = zeros(TG, 2); % frequencies of alleles: C0, J0

        FG = zeros(4); % frequencies of genotype: C0J0, C0J1, C1J0, C1J1
        WG = zeros(4); % mean fitness of genotype: C0J0, C0J1, C1J0, C1J1

        for ifG0 = 1:4
            FG(ifG0) = fG0(ifG0);
        end

        for t = 1 : TG
            FA(t, 1) = FG(1) + FG(2); % C0 = C0J0 + C0J1
            FA(t, 2) = FG(1) + FG(3); % J0 = C0J0 + C1J0

            FAA = p * (FG(2) + FG(3)) + (1 - p) * (FG(2) * (FG(1) + FG(2)) + FG(3) *
(FG(3) + FG(4)));

            W = W0 + (b - c) * FAA;

            WG(1) = W0 + (1 - p) * b * FG(2);
            WG(2) = W0 + p * (b - c) + (1 - p) * ((b - c) * FG(2) - c * FG(1));
            WG(3) = W0 + p * (b - c) + (1 - p) * ((b - c) * FG(3) - c * FG(4));
            WG(4) = W0 + (1 - p) * b * FG(3);

            FGW1 = FG(1) * WG(1) / W;
            FGW2 = FG(2) * WG(2) / W;
            FGW3 = FG(3) * WG(3) / W;
            FGW4 = FG(4) * WG(4) / W;
            FGWD = (FGW1 * FGW4 - FGW2 * FGW3) / 2;

            % from time t to t + 1
            FG(1) = (2 * (1 - L) - (1 - 2 * L) * FGW1) * FGW1 - (1 - L) * FGWD; % C0J0
            FG(2) = (2 * L + (1 - 2 * L) * FGW2) * FGW2 + L * FGWD; % C0J1
            FG(3) = (2 * L + (1 - 2 * L) * FGW3) * FGW3 + L * FGWD; % C1J0
            FG(4) = (2 * (1 - L) - (1 - 2 * L) * FGW4) * FGW4 - (1 - L) * FGWD; % C1J1

```



```

end

% plot the simulation results
iPlot = iPlot + 1;
subplot(nL, np, iPlot); hold;
plot(FA(:, 1), FA(:, 2));
plot(FA(1, 1), FA(1, 2), 'ok', 'MarkerSize', 4);
plot(FA(TG, 1), FA(TG, 2), 'k', 'MarkerSize', 16);
xlim([-0.01 1.01]); ylim([-0.01 1.01]); box on; set(gca, 'DataAspectRatio', [1
1 1]);
set(gca, 'XTick', 0:1:1); set(gca, 'YTick', 0:1:1);
if iL == nL && ip >= np / 2 && ip < np / 2 + 1
    xlabel('\itf(C_0)');
end
if ip == 1 && iL >= nL / 2 && iL < nL / 2 + 1
    ylabel('\itf(J_0)');
end
if iL < nL
    set(gca, 'XTickLabel', {'', ''});
end
if ip > 1
    set(gca, 'YTickLabel', {'', ''});
end
if iL == 1
    switch ip
        case 1
            title('p=0');
        case 2
            title('p=0.1');
        case 3
            title('p=0.2');
        case 4
            title('p=0.3');
        case 5
            title('p=0.4');
        case 6
            title('p=0.5');
        case 7
            title('p=0.6');
        case 8
            title('p=0.7');
        case 9
            title('p=0.8');
    end
end

```

```

        case 10
            title('p=0.9');
        case 11
            title('p=1');
        otherwise
        end
    end
end

if ip == np
    switch iL
        case 1
            text(1, 0.5, ' l=1', 'LineStyle', 'none', 'HorizontalAlignment', 'Left');
        case 2
            text(1, 0.5, ' l=0.65', 'LineStyle', 'none', 'HorizontalAlignment', 'Left');
        case 3
            text(1, 0.5, ' l=0.6', 'LineStyle', 'none', 'HorizontalAlignment', 'Left');
        case 4
            text(1, 0.5, ' l=0.58', 'LineStyle', 'none', 'HorizontalAlignment', 'Left');
        case 5
            text(1, 0.5, ' l=0.56', 'LineStyle', 'none', 'HorizontalAlignment', 'Left');
        case 6
            text(1, 0.5, ' l=0.55', 'LineStyle', 'none', 'HorizontalAlignment', 'Left');
        case 7
            text(1, 0.5, ' l=0.54', 'LineStyle', 'none', 'HorizontalAlignment', 'Left');
        case 8
            text(1, 0.5, ' l=0.52', 'LineStyle', 'none', 'HorizontalAlignment', 'Left');
        case 9
            text(1, 0.5, ' l=0.5', 'LineStyle', 'none', 'HorizontalAlignment', 'Left');
        case 10
            text(1, 0.5, ' l=0.48', 'LineStyle', 'none', 'HorizontalAlignment', 'Left');
        case 11
            text(1, 0.5, ' l=0.45', 'LineStyle', 'none', 'HorizontalAlignment', 'Left');
        case 12
            text(1, 0.5, ' l=0.2', 'LineStyle', 'none', 'HorizontalAlignment', 'Left');
        case 13
            text(1, 0.5, ' l=0', 'LineStyle', 'none', 'HorizontalAlignment', 'Left');
        otherwise
        end
    end
end
end
end

```

```

function analysisPR(c, fC0J0, fC0J1, fC1J0, fC1J1, TG)
% Xiaoyun Liao xiao@rice.edu 2009-1-11
% Analysis of the recognition model for altruism.

if (abs(fC0J0 + fC0J1 + fC1J0 + fC1J1 - 1) > 0.00001)
    error('f!= 1.0');
end

W0 = 1.0;
fG0 = [fC0J0, fC0J1, fC1J0, fC1J1]; % Genotypes: C0J0, C0J1, C1J0, C1J1
%fA0 = [fC0J0 + fC0J1, fC1J0 + fC1J1, fC0J0 + fC1J0, fC0J1 + fC1J1]; % Alleles: C0, C1,
J0, J1

r0 = [1, 0.95, 0.9, 0.85, 0.8, 0.5, 0.1];
nr = length(r0);

p0 = [0, 0.1, 0.3, 0.5, 0.7, 0.9, 1];
np = length(p0);

fig = figure('Position', [0 0 700 700], 'Color', 'w', 'Resize', 'off'); hold
iPlot = 0;

for ir = 1:nr
    r = r0(ir);

    for ip = 1:np
        p = p0(ip);

        b = c / r;

        FA = zeros(TG, 2); % frequencies of alleles: C0, J0
    end
end

```

```

FG = zeros(4); % frequencies of genotype: C0J0, C0J1, C1J0, C1J1
WG = zeros(4); % mean fitness of genotype: C0J0, C0J1, C1J0, C1J1

for ifG0 = 1:4
    FG(ifG0) = fG0(ifG0);
end

for t = 1 : TG
    FA(t, 1) = FG(1) + FG(2); % C0 = C0J0 + C0J1
    FA(t, 2) = FG(1) + FG(3); % J0 = C0J0 + C1J0

    FAA = p * (FG(2) + FG(3)) + (1 - p) * (FG(2) * (FG(1) + FG(2)) + FG(3) *
(FG(3) + FG(4)));

    W = W0 + (b - c) * FAA;

    WG(1) = W0 + (1 - p) * b * FG(2);
    WG(2) = W0 + p * (b - c) + (1 - p) * ((b - c) * FG(2) - c * FG(1));
    WG(3) = W0 + p * (b - c) + (1 - p) * ((b - c) * FG(3) - c * FG(4));
    WG(4) = W0 + (1 - p) * b * FG(3);

