Department of
Computer Science

Technical Report

An examination of the role of prestige in cultural evolution with the aid of Agent Based Modelling

Christos Bechlivanidis

Technical Report 2006-21 **December 2006**

ISSN 1740-9497

Contact Address:
Department of Computer Science
University of Bath
Bath, BA2 7AY
United Kingdom
URL: http://www.cs.bath.ac.uk

ISSN 1740-9497

An examination of the role of prestige in cultural evolution with the aid of Agent Based Modelling

Christos Bechlivanidis
BSc (Hons) Computer Science

Supervisor: Dr Joanna J. Bryson

8th May 2006

An examination of the role of prestige in cultural evolution with the aid of Agent Based Modelling

Submitted by Christos Bechlivanidis

COPYRIGHT

DECLARATION

This dissertation is submitted to the University of Bath in accordance with the requirements of the degree of Bachelor of Science in the Department of Computer Science. No portion of the work in this dissertation has been submitted in support of an application for any other degree or qualification of this or any other university or institution of learning. Except where specifically acknowledged, it is the work of the author.

Signed: .....
Dated: .....

This dissertation may be made available for consultation within the University Library and may be photocopied or lent to other libraries for the purposes of consultation.

Signed: .....
Dated: .....

Abstract

Current research shows that natural selection favours evolved capacities in distinguishing the most successful individuals and copying the skills, beliefs and behaviours that contribute to their success. Prestige is an indication of success, attributed to individuals by their social environment. Prestige-biased infocopying is adaptive because it greatly reduces the effort required to obtain the knowledge and skills necessary for survival and reproduction in complex environments. The prestigious share their knowledge because they benefit from the formation of a social environment where information is freely available. Identifying the successful is a complicated process for which a variety of heuristics, such as age, appearance and number of offspring are employed. Using agent based modelling, we demonstrate the adaptiveness of prestige bias and we evaluate some of the heuristics used to show that age, for example, although it improves the quality of the model selection, is nevertheless misguiding, if blindly trusted. Similarly, we show that conformity bias is useful only when no better information is available. Finally, we illustrate that prestige biased learning and the existence of prestige hierarchies speed up the propagation of information but, at the same time, allow maladaptive information to persist and spread.

Acknowledgements

Many thanks to Dr. Joanna Bryson for the original idea and the active support throughout this project. Her open-mindedness and her genuine interest for scientific truth have impressed and motivated me.

Thanks also to Avri Bilovich whose work on a similar project was very instructive and also very useful as a starting point.

Finally, many thanks to Stella, who spent many hours of her life discussing about evolution with me. Many of the ideas that can be found in this text originated in those lengthy conversations.

Contents

Chapter 1

Introduction

Most anthropologists today agree that what differentiates humans from other species is epitomised in what is generally known as culture: the set of those traits that are transferred from one individual to the next by learning and observation. More specifically, given that there is evidence of culture in other primates, humans seem to possess the unique capacity to accumulate cultural elements so complicated that no individual learner could possibly devise them in a single lifetime. This accumulated knowledge is largely the explanation for the profound success of humans in inhabiting extremely diverse environments and prevailing over the other species on this planet, often, perhaps to a worrying degree. A huge quantity of this accumulated knowledge is offered to every human to assist in the dangerous mission for survival and reproduction, in highly competitive environments.

Because of this immense quantity of available knowledge, though, humans are confronted with another very crucial task: selecting the behaviours and attitudes to adopt from those that already exist and perhaps exercising their ingenuity to make small but decisive improvements. For this purpose numerous heuristics are employed, some of which will be briefly overviewed in this work.

The main purpose of this work, however, is not to investigate the way humans choose what to copy but the way they decide who to copy it from. We investigate what is known as "prestige biased infocopying": copying the knowledge, beliefs, behaviours and attitudes, the culture of the most successful within a group, in matters that are considered crucial for survival and reproduction. Copying the successful is considered an adaptive[1] strategy because it is a very effective shortcut: learning from the best, avoids the costs of having to learn by ourselves and provides some guarantee that the copied knowledge will be as useful for our own success.

Based on the influential work of some of the most distinguished scholars

[1] *Adaptive*, in the context of evolution and in the rest of this text, refers to those traits that are favoured by natural selection

from various fields such as anthropology and sociology, we aim to investigate how exactly prestige biased infocopying works, how it affects the accumulated culture and most importantly in which cases, if any, it fails, what are the undesired effects it may provoke.

In order to achieve our purpose we will make use of Agent Based Modelling (ABM). Agent based modelling is a research and experimental tool, used to examine the global effects of the interactions between individual members of a population, called agents. Agents, in this context, are autonomous computer programs that are placed in an environment and interact with each other, in various ways. ABM is becoming increasingly popular, especially in social sciences, because it makes possible to generate and study highly complex systems that are, nevertheless, constructed from relatively small sets of rules (Tobias and Hofmann, 2004). In appendix A we will overview agent based modelling and the available platforms for building such models.