    FGW1 = FG(1) * WG(1) / W;
    FGW2 = FG(2) * WG(2) / W;
    FGW3 = FG(3) * WG(3) / W;
    FGW4 = FG(4) * WG(4) / W;
    FGWD = (FGW1 * FGW4 - FGW2 * FGW3) / 2;

    % from time t to t + 1
    FG(1) = FGW1 - FGWD; % C0J0
    FG(2) = FGW2 + FGWD; % C0J1
    FG(3) = FGW3 + FGWD; % C1J0
    FG(4) = FGW4 - FGWD; % C1J1

end

% plot the simulation results
iPlot = iPlot + 1;
subplot(nr, np, iPlot); hold;
plot(FA(:, 1), FA(:, 2));
plot(FA(1, 1), FA(1, 2), 'ok', 'MarkerSize', 4);
plot(FA(TG, 1), FA(TG, 2), 'k', 'MarkerSize', 16);
xlim([-0.01 1.01]); ylim([-0.01 1.01]); box on; set(gca, 'DataAspectRatio', [1
1 1]);

```

```

set(gca, 'XTick', 0:1:1); set(gca, 'YTick', 0:1:1);
if ir == nr && ip >= np / 2 && ip < np / 2 + 1
    xlabel('\itf(C_0)');
end
if ip == 1 && ir >= nr / 2 && ir < nr / 2 + 1
    ylabel('\itf(J_0)');
end
if ir < nr
    set(gca, 'XTickLabel', {'', ''});
end
if ip > 1
    set(gca, 'YTickLabel', {'', ''});
end
if ir == 1
    switch ip
        case 1
            title('p=0');
        case 2
            title('p=0.1');
        case 3
            title('p=0.3');
        case 4
            title('p=0.5');
        case 5
            title('p=0.7');
        case 6
            title('p=0.9');
        case 7
            title('p=1');
        otherwise
            end
    end
end

if ip == np
    switch ir
        case 1
            text(1, 0.5, ' c/b=1', 'LineStyle', 'none', 'HorizontalAlignment', 'Left');
        case 2
            text(1, 0.5, ' c/b=0.95', 'LineStyle', 'none', 'HorizontalAlignment',
'Left');
        case 3
            text(1, 0.5, ' c/b=0.9', 'LineStyle', 'none', 'HorizontalAlignment',
'Left');

```

```

        case 4
            text(1, 0.5, ' c/b=0.85', 'LineStyle', 'none', 'HorizontalAlignment',
'Left');
        case 5
            text(1, 0.5, ' c/b=0.8', 'LineStyle', 'none', 'HorizontalAlignment',
'Left');
        case 6
            text(1, 0.5, ' c/b=0.5', 'LineStyle', 'none', 'HorizontalAlignment',
'Left');
        case 7
            text(1, 0.5, ' c/b=0.1', 'LineStyle', 'none', 'HorizontalAlignment',
'Left');
        otherwise
        end
    end
end
end
end

```

```
function analysisRL(p, c, fC0J0, fC0J1, fC1J0, TG)
```

```
% Xiaoyun Liao xiao@rice.edu 2009-1-11
```

```
% Analysis of the recognition model for altruism.
```

```
    fC1J1 = 1 - fC0J0 - fC0J1 - fC1J0;
```

```
    if (fC1J1 < 0)
```

```
        error('f!= 1.0');
```

```
    end
```

```
    W0 = 1.0;
```

```
    fG0 = [fC0J0, fC0J1, fC1J0, fC1J1]; % Genotypes: C0J0, C0J1, C1J0, C1J1
```

```
    %fA0 = [fC0J0 + fC0J1, fC1J0 + fC1J1, fC0J0 + fC1J0, fC0J1 + fC1J1]; % Alleles: C0, C1,
J0, J1
```

```

L0 = [1, 0.65, 0.6, 0.58, 0.56, 0.55, 0.54, 0.52, 0.5, 0.48, 0.45, 0.2, 0];
nL = length(L0);

r0 = [0.1, 0.5, 0.6, 0.65, 0.7, 0.75, 0.8, 0.85, 0.9, 0.95, 1];
nr = length(r0);

fig = figure('Position', [0 0 700 700], 'Color', 'w', 'Resize', 'off'); hold
iPlot = 0;

for iL = 1:nL
    L = L0(iL);

    for ir = 1:nr
        r = r0(ir);

        b = c / r;

        FA = zeros(TG, 2); % frequencies of alleles: C0, J0

        FG = zeros(4); % frequencies of genotype: C0J0, C0J1, C1J0, C1J1
        WG = zeros(4); % mean fitness of genotype: C0J0, C0J1, C1J0, C1J1

        for ifG0 = 1:4
            FG(ifG0) = fG0(ifG0);
        end

        for t = 1 : TG
            FA(t, 1) = FG(1) + FG(2); % C0 = C0J0 + C0J1
            FA(t, 2) = FG(1) + FG(3); % J0 = C0J0 + C1J0

            FAA = p * (FG(2) + FG(3)) + (1 - p) * (FG(2) * (FG(1) + FG(2)) + FG(3) *
(FG(3) + FG(4)));

            W = W0 + (b - c) * FAA;

            WG(1) = W0 + (1 - p) * b * FG(2);
            WG(2) = W0 + p * (b - c) + (1 - p) * ((b - c) * FG(2) - c * FG(1));
            WG(3) = W0 + p * (b - c) + (1 - p) * ((b - c) * FG(3) - c * FG(4));
            WG(4) = W0 + (1 - p) * b * FG(3);

            FGW1 = FG(1) * WG(1) / W;
            FGW2 = FG(2) * WG(2) / W;

```

```

    FGW3 = FG(3) * WG(3) / W;
    FGW4 = FG(4) * WG(4) / W;
    FGWD = (FGW1 * FGW4 - FGW2 * FGW3) / 2;

    % from time t to t + 1
    FG(1) = (2 * (1 - L) - (1 - 2 * L) * FGW1) * FGW1 - (1 - L) * FGWD; % C0J0
    FG(2) = (2 * L + (1 - 2 * L) * FGW2) * FGW2 + L * FGWD; % C0J1
    FG(3) = (2 * L + (1 - 2 * L) * FGW3) * FGW3 + L * FGWD; % C1J0
    FG(4) = (2 * (1 - L) - (1 - 2 * L) * FGW4) * FGW4 - (1 - L) * FGWD; % C1J1

end

% plot the simulation results
iPlot = iPlot + 1;
subplot(nL, nr, iPlot); hold;
plot(FA(:, 1), FA(:, 2));
plot(FA(1, 1), FA(1, 2), 'ok', 'MarkerSize', 4);
plot(FA(TG, 1), FA(TG, 2), '.k', 'MarkerSize', 16);
xlim([-0.01 1.01]); ylim([-0.01 1.01]); box on; set(gca, 'DataAspectRatio', [1
1 1]);

set(gca, 'XTick', 0:1:1); set(gca, 'YTick', 0:1:1);
if iL == nL && ir >= nr / 2 && ir < nr / 2 + 1
    xlabel('\itf(C_0)');
end
if ir == 1 && iL >= nL / 2 && iL < nL / 2 + 1
    ylabel('\itf(J_0)');
end
if iL < nL
    set(gca, 'XTickLabel', {'', ''});
end
if ir > 1
    set(gca, 'YTickLabel', {'', ''});
end
if iL == 1
    switch ir
        case 1
            title('r=0.1');
        case 2
            title('r=0.5');
        case 3
            title('r=0.6');
        case 4
            title('r=0.65');
    end
end

```



```

        case 5
            title('r=0.7');
        case 6
            title('r=0.75');
        case 7
            title('r=0.8');
        case 8
            title('r=0.85');
        case 9
            title('r=0.9');
        case 10
            title('r=0.95');
        case 11
            title('r=1');
        otherwise
        end
    end
end

if ir == nr
    switch iL
        case 1
            text(1, 0.5, ' l=1', 'LineStyle', 'none', 'HorizontalAlignment', 'Left');
        case 2
            text(1, 0.5, ' l=0.65', 'LineStyle', 'none', 'HorizontalAlignment', 'Left');
        case 3
            text(1, 0.5, ' l=0.6', 'LineStyle', 'none', 'HorizontalAlignment', 'Left');
        case 4
            text(1, 0.5, ' l=0.58', 'LineStyle', 'none', 'HorizontalAlignment', 'Left');
        case 5
            text(1, 0.5, ' l=0.56', 'LineStyle', 'none', 'HorizontalAlignment', 'Left');
        case 6
            text(1, 0.5, ' l=0.55', 'LineStyle', 'none', 'HorizontalAlignment', 'Left');
        case 7
            text(1, 0.5, ' l=0.54', 'LineStyle', 'none', 'HorizontalAlignment', 'Left');
        case 8
            text(1, 0.5, ' l=0.52', 'LineStyle', 'none', 'HorizontalAlignment', 'Left');
        case 9
            text(1, 0.5, ' l=0.5', 'LineStyle', 'none', 'HorizontalAlignment', 'Left');
        case 10
            text(1, 0.5, ' l=0.48', 'LineStyle', 'none', 'HorizontalAlignment', 'Left');
        case 11
            text(1, 0.5, ' l=0.45', 'LineStyle', 'none', 'HorizontalAlignment', 'Left');
        case 12

```

```
        text(1, 0.5, ' l=0.2', 'LineStyle', 'none', 'HorizontalAlignment', 'Left');  
    case 13  
        text(1, 0.5, ' l=0', 'LineStyle', 'none', 'HorizontalAlignment', 'Left');  
    otherwise  
    end  
end  
end  
end  
end
```

A.3 Scripts for Exploring Fitness Cost/Benefit to Solve the Crozier's Paradox

```

function kinRecognition(cue, N, G)
% Simulate the model of kin recognition

% Author: Xiaoyun Liao
% Date: July 16, 2009
% Email: xiao@rice.edu

%
#####
##### %
% The MODEL OF KIN RECOGNITION
% Three genetic components: Action, Production, Perception
% ACTION: single action locus.
% PRODUCTION: single production locus
% PERCEPTION: multiple perception loci
% Matches of each production allele are accepted or rejected by certain perception
locus

%
#####
##### %
% Alleles in three components
% The set of action component alleles
    numberActionAllele = 5;
    setActionAllele = [1:numberActionAllele];
    %setActionAllele = ['Help' 'Recognized Help' 'Neutral' 'Recognized Exploit'
'Exploit'];
    %setActionAllele = [ 1      2          3      4      5  ];

% The set of production component alleles
    numberProductionAllele = cue;
    setProductionAllele = [1:numberProductionAllele];
    %setProductionAllele = ['A1' ; 'A2' ; 'A3' ];

```

```

%setProductionAllele = [ 1    2    3 ];

% The set of perception component alleles
numberPerceptionLocus = numberProductionAllele;
numberPerceptionAllele = 3;
setPerceptionAllele = [1:numberPerceptionAllele];
%setPerceptionAllele = [ 1    2    3 ];

%
#####
##### %
% RECOGNITION
% Match of production alleles
% Match only when they are the same allele
match_tmp = ones(1, numberProductionAllele);
matrixProductionAlleleMatch = diag(match_tmp);

% Accept of production alleles by perception alleles
% Accepted only by the first allele
matrixPerceptionAlleleAccept = [ 1  0  0.5];

%
#####
##### %
% FREQUENCY
freqActionAlleleSet = dlmread('actionallelefrequency.txt', '\t');
freqProductionAlleleSet = dlmread('productionallelefrequency.txt', '\t');
freqPerceptionAlleleSet = dlmread('perceptionallelefrequency.txt', '\t');

%
#####
##### %
% FITNESS
costBenefit = dlmread('costbenefit.txt', '\t');
costSet = costBenefit(1,:);
benefitSet = costBenefit(2, :);
cbratioSet = costBenefit(3, :);

```

```

%
#####
##### %
% CLONEMATE
probabilitySet = dlmread('probability.txt', '\t');

%
#####
#####%
% Simulation
oneSimulation;
%matrixCostCBRatio;

%
#####
#####%
function oneSimulation
    productionAlleleFrequency = multiplePerceptionLocus(freqActionAlleleSet(1, :),
freqProductionAlleleSet(1, :), freqPerceptionAlleleSet(1, :), probabilitySet(1),
costSet(1), benefitSet(3));
    figure('Position', [0 0 600 300], 'Color', 'w', 'Resize', 'off');
    plot(productionAlleleFrequency, '-o');
    ylim([0,1]);
    legend('A1', 'A2', 'A3');

end

%
#####
#####%
function matrixCostCBRatio
    [mCost, nCost] = size(costSet);
    [mCBRatio, nCBRatio] = size(cbratioSet);
    iplot = 0;
    mstep = 1;
    nstep = 1;

```

```

    mplot = ceil(nCBRatio / mstep);
    nplot = ceil(nCost / nstep);
    for iCBRatio = 1:mstep:nCBRatio
        for iCost = 1:nstep:nCost
            benefit = cost(iCost) / cbratio(iCBRatio);
            productionAlleleFrequency =
multiplePerceptionLocus(freqActionAlleleSet(1, :), freqProductionAlleleSet(1, :),
freqPerceptionAlleleSet(1, :), probabilitySet(1), costSet(iCost), benefit);

            iplot = iplot + 1;
            subplot(mplot, nplot, iplot);
            plot(productionAlleleFrequency(:, 1), productionAlleleFrequency(:, 2));
            axis([0, 1, 0, 1]);
        end
    end
end

%
#####
#####%
function matrixCostBenefit()
end

%
#####
#####%
function matrixCostClonemate()
end

%
#####
#####%
function matrixProductionAlleleFrequency()
end

%
#####
#####%

function SingleLocus ()
end

```

```

%
#####
#####%
%
function productionAlleleFrequency = multiplePerceptionLocus(freqActionAllele,
freqProductionAllele, freqPerceptionAllele, probability, cost, benefit)

    %rand('twister', sum(100*clock));
    rand('twister', 1000);

    productionAlleleFrequency = zeros(G, numberProductionAllele);
    perceptionAlleleFrequency = zeros(numberPerceptionLocus,
numberPerceptionAllele, G);
    %actionAlleleFrequency = zeros(G, numberActionAllele);

    offspringPopulation = createAncestor(freqActionAllele, freqProductionAllele,
freqPerceptionAllele);

    parentPopulation = offspringPopulation;
    for iG = 1:G
        productionAlleleFrequency(iG, 1:numberProductionAllele) =
calculateProductionAlleleFrequency(parentPopulation);
        perceptionAlleleFrequency(1:numberPerceptionLocus,
1:numberPerceptionAllele, iG) =
calculatePerceptionAlleleFrequency(parentPopulation);
        % actionAlleleFrequency(iG, 1:numberActionAllele) =
calculateActionAlleleFrequency(parentPopulation);

        parentPopulation = performAltruism(parentPopulation, probability, cost,
benefit);

        offspringPopulation = reproduce(parentPopulation);

        parentPopulation = offspringPopulation;
    end

    for iG = 1:G
        for iP = 1:numberPerceptionLocus
            freqtmp(iG, iP) = perceptionAlleleFrequency(iP, 1, iG);
        end
    end
    figure('Position', [0 0 600 300], 'Color', 'w', 'Resize', 'off');

```