In the remaining of this text we will:

- Overview genetical evolution as first described by Charles Darwin.
- Describe cultural evolution and explain the recent theory of memetics.
- Discuss the various ways with which culture is acquired by humans.
- Explain the way prestige bias works and describe the model we have build to further investigate it.
- Present and discuss the results of the various experiments we have conducted to investigate:
  - some of the heuristics that are used to find the prestigious and how efficient they are,
  - how prestige biased infocopying performs in highly complicated and in variable environments,
  - how prestige bias speeds up information propagation, and
  - how prestige bias allows maladaptive information to spread and persist
- Present the conclusions drawn from this research work.
- Discuss improvements and future work.

Chapter 2

Biological and cultural evolution

2.1 Darwinian evolution

The ground breaking theory of Charles Darwin, as first presented in "On the origin of species" in 1859, completely changed the way life development was understood and studied. Before Darwin, the platonian and aristotelian belief of species being fixed and classifiable according to their complexity was widely accepted. Darwin's *evolution* offered a better and more complete explanation of the origin and development of natural organisms and it had implications that are not yet fully understood and investigated.

In the early 19th century, Jean-Baptiste Lamarck was the first to express the idea that organisms are changing in order to adapt to their environment and moreover those changes are passed to their offspring (Eiseley, 1958; Packard, 1901). Based on this new idea, Charles Darwin in 1859, and, to a lesser extent, Alfred Russell Wallace one year earlier, explained the diversity observed in nature as the evolution of the species from a common ancestor (Darwin, 1859).

Initially a hypothesis and during the 20th century empirically confirmed, it provoked and at some level still provokes resistance, perhaps due to the uniform manner with which it views humans and other species. Similar to the observations of Copernicus in the 16th century, we are no longer in the centre with everything else revolving around us.

The main mechanism that drives evolution is known as *natural selection*. It is the process by which some traits persist and spread while others diminish and ultimately disappear. In brief, this occurs when certain characteristics improve the chances an individual organism has to survive, its fitness. This individual will probably live for longer, will produce more offspring and may pass these traits to some of his offspring. Therefore, the traits that increase the fitness of individuals that carry them, have increased

chances to spread. Natural selection occurs when the following conditions are met:

1. Variation: Not all individuals share the same traits. More importantly there is variation in the traits that increase the fitness of their carriers.

2. Competition: Resources are limited in the environment allowing only a sample of the population to survive and reproduce and thus leading to competition between individuals.

3. Heredity: Traits are transmissible by one individual to the next during reproduction.

However, since selection is not responsible for introducing new traits but only filtering those existing, it is not the only process in evolution. Genetic drift, mutation, recombination and gene flow are all important for evolution. Natural selection is, nevertheless, the main non-random mechanism that operates according to certain "rules".

2.2 Culture and cultural evolution

Variation, especially in humans can not be explained by genetic evolution alone. Even if our genes determine the colour of our eyes or that of our skin, they surely can not be held responsible for whether we speak English or Greek whether we vote for Bush or Kerry. The complexity observed in the human behaviour is amazing and one of the most amazing examples is the capacity of humans to inhabit extremely diverse physical environments. This capacity is not due only to our evolved DNA. If we take a infant born in a dessert in Africa and give it to an Inuit family to raise it near the Arctic, it will, all else being equal, have the same chances to survive and thrive as her peers. Thus, the classical question, nature vs. nurture, is misplaced. As Boyd and Richerson (2005) very eloquently pose it "...genes are like a recipe, but one in which the ingredients, cooking temperature, and so on are set by the environment".

The set of all those traits that determine our behaviour and are transmitted via social interaction, form what is known as *culture* (Henrich and McElreath, 2003; Richerson and Boyd, 2005; Bryson and Wood, 2005). They include but are not limited to knowledge, beliefs, ideas, attitudes etc. Not only are those elements of culture largely responsible for the variation among humans but most importantly they play a major role in our survival and reproduction. For example, the ability to successfully hunt for animals, although partially determined by genetic characteristics such as speed or physical strength, culturally acquired knowledge, such as effective tracing, is, in most cases, equally crucial. In a foraging society, hunting and therefore survival and reproduction are highly affected by culture.

Culture is not an exclusive privilege of humans. Although there is a debate on the issue, many scholars argue that cultural variation is common in nature (Boyd and Richerson, 1996; Boesch and Tomasello, 1998). What makes humans so different, especially in their capacity to adapt to different environments, is the fact that human culture is cumulative (Henrich and McElreath, 2003; Boesch and Tomasello, 1998; Richerson and Boyd, 2005). We rarely reinvent the wheel, as many primates do (see 3.2.1) but usually build on existing knowledge. As described in section 2.1, for example, Darwin's ideas were not a result of his ingenuity alone, since Lamarck and probably other scholars had already paved the way.

Thus, in humans, there is (1) competition, (2) variation in cultural items that contribute to the fitness of their carriers and (3) heredity, not only vertically but mainly horizontally, among members of the same generation. Which means that natural selection acts not only on biologically inherited traits but also on element of culture.

The above line of thought, lead some scholars in the mid 1970s (e.g. Richard Alexander, E. O. Wilson, Napoleon Chagnon, Bill Irons, and Don Symons (Boyd and Richerson, 2005a)) start thinking that *culture also evolves* not only in the obvious sense that societies change over time and they become larger and more complex but most interestingly in the Darwinian sense of evolution. At any given time and space, a set of cultural alternatives are proposed for any given problem such as hunting techniques, construction of tools, human dialects etc. The most beneficial of those usually survive generation after generation, while others either mutate and adapt or are unavoidably lost. Of course, this process is not perfect and that is also an interesting issue that will be addressed later. What is most important, at this point, is that culture seems to undergo a process that is very similar to biological natural selection, as described by Charles Darwin. That is why we said earlier that the observations of Darwin are more profound than many of us think and have consequences that are not yet studied completely. Evolution seems to be ubiquitous, seems to be strongly tied to humans and human history.

2.3 Memetics

The idea that culture evolves was carried even further by the rather recent and very controversial theory of memetics. The term meme was coined by zoologist Richard Dawkins in the last chapter of his book "The Selfish Gene" (1976). Dawkins was actually trying to present another example of replicators, i.e. mechanisms that copy themselves in a way similar to genes. However, his idea that cultural elements or memes, behave in a way very similar to genes and thus culture can be studied using approved methods borrowed from biology and genetics was very influential to many scholars.

Although there are various ideas of what a meme actually is, the Oxford English Dictionary defines it as "a cultural or behavioural element passed on by imitation or other non-genetic means"(Fowler et al., 2004). A vast array of things can be thought of as memes: ideas, inventions, religious, political and other beliefs, musical tunes, behaviours etc, in other words elements of culture. Memes exist in human brains or are stored in written or other form and are transmitted using some form of social interaction. In fact, this transmission is thought by many to operate in a fashion very similar to the way viruses spread (Brodie, 1996; Dennett, 1995). A person is not just "learning" a meme, he is rather infected by it. Memes are competing for a place in a person's mind and the most "fit" manage to survive and spread. What makes a meme fitter than a competing one is a complicated issue but it primarily depends on meme content (e.g. simplicity) and the benefits it promises to its carrier. For example, memes related to religion, often promise life after-death, that is both simple as an idea and, if believed, extremely comforting. A rich terminology has been proposed to support the theory of memes. For example, meme complex or memeplex is a set of cooperating memes that co-evolve and are replicated together such as religions, ideologies etc.(Blackmore, 1999, p.19).

Psychologist Susan Blackmore, one of the strongest supporters of memetics, has taken the theory one step further by proposing that our beliefs and even our consciousness are "created by and for the replication of memes. Ḿyb́eliefs and opinions are survival tricks used by memes for their own perpetuation. Ḿyćreativity is really design by memetic evolution ... human nature is a product of memes and genes competing for replication in a complex environment..." (Blackmore, 2000, p.41).

The memetics meme spread very quickly, mainly for providing a simple and "user-friendly" way to decode and study culture. This popularity had, of course, the disorientating effect of many different people from various disciplines and with unknown methodologies expressing innumerable views on memetics and taking the theory even further, sometimes even to the point of attributing almost metaphysical properties to memes (e.g. completely autonomous entities, with their own beliefs and desires that determine both individuals and the human species as a whole).

Due to that and also due to issues related with the core of memetics, many voices were raised, disagreeing and even being dismissive of the whole theory. Among others, Dan Sperber argued that memes are not copied with the same high fidelity as genes and thus they cannot be thought as replicators (Sperber, 2000). When someone explains an idea to someone else it is not unlikely that during the process of transforming thoughts into language and back again in the listener's mind, the original meme will be transformed as well leading to the listener finally not possessing the meme that was transmitted by the speaker.

Another objection is based on the fact that we cannot see memes in any

way, we cannot prove their existence, there is not even a concrete theory as to where memes reside and what their physical form is. On the other hand, memeticists, argue that neither Darwin could see or prove the existence of genes when he formed his, now proved and widely accepted, theory. Moreover, Dennett argues that even genes are not clearly defined, so neither do memes have to be. (Dennett, 2002; Bryson and Wood, 2005).

What is most important, though, is not whether memes are replicators, as Dawkins said, or whether they behave exactly as genes, even if that would simplify parts of sociological research. The most crucial point is that culture evolves. Whether we call them memes or cultural variants or any other name, the point is that they undergo a process that is at least similar to Darwinian evolution. Or as Boyd and Richerson argue:"... culture need not be closely analogous to genes. Ideas must be gene-like to the extent that they are somehow capable of carrying the cultural information necessary to give rise to the cumulative evolution of complex cultural patterns that differentiate human groups. They exhibit the essential Darwinian properties ... but ... this can be accomplished by a most ungene-like, replicatorless process" (Boyd and Richerson, 2000, p.158).

In what follows, we will be using the terms "cultural item" and "meme" interchangeably.