```

    plot(freqtmp, '-o');
    ylim([0,1]);
    legend('A1 Accept', 'A2 Accept', 'A3 Accept');
end

```

```

%
#####
##### %
% Create ancestor population
function ancestor = createAncestor(freqActionAllele, freqProductionAllele,
freqPerceptionAllele)
    freqActionAlleleCumulativeSum = cumsum(freqActionAllele);
    freqProductionAlleleCumulativeSum = cumsum(freqProductionAllele);
    freqPerceptionAlleleCumulativeSum = cumsum(freqPerceptionAllele);

    if ((freqActionAlleleCumulativeSum(numberActionAllele) - 1.0) > 0.001)
        error('Action allele frequency initialized.');
```

```

    end
    if ((freqProductionAlleleCumulativeSum(numberProductionAllele) - 1.0) >
0.001)
        error('Production allele frequency initialized.');
```

```

    end
    if ((freqPerceptionAlleleCumulativeSum(numberPerceptionAllele) - 1.0) >
0.001)
        error('Perception allele frequency initialized.');
```

```

    end

    for iN = 1:2:N
        ancestor(iN) = createIndividual;
        ancestor(iN + 1) = ancestor(iN);
    end

function indi = createIndividual
    indi.action = createActionAllele;
    indi.production = createProductionAllele;
    indi.perception = createPerceptionAllele;
    indi.fitness = 1.0;
end

```



```

function actionAllele = createActionAllele
    randomProbability = rand;
    iA = 1;
    while(randomProbability >= freqActionAlleleCumulativeSum(iA))
        iA = iA + 1;
    end
    actionAllele = setActionAllele(iA);
end

function productionAllele = createProductionAllele
    randomProbability = rand;
    iP = 1;
    while(randomProbability >= freqProductionAlleleCumulativeSum(iP))
        iP = iP + 1;
    end
    productionAllele = setProductionAllele(iP);
end

function perceptionAllele = createPerceptionAllele
    perceptionAllele = zeros(1, numberPerceptionLocus);
    randomProbability = rand;
    for iPerLocus = 1:numberPerceptionLocus
        iPer = 1;
        while(randomProbability >= freqPerceptionAlleleCumulativeSum(iPer))
            iPer = iPer + 1;
        end
        perceptionAllele(iPerLocus) = setPerceptionAllele(iPer);
    end
end

end

%
#####
##### %

function productionAlleleFreq =
calculateProductionAlleleFrequency(population)
    productionAlleleFreq = zeros(1, numberProductionAllele);
    countProduction = zeros(1, numberProductionAllele);
    for iN = 1:N

```

```

        iProduction = population(iN).production;
        countProduction(1, iProduction) = countProduction(1, iProduction) + 1;
    end
    for iProduction = 1:numberProductionAllele
        productionAlleleFreq(1, iProduction) = 1.0 * countProduction(1,
iProduction) / N;
    end
end

function perceptionAlleleFreq =
calculatePerceptionAlleleFrequency(population)
    perceptionAlleleFreq = zeros(numberPerceptionLocus,
numberPerceptionAllele);
    countPerception = zeros(numberPerceptionLocus, numberPerceptionAllele);
    for iN = 1:N
        for iPerLocus = 1:numberPerceptionLocus
            iPerception = population(iN).perception(iPerLocus);
            countPerception(iPerLocus, iPerception) = countPerception(iPerLocus,
iPerception) + 1;
        end
    end
    for iPerLocus = 1:numberPerceptionLocus
        for iPerception = 1:numberPerceptionAllele
            perceptionAlleleFreq(iPerLocus, iPerception) = 1.0 *
countPerception(iPerLocus, iPerception) / N;
        end
    end
end

function actionAlleleFreq = calculateActionAlleleFrequency(population)
    actionAlleleFreq = zeros(1, numberActionAllele);
    countAction = zeros(1, numberActionAllele);
    for iN = 1:N
        iAction = population(iN).action;
        countAction(1, iAction) = countAction(1, iAction) + 1;
    end
    for iAction = 1:numberActionAllele
        actionAlleleFreq(1, iAction) = 1.0 * countAction(1, iAction) / N;
    end
end

```

```

%
#####
##### %
% Perform altruistic behaviors among parent population
function populationAfter = performAltruism(populationBefore, probability, cost,
benefit)

    for iN = 1:N
        iActor = ceil(N * rand);

        if (rand < probability) % clonemate as partner
            if (mod(iActor, 2) == 0)
                iRecipient = iActor - 1;
            else
                iRecipient = iActor + 1;
            end
        else % genetically random partner
            iRecipient = ceil(N * rand);
        end

        if (help(iActor, iRecipient))
            if (populationBefore(iActor).fitness >= cost)
                populationBefore(iActor).fitness = populationBefore(iActor).fitness - cost;
                populationBefore(iRecipient).fitness =
populationBefore(iRecipient).fitness + benefit;
            else
                %???
            end
        end
    end
    populationAfter = populationBefore;

function hel = help(iActor, iRecipient)
    hel = false;
    switch populationBefore(iActor).action
        case 1
            hel = true;
        case 2
            if (recognition(iActor, iRecipient))
                hel = true;
            end
        case 3

```

```

        hel = false;
    otherwise
        hel = false;
    end
end

```

```

function rec = recognition(iActor, iRecipient)
    rec = false;
    if (match(populationBefore(iActor).production,
populationBefore(iRecipient).production))
        if (accept(populationBefore(iActor).production,
populationBefore(iActor).perception))
            rec = true;
        end
    end
end

```

```

function mat = match(actorProduction, recipientProduction)
    if (rand < matrixProductionAlleleMatch(actorProduction, recipientProduction))
        mat = true;
    else
        mat = false;
    end
end

```

```

function acc = accept(actorProduction, actorPerception)
    if (rand < matrixPerceptionAlleleAccept(actorPerception(actorProduction)))
        acc = true;
    else
        acc = false;
    end
end

```

```

end

```

```

%
#####
##### %
% Create offspring population from parent population

```

```
function offspringPopulation = reproduce(parentPopulation)
```

```
    for iN = 1:N
        fit(iN) = parentPopulation(iN).fitness;
    end
    fitCumulativeSum = cumsum(fit);
    fitnessSum = fitCumulativeSum(N);

    for iN = 1:2:N
        offspringPopulation(iN) = reproduceIndividual();
        offspringPopulation(iN + 1) = offspringPopulation(iN);
    end
```

```
function indi = reproduceIndividual()
```

```
    iFather = randParent();
    iMother = randParent();

    indi = reproduceChild(iFather, iMother);
end
```

```
function iParent = randParent()
```

```
    randFitness = fitnessSum * rand();
    iParent = binarySearch(fitCumulativeSum, randFitness);
end
```

```
function index = binarySearch(dataSet, key)
```

```
    [m, n] = size(dataSet);
    left = 1;
    right = n;
    while (left <= right)
        middle = floor((left + right) / 2);
        if (key < dataSet(1, middle))
            right = middle - 1;
        else
            left = middle + 1;
        end
    end
    if (key < dataSet(1, middle))
        index = middle;
    else
        index = middle + 1;
    end
end
```