2.4 Relation between biology and culture: dual inheritance theory

Genetic and cultural evolution are interrelated. According to *dual inheritance theory*, a term due to Boyd and Richerson (1985), human behaviour is the result of the interaction between genetic and cultural traits. "Culture affects the success and survival of individuals and groups", " ... culturally evolved environments then affect which genes are favoured by natural selection"(Richerson and Boyd, 2005, p.4)

First of all, various biological factors influence or determine the cultural variants that appear and those that persist. For example, the shape of a fork is determined by the physical properties of the human mouth but the use or not of a fork is related to beliefs, i.e. purely elements of culture, such as tidiness, elegance etc. Therefore, although the use of fork is not present in every culture, wherever it exists, it has common properties (whether it's a fork, chopsticks etc.), imposed by the biology of the human body.

At the same time, as mentioned earlier, culture influences biological natural selection. Cultural evolution changes the terms of the Darwinian competition, imposes new rules and weakens some others. In modern societies, physical strength is no longer a crucial requirement for success as it was in the societies of our ancestors. On the other hand, access to and effective evaluation of information, for example, seems a much needed skill. So cul-

tural evolution does not change the Darwinian game, it changes the rules of the game.

Chapter 3

Sources of knowledge and behaviour

In the evolved environment formed by genetic and cultural evolution, acquiring the knowledge and behaviour that will enable an individual to succeed in surviving and reproducing is a very complicated and demanding task.

In this chapter, we will overview, the various ways with which knowledge is acquired by humans. We have arranged these sources of knowledge in two broad categories: those that depend on social interactions (epidemiological culture) and those that are either genetically inherited (evoked culture) or developed by an individual alone (individual learning) but in any case are not socially transmitted.

3.1 Evoked culture and individual learning

3.1.1 Evoked culture

In previous chapters we referred to culture as the set of traits that are socially and not genetically transmitted. Even if the term "evoked culture" is not perfectly consistent with this definition of culture, it is widely used to refer to cognitive models that are genetically transmitted by our ancestors, encoded in our DNA and "activated" by the environment. According to Tooby and Cosmides (Tooby and Cosmides, 1989) who coined the term, different environments will trigger different behaviours and thus evoked culture can explain both the similarities and, at least partially, the variations observed in human behaviour.

A suitable domain in which this idea is widely studied is the human language. Evoked culture is a very good candidate in order to explain the capacity of humans at a very early stage of their development to effectively use such a highly complicated mechanism as language. The existence of basic structural linguistic rules that are stored in our DNA, can explain

both the above observation as well as the existence of some striking similarities in the way all languages are structured. Extensive work on this subject was undertaken by Noam Chomsky whose "Universal Grammar" is used to illustrate those similarities. According to the famous linguist: "... certain aspects of our knowledge and understanding are innate, part of our biological endowment, genetically determined, on a par with the elements of our common nature that cause us to grow arms and legs rather than wings."(Chomsky, 1988).

The extreme view that some nativists and innate psychologists are taking, is considering evoked culture by itself sufficient to explain diversity in human behaviour. In other words, they believe that most of our behaviour is genetically transmitted. This view, perhaps first expressed by Plato, according to whom we never learn anything we just "remember", takes us once again to the nature vs. nurture dichotomy. However, even if evoked culture can explain the variations in space, since different environments are triggering different behaviours, it seems extremely difficult to explain the variations in time: the rate at which our culture is changing is undeniably much faster than the speed with which our genes are evolving.

3.1.2 Individual learning

In whatever fashion knowledge is transmitted, genetically or via social interactions, in order to explain evolution, we require a mechanism that will result in the emergence of new cultural items and will allow adaptation in situations where no prior information is available. Individual learning is a crucial process, that introduces new candidates for cultural natural selection.

Innovations in the human culture are accomplished in "small incremental steps" (Richerson and Boyd, 2005). Each "innovator" is actually extending or combining existing cultural items and thus making a relatively small contribution in the evolving puzzle. Boyd and Richerson (2005) give the example of the discovery of the magnetic compass to demonstrate the fact that "culture evolves by the accumulation of small steps" and to show that even relatively simple tools are "the product of numerous innovations over centuries". They explain that in order for this relatively simple device to acquire its present form, dozens of known or unknown inventors have contributed a piece, from the observation of the attribute of some objects to point to the North to the use of iron balls to cancel the magnetic influence of its surroundings. This example also illustrates another important property of human culture, its cumulative nature (see also 3.3).

Trial and error is the basic method in individual learning. From the immense pool of available cultural variants, individuals test a number of them and then decide what to keep and what to discard. This process is very accurate but also extremely slow, given the number of available alternatives even for simple problems, expensive and sometimes dangerous.

The classical example of the costs related with individual trial and error is the one in which we try to determine whether a mushroom is edible. The safest way would be, of course, to have this knowledge hard-coded in our DNA, like many other animals. But given that nature has not equipped as with accurate enough sensors for that purpose, the possible error coming from our trial would prove to be lethal.

Rats are very good in individual learning, for they are willing to experiment with new foods and that partially explains their adaptive success (although current research (Noble and Todd, 2006) indicates that rats are also relatively good cultural learners). However, the mean birth rate in humans is not even close to that of rats, to compensate for the costs of trial and error. Other solutions are needed to allow adaptation to the environment while at the same time limiting the costs.

3.2 Epidemiological culture: Social learning

Epidemiological culture or infocopying is a much cheaper way to acquire knowledge. Epidemiological culture is what in this text we refer to simply as culture and the term is used to discriminate this source of knowledge from evoked culture. It is the knowledge, ideas and attitudes, in other words, the memes that we acquire, by directly or indirectly copying others. It is also the inheritance mechanism that allows natural selection to act on culture (see 2.1).

In the example with the mushrooms, where the cost of individual learning is unbearable, a much more efficient strategy and the one most commonly used by humans is to listen to or observe other people in order to decide which mushroom to eat. In this way, under ideal conditions, only one of us will actually have to eat a poisonous mushroom and the rest will take advantage of his discovery and further deaths will be avoided.

Another real-life example mentioned by Henrich and McElreath (2003) clearly illustrates the importance of infocopying. In 1860, Robert Burke led an expedition from the south to the north of Australia. Despite, the fact that the expedition was extremely well equipped and the participants very experienced, only one of them finally survived. They were all beaten by the same environment that local aboriginals inhabited for many years. The survivor was rescued by a group of locals and lived with them for several months. The importance of this example is to illustrate that evoked culture and individual trial and error alone cannot explain human adaptability. The survivor managed to adapt to an environment for which neither his genes nor his experiences have prepared him, by copying other people's knowledge and minimising the costs of individual learning.

3.2.1 Between individual and social learning: local enhancement

Another form of learning, observed commonly in humans and other primates and very often mistakenly considered as imitating is *local enhancement*. Local enhancement resembles imitation because it depends on social interactions but knowledge coming from it does not accumulate. When an individual is, for various reasons, following another they will end up living in the same environment. It is possible, therefore, to discover and use the same or very similar tools without anyone copying the other but simply because they both have the chance to experiment with the same materials.

Local enhancement, is observed in many primates but it can not be considered as true imitation and most importantly it does not result in accumulation of knowledge. The know-how or the effective strategy will not spread as rapidly and, more importantly, individuals will keep on reinventing it and very rarely improving it.

3.3 Why is social learning adaptive?

For a long time, social learning was considered adaptive simply because it avoided the costs of individual learning. However, this view gives such an adaptive advantage to social learning that does not explain why do we ever innovate and thus why evolution is possible.

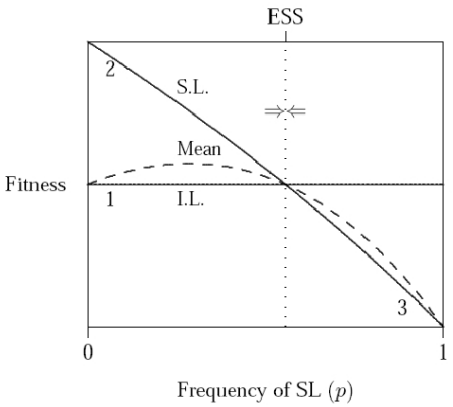


Figure 3.1: Rogers' model (1988) that shows that the fitness of social learners (S.L.) and of the mean population decreases as the frequency of social learning increases

There are two problems related to infocopying that should not be underestimated. A very influential work by Alan Rogers (1988) gave evidence for the first problem, the inefficiency of social learning to respond to a changing

environment. Using a "thought experiment" as he calls it, Rogers was the first to construct a mathematical model to show that social learning alone does not increase but rather decreases the average fitness of the population. As shown in the graph 3.1, the fitness of social learners is higher than the mean fitness of the population only when social learning is rare. The explanation for this result is that "without any individual learners, social learners cannot track changes in the environment" (Henrich and McElreath, 2003). In other words, when the environment varies either spatially or temporally, learning from others alone does not help, since the existing information for a given environment does not solve the problems of another.

Additionally, infocopying compared to individual learning also suffers from reliability, given that information acquired from others can for various reasons be less accurate than information resulting from one's own experience. This problem is known as "accuracy-generality tradeoff".

Therefore, copying requires additional psychological capabilities in order to assess the incoming information and its source (see 3.4), and "noise" due to poor judgements or the existence of defectors will unavoidably appear in a greater or lesser extent.

So, why is infocopying adaptive? Boyd and Richerson (1995) extended Rogers' work and constructed a more advanced model that nevertheless produced the same results. In the same paper, however, they also provided a solution to this puzzle: Infocopying increases the overall fitness of the population *only when exercised in conjunction with individual learning.* In this case, it allows individual learners to build on existing knowledge, i.e. acquire some knowledge from others and then expand it. Additionally, it improves the efficiency of individual learning: we use individual learning when it is cheap and we learn from others in every other case.