```

function child = reproduceChild(iFather, iMother)
    child.action = alternativeAction(parentPopulation(iFather).action,
parentPopulation(iMother).action);
    child.production =
alternativeProduction(parentPopulation(iFather).production,
parentPopulation(iMother).production);
    child.perception = alternativePerception(parentPopulation(iFather).perception,
parentPopulation(iMother).perception);
    child.fitness = 1.0;
end

function action = alternativeAction(fatherAction, motherAction)
    if (rand < 0.5)
        action = fatherAction;
    else
        action = motherAction;
    end
end

function production = alternativeProduction(fatherProduction,
motherProduction)
    if (rand < 0.5)
        production = fatherProduction;
    else
        production = motherProduction;
    end
end

function perception = alternativePerception(fatherPerception, motherPerception)
    for iPerLocus = 1:numberPerceptionLocus
        if (rand < 0.5)
            perception(iPerLocus) = fatherPerception(iPerLocus);
        else
            perception(iPerLocus) = motherPerception(iPerLocus);
        end
    end
end

end

```

```

%
#####
##### %
% Plot
function plotAlleleFrequency(actionAlleleFreq, productionAlleleFreq,
perceptionAlleleFreq)
    subplot(3,1,1);
    plot(actionAlleleFreq(1:G,1:3), '-o');
    legend('Alawys help','Recognized help','Never help');
    ylim([0 1]);
    subplot(3,1,2);
    plot(productionAlleleFreq(1:G,1:3), '-o');
    legend('A1','A2','A3');
    ylim([0 1]);
    subplot(3,1,3);
    % plot(perceptionAlleleFreq(1:G,1:3), '-o');
    % legend('a1','a2','a3');
    % ylim([0 1]);
end

function plotFrequency(productionAlleleFreq)
    plot(productionAlleleFreq(:,1:numberProductionAllele), '-o');
    %legend('A1','A2','A3');
    ylim([0 1]);
end

function plotFrequencySpace(productionAlleleFreq)
    plot(productionAlleleFreq(:,1), productionAlleleFreq(:,2), '-o');
    xlim([0 1]); ylim([0 1]);
end
end

```

```
function kinRecognitionSingleLocus(cue, N, G, iAction, iProduction, iPerception,
iProbability, iCost, iBenefit)
```

```
% Simulate the model of kin recognition
```

```
% Author: Xiaoyun Liao
```

```
% Date: 9/3/2009
```

```
% Email: xliao@rice.edu
```

```
%
```

```
#####
```

```
##### %
```

```
% The MODEL OF KIN RECOGNITION
```

```
% Three genetic components: Action, Production, Perception
```

```
% ACTION: single action locus.
```

```
% PRODUCTION: single production locus
```

```
% PERCEPTION: multiple perception loci
```

```
% Matches of each production allele are accepted or rejected by certain perception
locus
```

```
%
```

```
#####
```

```
##### %
```

```
% Alleles in three components
```

```
% The set of action component alleles
```

```
    numberActionAllele = 2;
```

```
    setActionAllele = [1:numberActionAllele];
```

```
    %setActionAllele = [ 1      2];
```

```
    %setActionAllele = ['Recognized Help' 'Always Help'];
```

```
% The set of production component alleles
```

```
    %numberProductinLocus = 1;
```

```
    numberProductionAllele = cue;
```

```
    setProductionAllele = [1:numberProductionAllele];
```

```
    %setProductionAllele = [ 1    2    3 ];
```

```
    %setProductionAllele = ['C1' 'C2' 'C3'];
```

```
% The set of perception component alleles
```

```
    %numberPerceptionLocus = numberProductionAllele;
```

```
    %numberPerceptionLocus = numberProductinLocus;
```

```
    numberPerceptionAllele = numberProductionAllele;
```

```
    setPerceptionAllele = [1:numberPerceptionAllele];
```



```

%setPerceptionAllele = [ 1  2  3];
%setPerceptionAllele = ['R1' 'R2' 'R3'];

%
#####
##### %
% RECOGNITION
% Match of production alleles
% Match only when they are the same allele
    match_tmp = ones(1, numberProductionAllele);
    matrixProductionAlleleMatch = diag(match_tmp);

% Accept of production alleles by perception alleles
% Accepted only by the first allele
    accept_tmp = ones(1, numberProductionAllele);
    matrixPerceptionAlleleAccept = diag(accept_tmp);
%accept_random = 0.5 * ones(numberProductionAllele, 1);
%matrixPerceptionAlleleAccept = cat(2, diag(accept_tmp), accept_random);

%
#####
##### %
% FREQUENCY
    freqActionAlleleSet = dlmread('actionallelefrequency.txt', '\t');
    freqProductionAlleleSet = dlmread('productionallelefrequency.txt', '\t');
    freqPerceptionAlleleSet = dlmread('perceptionallelefrequency.txt', '\t');

%
#####
##### %
% FITNESS
    costBenefit = dlmread('costbenefit.txt', '\t');
    costSet = costBenefit(:,1);
    benefitSet = costBenefit(:, 2);
    cbratioSet = costBenefit(:, 3);

    altruismFrequency = zeros(G, 1);
    iAltruism = 1;
%
#####
##### %
% CLONEMATE

```

```

probabilitySet = dlmread('probability.txt', '\t');

%
#####
#####%
% Simulation
oneSimulation;
%matrixCostCBRatio;

%
#####
#####%
function oneSimulation
    singleLocus(freqActionAlleleSet(iAction, :), freqProductionAlleleSet(iProduction, :), freqPerceptionAlleleSet(iPerception, :), probabilitySet(iProbability), costSet(iCost), benefitSet(iBenefit));

end

%
#####
#####%
function matrixCostCBRatio
    [mCost, nCost] = size(costSet);
    [mCBRatio, nCBRatio] = size(cbratioSet);
    iplot = 0;
    mstep = 1;
    nstep = 1;
    mplot = ceil(nCBRatio / mstep);
    nplot = ceil(nCost / nstep);
    for iCBRatio = 1:mstep:nCBRatio
        for iCost = 1:nstep:nCost
            benefit = cost(iCost) / cbratio(iCBRatio);
            productionAlleleFrequency =
multiplePerceptionLocus(freqActionAlleleSet(1, :), freqProductionAlleleSet(1, :),
freqPerceptionAlleleSet(1, :), probabilitySet(1), costSet(iCost), benefit);

```

```

        iplot = iplot + 1;
        subplot(mplot, nplot, iplot);
        plot(productionAlleleFrequency(:, 1), productionAlleleFrequency(:, 2));
        axis([0, 1, 0, 1]);
    end
end
end

%
#####
#####%
function matrixCostBenefit()
end

%
#####
#####%
function matrixCostClonemate()
end

%
#####
#####%
function matrixProductionAlleleFrequency()
end

%
#####
#####%

%
#####
#####%
%
function singleLocus(freqActionAllele, freqProductionAllele, freqPerceptionAllele,
probability, cost, benefit)

    rand('twister', sum(100*clock));
    %rand('twister', 500000);

```

```

productionAlleleFrequency = zeros(G, numberProductionAllele);
perceptionAlleleFrequency = zeros(G, numberPerceptionAllele);
actionAlleleFrequency = zeros(G, numberActionAllele);

productionAlleleFrequency(1, 1:numberProductionAllele) =
freqProductionAllele;
perceptionAlleleFrequency(1, 1:numberPerceptionAllele) =
freqPerceptionAllele;
actionAlleleFrequency(1, 1:numberActionAllele) = freqActionAllele;

offspringPopulation = createAncestor(freqActionAllele, freqProductionAllele,
freqPerceptionAllele);

parentPopulation = offspringPopulation;
for iG = 2:G
    productionAlleleFrequency(iG, 1:numberProductionAllele) =
calculateProductionAlleleFrequency(parentPopulation);
    perceptionAlleleFrequency(iG, 1:numberPerceptionAllele) =
calculatePerceptionAlleleFrequency(parentPopulation);
    actionAlleleFrequency(iG, 1:numberActionAllele) =
calculateActionAlleleFrequency(parentPopulation);

    parentPopulation = performAltruism(parentPopulation, probability, cost,
benefit);

    offspringPopulation = reproduce(parentPopulation);

    parentPopulation = offspringPopulation;
end
%plotAlleleFrequency(productionAlleleFrequency,
perceptionAlleleFrequency);
plotAlleleFrequency(actionAlleleFrequency, productionAlleleFrequency,
perceptionAlleleFrequency, altruismFrequency);

end

%
#####
##### %

```

```

% Create ancestor population
function ancestor = createAncestor(freqActionAllele, freqProductionAllele,
freqPerceptionAllele)

    freqActionAlleleCumulativeSum = cumsum(freqActionAllele);
    freqProductionAlleleCumulativeSum = cumsum(freqProductionAllele);
    freqPerceptionAlleleCumulativeSum = cumsum(freqPerceptionAllele);

    if ((freqActionAlleleCumulativeSum(numberActionAllele) - 1.0) > 0.001)
        error('Action allele frequency initialized.');
```

end