Furthermore, the accumulation of knowledge allowed by social learning is, according to Henrich and McElreath (2003), what differentiates humans from other primates capable of maintaining culture. Only humans accumulate knowledge generation after generation and that may be a good explanation of why we are so good in adapting and expanding. "Natural selection favors cultural learning only when the costs of developing and maintaining cultural learning mechanisms are smaller than the benefits gained by acquiring simple behaviors that could be learned on one's own"' (Henrich and McElreath, 2003)

3.4 Copying is not random

In biological evolution, apart from natural selection, another process, known as random genetic drift, highly influences the spread of genetic information. In short, random drift occurs during reproduction when some genes of each parent may or may not be passed to the offspring, thus resulting in a non-

perfect representative sampling of parents' genes. This is a random force that decides whether a genetic trait will be inherited or not.

Some researchers attempt to determine the direct analogy of random genetic drift in cultural evolution. Bentley and his colleagues (2005; 2003) constructed a model to explain the evolution of various cultural items, such as fashion, dog breeds or baby names, as a random process.

We believe, however, that random drift is present but much less common in cultural evolution. Although at the population level cultural changes may seem like random, at the individual level copying is determined by specific forces. If copying were random, then it would not increase the average fitness of the population by not increasing the average quality of information and thus it would not be adaptive. In our model, (section 4.5) we address the possibility of random copying and show that it can be adaptive only under very specific and usually unrealistic circumstances.

We don't copy information randomly. We need additional psychological capacities, rules of thumb, to determine what to copy and from what source. Some degree of randomness can still be present, because these heuristics are less than perfect and erroneous information may be copied, nevertheless.

Next, we will describe the major biases that determine those non-random forces that govern cultural evolution: content bias and two content-random biases, conformity and prestige bias. These heuristics were first identified by Richard Boyd and Peter Richerson (1985).

3.4.1 Content bias

The actual content of any cultural item is of course a major factor that influences the decision to copy it. Heylighen (1997; 1998) provides an overview of the various properties of a meme that influence its success. Some of them are:

- Simplicity: The easier a cultural item is to be understood, in other words the less effort required for understanding it, the more probable it is to persist. Additionally, coherence, i.e. how well it fits with pre-existing knowledge, also facilitates the meme's spread.

- Distinctiveness: Related to the simplicity factor, cultural items that are "distinct, detailed or contrasted are more likely to be noticed and understood, and therefore assimilated" (Heylighen, 1997).

- Utility: When an individual believes that an idea or a behaviour is likely to improve her fitness, her chances to survive and reproduce, it is highly likely that she will adopt it.

- Novelty: Finally, a new idea that will more easily attract the copier's attention has also increased chances to be copied. According to Dessalles

(2000), novelty is a crucial property of an interesting conversation among humans and individuals spend increasing amount of effort to provide it.

However, the actual content of a cultural item is not the only reason for someone to copy it, nor necessarily the most important. In a complex environment, it is very often extremely difficult and costly to assess a meme based on its content. Conformity and prestige bias, described next, reduce the costs related to meme assessment and may even provide more reliable results. These heuristics have also the effect that cultural items of arbitrary content may be propagated.

3.4.2 Conformity bias

Conformity bias, or "copying the majority", is another heuristic that is extremely useful in limiting the need to assess cultural items in our environment. According the Henrich and McElreath (Henrich and McElreath, 2003), given that the average population behaviour "implicitly contains the effects of each individual's experience and learning efforts, conformist transmission can be the best route to adaptation in information poor environments".

We may choose, for example, to enter in a crowded restaurant than an empty one, when we have no other information about the quality of either. For that reason, enterprises often advertise the size of their clientele to convince us regarding the quality of their services.

Of course, it is also quite likely that the crowded restaurant owes its rich clientele to other reasons, such as extensive publicity, which means that copying the majority is not in every case the optimal solution. As with prestige bias, that will be investigated in depth, these rules of thumb suffer from various inefficiencies, but they are adaptive because they considerably limit the costs of individual learning and content-biased copying and allow accumulation of knowledge as mentioned earlier (see section 3.3)

3.4.3 Prestige bias

Prestige bias or, in other words, copying the successful, is based on the principle that if we can copy the ideas or the behaviours of a successful individual, we are more likely to increase our own chances to succeed. Thus, using prestige bias, we manage to avoid the costs related to individual learning and the need to assess the utility of each meme ourselves.

Prestige-biased cultural learning is the subject of this research work. However, before going deeper into explaining prestige and the results of our research, we need to address one important issue related to direct infocopying, i.e. learning directly by communication or observation.

3.5 Sharing information

We already saw in section 3.3 that there are various adaptive advantages into copying knowledge and behaviour from others. What is not clear, on the other hand, is the reason for which it is also adaptive for individuals to allow information to be copied from them. Why the prestigious, in particular, give away knowledge to their potential competitors? This behaviour seems awkward especially because, in doing so, an important competitive advantage is lost. When there is competition in the same environment for the same resources, sharing knowledge not only means sharing the resources but it also means that the receiver of information is relieved from the costs related to individual trial and error.