```

    if ((freqProductionAlleleCumulativeSum(numberProductionAllele) - 1.0) >
0.001)
        error('Production allele frequency initialized.');
```

end

```

    if ((freqPerceptionAlleleCumulativeSum(numberPerceptionAllele) - 1.0) >
0.001)
        error('Perception allele frequency initialized.');
```

end

```

    for iN = 1:2:N
        ancestor(iN) = createIndividual;
        ancestor(iN + 1) = ancestor(iN); %clonemate
    end

function indi = createIndividual
    indi.action = createActionAllele;
    indi.production = createProductionAllele;
    indi.perception = createPerceptionAllele;
    indi.fitness = 1.0;
end

function actionAllele = createActionAllele
    iA = 1;
    while(rand >= freqActionAlleleCumulativeSum(iA))
        iA = iA + 1;
    end
    actionAllele = setActionAllele(iA);
end

function productionAllele = createProductionAllele
    iP = 1;
```

```

        while(rand >= freqProductionAlleleCumulativeSum(iP))
            iP = iP + 1;
        end
        productionAllele = setProductionAllele(iP);
    end

    function perceptionAllele = createPerceptionAllele
        iPer = 1;
        while(rand >= freqPerceptionAlleleCumulativeSum(iPer))
            iPer = iPer + 1;
        end
        perceptionAllele = setPerceptionAllele(iPer);
    end

end

%
#####
##### %

function productionAlleleFreq =
calculateProductionAlleleFrequency(population)
    productionAlleleFreq = zeros(1, numberProductionAllele);
    countProduction = zeros(1, numberProductionAllele);
    for iN = 1:N
        iProduction = population(iN).production;
        countProduction(iProduction) = countProduction(iProduction) + 1;
    end
    for iProduction = 1:numberProductionAllele
        productionAlleleFreq(iProduction) = 1.0 * countProduction(iProduction) /
N;
    end
end

function perceptionAlleleFreq =
calculatePerceptionAlleleFrequency(population)
    perceptionAlleleFreq = zeros(1, numberPerceptionAllele);
    countPerception = zeros(1, numberPerceptionAllele);
    for iN = 1:N
        iPerception = population(iN).perception;
        countPerception(iPerception) = countPerception(iPerception) + 1;
    end
end

```

```

        for iPerception = 1:numberPerceptionAllele
            perceptionAlleleFreq(iPerception) = 1.0 * countPerception(iPerception) /
N;
        end
    end

function actionAlleleFreq = calculateActionAlleleFrequency(population)
    actionAlleleFreq = zeros(1, numberActionAllele);
    countAction = zeros(1, numberActionAllele);
    for iN = 1:N
        iAction = population(iN).action;
        countAction(iAction) = countAction(iAction) + 1;
    end
    for iAction = 1:numberActionAllele
        actionAlleleFreq(iAction) = 1.0 * countAction(iAction) / N;
    end
end

%
#####
##### %
% Perform altruistic behaviors among parent population
function populationAfter = performAltruism(populationBefore, probability, cost,
benefit)

    countAltruism = 0;
    for iN = 1:N
        iActor = ceil(N * rand);

        if (rand < probability) % clonemate as partner
            if (mod(iActor, 2) == 0)
                iRecipient = iActor - 1;
            else
                iRecipient = iActor + 1;
            end
        else % genetically random partner
            iRecipient = ceil(N * rand);
        end

        if (help(iActor, iRecipient))
            if (populationBefore(iActor).fitness >= cost)
                populationBefore(iActor).fitness = populationBefore(iActor).fitness - cost;
            end
        end
    end
end

```

```

        populationBefore(iRecipient).fitness =
populationBefore(iRecipient).fitness + benefit;
        countAltruism = countAltruism + 1;
    else
        %???
    end
end
end
altruismFrequency(iAltruism) = 1.0 * countAltruism / N;
iAltruism = iAltruism + 1;

populationAfter = populationBefore;

```

```

function hel = help(iActor, iRecipient)
hel = false;
switch populationBefore(iActor).action
    case 1
        if (recognition(iActor, iRecipient))
            hel = true;
        end
    case 2
        hel = true;
    case 3
        hel = false;
    otherwise
        hel = false;
    end
end
end

```

```

function rec = recognition(iActor, iRecipient)
rec = false;
if (match(populationBefore(iActor).production,
populationBefore(iRecipient).production))
    if (accept(populationBefore(iActor).production,
populationBefore(iActor).perception))
        rec = true;
    end
end
end
end

```

```

function mat = match(actorProduction, recipientProduction)

```



```

    if (rand < matrixProductionAlleleMatch(actorProduction, recipientProduction))
        mat = true;
    else
        mat = false;
    end
end

```

```

function acc = accept(actorProduction, actorPerception)
    if (rand < matrixPerceptionAlleleAccept(actorProduction, actorPerception))
        acc = true;
    else
        acc = false;
    end
end

```

```

end

```

```

%
#####
##### %
% Create offspring population from parent population

```

```

function offspringPopulation = reproduce(parentPopulation)

```

```

    for iN = 1:N
        fit(iN) = parentPopulation(iN).fitness;
    end
    fitCumulativeSum = cumsum(fit);
    fitnessSum = fitCumulativeSum(N);

    for iN = 1:2:N
        offspringPopulation(iN) = reproduceIndividual();
        offspringPopulation(iN + 1) = offspringPopulation(iN);
    end

```

```

function indi = reproduceIndividual()

```

```

    iFather = randParent();
    iMother = randParent();

```

```

    indi = reproduceChild(iFather, iMother);
end

```

```

function iParent = randParent()
    randFitness = fitnessSum * rand();
    iParent = binarySearch(fitCumulativeSum, randFitness);
end

```

```

function index = binarySearch(dataSet, key)
    [m, n] = size(dataSet);
    left = 1;
    right = n;
    while (left <= right)
        middle = floor((left + right) / 2);
        if (key < dataSet(1, middle))
            right = middle - 1;
        else
            left = middle + 1;
        end
    end
    if (key < dataSet(1, middle))
        index = middle;
    else
        index = middle + 1;
    end
end

```

```

function child = reproduceChild(iFather, iMother)
    child.action = alternativeAction(parentPopulation(iFather).action,
parentPopulation(iMother).action);
    child.production =
alternativeProduction(parentPopulation(iFather).production,
parentPopulation(iMother).production);
    child.perception = alternativePerception(parentPopulation(iFather).perception,
parentPopulation(iMother).perception);
    child.fitness = 1.0;
end

```

```

function action = alternativeAction(fatherAction, motherAction)
    if (rand < 0.5)
        action = fatherAction;
    else
        action = motherAction;
    end
end

```