This question is related with the general question of altruism. Even if some of Dawkins' ideas expressed in "The selfish gene"(1976) are controversial, mainly because of misunderstandings, the notion that in the core of natural evolution is the tendency of genes or organisms to propagate is by now widely accepted. In such an environment, altruistic actions such as providing information crucial for survival to others with no obvious profit have not been completely understood and explained. Charles Darwin himself saw altruism as a problem in his theory and did not manage to provide any satisfying explanation.

Next, we will review briefly the various theories that have been proposed regarding information sharing and focus on the very interesting theory by Čače and Bryson(2005) that we believe it provides an adequate explanation for the behaviour of the agents in our model.

Kin and Group selection theory

In the 1960's W.D. Hamilton formed the very influential "Hamilton's Rule":

```
c < b*r
```

According to this rule, an altruistic action can be performed when the cost c of the action to the altruist is less than the product of the benefit b for the receivers of the action times the probability r of the receivers sharing the same genes with the altruist.

The general idea behind kin selection is that genes may manage to propagate by acting in favour of other organisms that share the same genes. Group selection theory extends the same idea, by arguing that several altruistic actions can be explained as aiming to increase the fitness of a whole group rather than that of an individual.

Reciprocal altruism

According to a widely supported theory, sharing information is actually not altruistic. In their paper about prestige, for example, Henrich and Gill-

White (2001) argue that prestigious individuals give away information in exchange for deference. Thus, the prestigious enjoys preferential treatment by having access to food and other resources, greater mating opportunities and often by being excused from unpleasant tasks. Henrich and Gill-White refer to various ethnological evidence to support their view. Hawkes, for example, reports evidence according to which Ache males "more frequently overlook sexual liaisons between their wives and highly skilled hunters"(Henrich and Gil-White, 2001).

Strong Reciprocity

The theory of reciprocal altruism suffers from the existence of defectors, i.e. individuals who may accept the beneficial actions of others without ever returning the favour. Strong reciprocity is another recent theory attempting to explain altruism. In brief, strong reciprocity refers to the tendency of individuals to cooperate but at the same time to punish those who do not cooperate.

Gintis et al (2003) argue, based on behavioural experiments, that strong reciprocity is an evolutionary stable strategy. In another paper, Bowles and Gintis (2004) argue that "many humans have a predisposition to punish those who violate group-beneficial norms, even when this imposes a fitness cost on the punisher". They have also developed an agent based model to show that strong reciprocity is a major factor in sustaining high levels of cooperation within a social group.

3.5.1 Sharing information can be adaptive without reciprocity

Most of the above theories treat altruistic behaviour or sharing information, in our case, as simply one part of an exchange. They argue, that in order for sharing information to be adaptive, the copier must in one way or another reciprocate and may be punished otherwise.

Čače and Bryson (2005), on the other hand, have recently developed the idea that exchanging knowledge for other tangible or intangible goods does not have to be the only explanation for altruistic behaviour. According their theory, "sharing knowledge can be adaptive for its own sake". They have build an agent based model in which agents compete for food resources, for the most nutritious of which special knowledge is required in order to be consumed. In their model there are two types of agents, those who share their knowledge, and those who remain silent.

In the model, there is no implementation of deference. The agents who share their knowledge have no direct profit from doing so. In such circumstances, most of the theories mentioned so far predict that the loss of the competitive advantage should lead the talkers into extinction.

The experiments of Čače and Bryson, however, had totally different results: it is clearly shown that even if talking agents are initially the minority in the environment, they finally manage to propagate and prevail. Their explanation for this result is very enlightening: the act of sharing knowledge can itself be inherited and thus talking agents end up living in an environment where there is high probability to learn something new. On the other hand, silent agents, even if avoiding the costs of sharing, they live among relatives with the same behaviour and thus the chance to learn something new is much more limited.

We, thus, share our knowledge, because this behaviour contributes to a formation of a social environment where information is freely available. This result is very important for our research. In fact, the model we have implemented to investigate prestige and which is described next, is based on the existence of an adaptive pressure to share knowledge with minimal or with no deference at all, and the results of our experiments verify Čače's and Bryson's findings.

Chapter 4

The role of prestige

Prestige is defined as "widespread respect and admiration attracted through a perception of high achievements or quality"(Fowler et al., 2004). In their excellent paper, Henrich and Gil-White (2001) define prestige as "noncoerced, interindividual, within-group human status assymetries". Prestigious is the person who is *believed* by some of the members of his community to possess better skills and/or increased knowledge on issues of general interest, issues directly or indirectly related with survival and reproduction.The prestigious individual enjoys the respect of the others, is listened to and, thus, influences the decisions and choices of others.

Henrich and Gill-White (2001) are correct to distinguish prestige from dominance. Although both share some common properties, such as the fact that both dominant and prestigious individuals are usually high in the social hierarchy and enjoy special privileges, the prestigious are not exerting any physical or other force on others. The special treatment prestigious people enjoy is freely conferred by others and although there are various explanations for this (see section 3.5), fear is certainly not the principle among them.