```

function production = alternativeProduction(fatherProduction,
motherProduction)
    if (rand < 0.5)
        production = fatherProduction;
    else
        production = motherProduction;
    end
end

function perception = alternativePerception(fatherPerception, motherPerception)
    if (rand < 0.5)
        perception = fatherPerception;
    else
        perception = motherPerception;
    end
end

end

%
#####
##### %
% Plot
function plotAlleleFrequency(actionAlleleFreq, productionAlleleFreq,
perceptionAlleleFreq, altruismFreq)
    subplot(4,1,1);
    plot(actionAlleleFreq(1:G,1:numberActionAllele), '-o');
    legend('Recognized help','Alawys help');
    ylim([0 1]);
    subplot(4,1,2);
    plot(productionAlleleFreq(1:G, 1:numberProductionAllele), '-o');
    legend('Cue1', 'Cue2');
    ylim([0 1]);
    subplot(4,1,3);
    plot(perceptionAlleleFreq(1:G, 1:numberPerceptionAllele), '-o');
    legend('Rec1', 'Rec2');
    ylim([0 1]);
    subplot(4,1,4);
    plot(altruismFreq(1:G), '-o');
    %legend('R1', 'R2');

```

```

        ylim([0 1]);

    end

    function plotFrequency(productionAlleleFreq)
        plot(productionAlleleFreq(:,1:numberProductionAllele), '-o');
        %legend('A1','A2','A3');
        ylim([0 1]);
    end

    function plotFrequencySpace(productionAlleleFreq)
        plot(productionAlleleFreq(:,1), productionAlleleFreq(:,2), '-o');
        xlim([0 1]); ylim([0 1]);
    end

end

end

function kinRecognitionMultiplePerceptionLoci(cue, N, G)
% Simulate the model of kin recognition

% Author: Xiaoyun Liao
% Date: August 3, 2009
% Email: xliao@rice.edu

%
#####
##### %
% The MODEL OF KIN RECOGNITION
% Three genetic components: Action, Production, Perception
% ACTION: single action locus.
% PRODUCTION: single production locus
% PERCEPTION: multiple perception loci
% Matches of each production allele are accepted or rejected by certain perception
locus

```

```

%
#####
##### %
% Alleles in three components
% The set of action component alleles
    numberActionAllele = 5;
    setActionAllele = [1:numberActionAllele];
    %setActionAllele = [ 1    2        3    4    5 ];
    %setActionAllele = ['Help' 'Recognized Help' 'Neutral' 'Recognized Exploit'
'Exploit'];

% The set of production component alleles
    %numberProductinLocus = 1;
    numberProductionAllele = cue;
    setProductionAllele = [1:numberProductionAllele];
    %setProductionAllele = [ 1    2    3 ];
    %setProductionAllele = ['A1' 'A2' 'A3'];

% The set of perception component alleles
    numberPerceptionLocus = numberProductionAllele;
    %numberPerceptionLocus = numberProductinLocus;
    %numberPerceptionAllele = numberProductionAllele + 1;
    numberPerceptionAllele = 3;
    setPerceptionAllele = [1:numberPerceptionAllele];
    %setPerceptionAllele = [ 1    2    3    4 ];
    %setPerceptionAllele = ['a1' 'a2' 'a3' 'a0'];

%
#####
##### %
% RECOGNITION
% Match of production alleles
% Match only when they are the same allele
    match_tmp = ones(1, numberProductionAllele);
    matrixProductionAlleleMatch = diag(match_tmp);

% Accept of production alleles by perception alleles
% Accepted only by the first allele
    accept_tmp = ones(1, numberProductionAllele);
    accept_random = 0.5 * ones(numberProductionAllele, 1);

```

```

matrixPerceptionAlleleAccept = cat(2, diag(accept_tmp), accept_random);

%
#####
##### %
% FREQUENCY
freqActionAlleleSet = dlmread('actionallelefrequency.txt', '\t');
freqProductionAlleleSet = dlmread('productionallelefrequency.txt', '\t');
freqPerceptionAlleleSet = dlmread('perceptionallelefrequency.txt', '\t');

%
#####
##### %
% FITNESS
costBenefit = dlmread('costbenefit.txt', '\t');
costSet = costBenefit(1,:);
benefitSet = costBenefit(2, :);
cbratioSet = costBenefit(3, :);

%
#####
##### %
% CLONEMATE
probabilitySet = dlmread('probability.txt', '\t');

%
#####
#####%
% Simulation
oneSimulation;
%matrixCostCBRatio;

%
#####
#####%
function oneSimulation

```

```

    singleLocus(freqActionAlleleSet(1, :), freqProductionAlleleSet(6, :),
    freqPerceptionAlleleSet(5, :), probabilitySet(2), costSet(9), benefitSet(10));

```

```

end

```

```

%
#####
#####%

```

```

function matrixCostCBratio
    [mCost, nCost] = size(costSet);
    [mCBratio, nCBratio] = size(cbratioSet);
    iplot = 0;
    mstep = 1;
    nstep = 1;
    mplot = ceil(nCBratio / mstep);
    nplot = ceil(nCost / nstep);
    for iCBratio = 1:mstep:nCBratio
        for iCost = 1:nstep:nCost
            benefit = cost(iCost) / cbratio(iCBratio);
            productionAlleleFrequency =
multiplePerceptionLocus(freqActionAlleleSet(1, :), freqProductionAlleleSet(1, :),
freqPerceptionAlleleSet(1, :), probabilitySet(1), costSet(iCost), benefit);

```

```

            iplot = iplot + 1;
            subplot(mplot, nplot, iplot);
            plot(productionAlleleFrequency(:, 1), productionAlleleFrequency(:, 2));
            axis([0, 1, 0, 1]);
        end
    end
end

```

```

%
#####
#####%

```

```

function matrixCostBenefit()
end

```

```

%
#####
#####%

```

```

function matrixCostClonemate()

```

```

end

%
#####
#####%
function matrixProductionAlleleFrequency()
end

%
#####
#####%

%
#####
#####%
%
function singleLocus(freqActionAllele, freqProductionAllele, freqPerceptionAllele,
probability, cost, benefit)

    rand('twister', sum(100*clock));
    %rand('twister', 500000);

    productionAlleleFrequency = zeros(G, numberProductionAllele);
    perceptionAlleleFrequency = zeros(G, numberPerceptionAllele);
    % actionAlleleFrequency = zeros(G, numberActionAllele);

    offspringPopulation = createAncestor(freqActionAllele, freqProductionAllele,
freqPerceptionAllele);

    parentPopulation = offspringPopulation;
    for iG = 1:G
        productionAlleleFrequency(iG, 1:numberProductionAllele) =
calculateProductionAlleleFrequency(parentPopulation);
        perceptionAlleleFrequency(iG, 1:numberPerceptionAllele) =
calculatePerceptionAlleleFrequency(parentPopulation);
        % actionAlleleFrequency(iG, 1:numberActionAllele) =
calculateActionAlleleFrequency(parentPopulation);

        parentPopulation = performAltruism(parentPopulation, probability, cost,
benefit);

```



```

        offspringPopulation = reproduce(parentPopulation);

        parentPopulation = offspringPopulation;
    end

    plotAlleleFrequency(productionAlleleFrequency, perceptionAlleleFrequency);
    %plotFrequencySpace(freqActionAllele, freqProductionAllele,
    freqPerceptionAllele);

    end

%
#####
##### %
% Create ancestor population
    function ancestor = createAncestor(freqActionAllele, freqProductionAllele,
    freqPerceptionAllele)

        freqActionAlleleCumulativeSum = cumsum(freqActionAllele);
        freqProductionAlleleCumulativeSum = cumsum(freqProductionAllele);
        freqPerceptionAlleleCumulativeSum = cumsum(freqPerceptionAllele);

        if ((freqActionAlleleCumulativeSum(numberActionAllele) - 1.0) > 0.001)
            error('Action allele frequency initialized.');
```

end

```

        if ((freqProductionAlleleCumulativeSum(numberProductionAllele) - 1.0) >
0.001)
            error('Production allele frequency initialized.');
```

end

```

        if ((freqPerceptionAlleleCumulativeSum(numberPerceptionAllele) - 1.0) >
0.001)
            error('Perception allele frequency initialized.');
```

end

```

        for iN = 1:2:N
            ancestor(iN) = createIndividual;
            ancestor(iN + 1) = ancestor(iN); %clonemate
        end
    end
end

```

```

function indi = createIndividual
    indi.action = createActionAllele;
    indi.production = createProductionAllele;
    indi.perception = createPerceptionAllele;
    indi.fitness = 1.0;
end

function actionAllele = createActionAllele
    iA = 1;
    while(rand >= freqActionAlleleCumulativeSum(iA))
        iA = iA + 1;
    end
    actionAllele = setActionAllele(iA);
end

function productionAllele = createProductionAllele
    iP = 1;
    while(rand >= freqProductionAlleleCumulativeSum(iP))
        iP = iP + 1;
    end
    productionAllele = setProductionAllele(iP);
end

function perceptionAllele = createPerceptionAllele
    iPer = 1;
    while(rand >= freqPerceptionAlleleCumulativeSum(iPer))
        iPer = iPer + 1;
    end
    perceptionAllele = setPerceptionAllele(iPer);
end

end

%
#####
##### %

function productionAlleleFreq =
calculateProductionAlleleFrequency(population)
    productionAlleleFreq = zeros(1, numberProductionAllele);
    countProduction = zeros(1, numberProductionAllele);

```

```

    for iN = 1:N
        iProduction = population(iN).production;
        countProduction(iProduction) = countProduction(iProduction) + 1;
    end
    for iProduction = 1:numberProductionAllele
        productionAlleleFreq(iProduction) = 1.0 * countProduction(iProduction) /
N;
    end
end

```

```

function perceptionAlleleFreq =
calculatePerceptionAlleleFrequency(population)
    perceptionAlleleFreq = zeros(1, numberPerceptionAllele);
    countPerception = zeros(1, numberPerceptionAllele);
    for iN = 1:N
        iPerception = population(iN).perception;
        countPerception(iPerception) = countPerception(iPerception) + 1;
    end
    for iPerception = 1:numberPerceptionAllele
        perceptionAlleleFreq(iPerception) = 1.0 * countPerception(iPerception) /
N;
    end
end

```

```

function actionAlleleFreq = calculateActionAlleleFrequency(population)
    actionAlleleFreq = zeros(1, numberActionAllele);
    countAction = zeros(1, numberActionAllele);
    for iN = 1:N
        iAction = population(iN).action;
        countAction(iAction) = countAction(iAction) + 1;
    end
    for iAction = 1:numberActionAllele
        actionAlleleFreq(iAction) = 1.0 * countAction(iAction) / N;
    end
end

```

```

%
#####
##### %
% Perform altruistic behaviors among parent population
function populationAfter = performAltruism(populationBefore, probability, cost,
benefit)

```

```

for iN = 1:N
    iActor = ceil(N * rand);

    if (rand < probability) % clonemate as partner
        if (mod(iActor, 2) == 0)
            iRecipient = iActor - 1;
        else
            iRecipient = iActor + 1;
        end
    else % genetically random partner
        iRecipient = ceil(N * rand);
    end

    if (help(iActor, iRecipient))
        if (populationBefore(iActor).fitness >= cost)
            populationBefore(iActor).fitness = populationBefore(iActor).fitness - cost;
            populationBefore(iRecipient).fitness =
populationBefore(iRecipient).fitness + benefit;
        else
            %???
        end
    end
end
populationAfter = populationBefore;

function hel = help(iActor, iRecipient)
    hel = false;
    switch populationBefore(iActor).action
        case 1
            hel = true;
        case 2
            if (recognition(iActor, iRecipient))
                hel = true;
            end
        case 3
            hel = false;
        otherwise
            hel = false;
        end
    end
end

```

```

function rec = recognition(iActor, iRecipient)
    rec = false;
    if (match(populationBefore(iActor).production,
populationBefore(iRecipient).production))
        if (accept(populationBefore(iActor).production,
populationBefore(iActor).perception))
            rec = true;
        end
    end
end

function mat = match(actorProduction, recipientProduction)
    if (rand < matrixProductionAlleleMatch(actorProduction, recipientProduction))
        mat = true;
    else
        mat = false;
    end
end

function acc = accept(actorProduction, actorPerception)
    if (rand < matrixPerceptionAlleleAccept(actorProduction, actorPerception + 1))
        acc = true;
    else
        acc = false;
    end
end

end

%
#####
##### %
% Create offspring population from parent population

function offspringPopulation = reproduce(parentPopulation)

    for iN = 1:N
        fit(iN) = parentPopulation(iN).fitness;
    end
    fitCumulativeSum = cumsum(fit);

```

```

fitnessSum = fitCumulativeSum(N);

for iN = 1:2:N
    offspringPopulation(iN) = reproduceIndividual();
    offspringPopulation(iN + 1) = offspringPopulation(iN);
end

function indi = reproduceIndividual()
    iFather = randParent();
    iMother = randParent();

    indi = reproduceChild(iFather, iMother);
end

function iParent = randParent()
    randFitness = fitnessSum * rand();
    iParent = binarySearch(fitCumulativeSum, randFitness);
end

function index = binarySearch(dataSet, key)
    [m, n] = size(dataSet);
    left = 1;
    right = n;
    while (left <= right)
        middle = floor((left + right) / 2);
        if (key < dataSet(1, middle))
            right = middle - 1;
        else
            left = middle + 1;
        end
    end
    if (key < dataSet(1, middle))
        index = middle;
    else
        index = middle + 1;
    end
end

function child = reproduceChild(iFather, iMother)
    child.action = alternativeAction(parentPopulation(iFather).action,
    parentPopulation(iMother).action);

```

```

        child.production =
alternativeProduction(parentPopulation(iFather).production,
parentPopulation(iMother).production);
        child.perception = alternativePerception(parentPopulation(iFather).perception,
parentPopulation(iMother).perception);
        child.fitness = 1.0;
    end

function action = alternativeAction(fatherAction, motherAction)
    if (rand < 0.5)
        action = fatherAction;
    else
        action = motherAction;
    end
end

function production = alternativeProduction(fatherProduction,
motherProduction)
    if (rand < 0.5)
        production = fatherProduction;
    else
        production = motherProduction;
    end
end

function perception = alternativePerception(fatherPerception, motherPerception)
    if (rand < 0.5)
        perception = fatherPerception;
    else
        perception = motherPerception;
    end
end

end

%
#####
##### %
% Plot
function plotAlleleFrequency(productionAlleleFreq, perceptionAlleleFreq)
    %subplot(3,1,1);

```

```

    %plot(actionAlleleFreq(1:G,1:3), '-o');
    % legend('Alawys help','Recognized help','Never help');
    % ylim([0 1]);
    subplot(2,1,1);
    plot(productionAlleleFreq(1:G, 1:numberProductionAllele), '-o');
    legend('A1', 'A2');
    ylim([0 1]);
    subplot(2,1,2);
    plot(perceptionAlleleFreq(1:G, 1:numberPerceptionAllele), '-o');
    legend('a1', 'a2', 'a0');
    ylim([0 1]);
end

function plotFrequency(productionAlleleFreq)
    plot(productionAlleleFreq(:,1:numberProductionAllele), '-o');
    %legend('A1','A2','A3');
    ylim([0 1]);
end

function plotFrequencySpace(productionAlleleFreq)
    plot(productionAlleleFreq(:,1), productionAlleleFreq(:,2), '-o');
    xlim([0 1]); ylim([0 1]);

end

end

```