The discrimination between prestige and dominance is a very important one for our research. Prestige is a mechanism that allows knowledge spreading and thus cumulative cultural evolution. On the contrary, although dominance is a also a mechanism that results in the formation of social hierarchies, it is nevertheless unrelated to knowledge sharing. The dominant individual is usually obeyed and not respected (although there are cases of both obeyed and respected individuals) and his ideas, attitudes and behaviours are rarely copied, unless, of course, his ability to dominate is considered an important skill in a certain social setting. In addition, prestige bias usually results in less *formal* social hierarchies than dominance and, as we will see in our model, the hierarchy, although probably present, can not be clearly described.

In order to investigate the way prestige bias works and its effect in knowl-

edge spreading we have developed an agent based model. In the following sections we describe this model, in order to study and analyse prestige bias in depth.

4.1 The model

The world where our agents "live" consists of 200 by 200 squares on a torus space. Agents that walk off one edge, will reappear in the opposite edge. Each square may contain only one unit of food and may be occupied by one or more agents. The simulation proceeds in discrete steps, in which food may grow in squares and where each agent performs a set of actions (see 4.1.2). In all the experiments presented here the simulations were allowed to run for 8000 steps before termination, by which point stable equilibria have usually been reached.

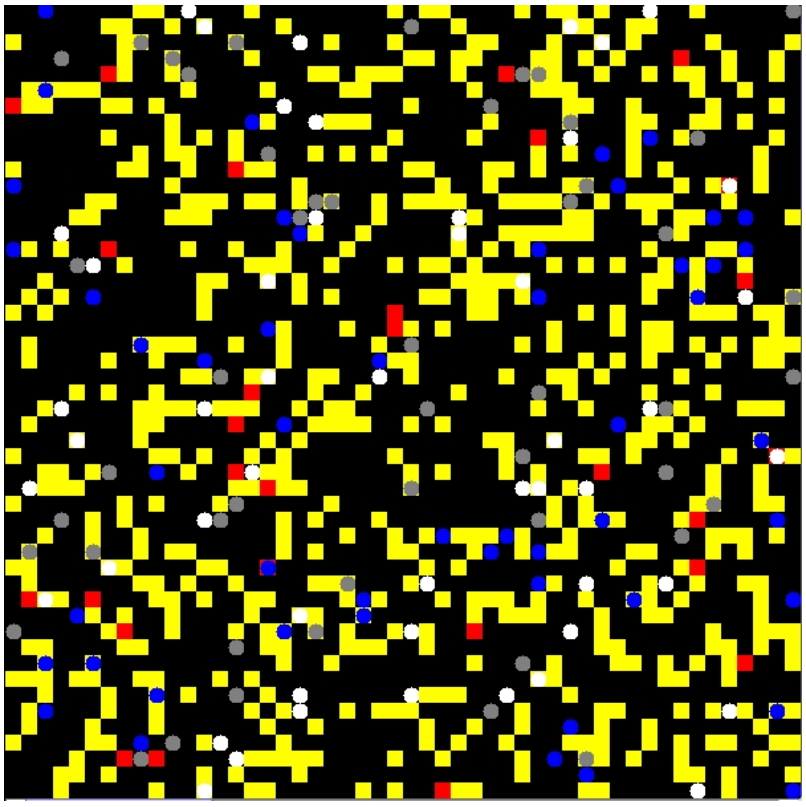


Figure 4.1: An image of a part of our world a few steps after the simulation is started. Coloured squares represent food sources (yellow are nutritious, red are harmful) and coloured circles are agents (white never copy, grey copy anyone, blue copy the most prestigious)

4.1.1 The environment

In our model, there are two types of food: those that are nutritious and will increase the energy level of the agent that consumes them (represented with yellow squares) and those that will decrease the energy once eaten (represented with red squares). Unhealthy food items are approximately 1/3 of the total food resources and the rest are nutritious. More specifically, in most of the experiments that will described in the next sections, unless otherwise mentioned, there are 60 different types of food in the environment of which 42 are nutritious and 18 are unhealthy.

All nutritious foods add the same amount of energy and all unhealthy foods subtract as much as each other but nutritious food items add more energy (+12 units) compared to energy subtracted by unhealthy foods (-6 units). At every step, a random food item may grow in a certain square with a given probability, such that, on average there is food in 70%-80% of all the squares.

These settings were selected, after initial testing, specifically so that the environment is viable and it allows a constant population of approximately 3000 agents (although this also depends on other parameters, such as breeding, for example).

It must be understood here that the naive environment with good and bad foods is simply a metaphor, an abstraction that allows us to focus on certain interactions while restricting other information to a minimum. As we will see next, the agents possess no sensoring mechanism to determine the energy value of a food item. We imagine food items as abstractions of more complex entities, such as ideas or beliefs, where individual learning is difficult and so costly that other mechanisms must be employed to allow survival. We call the set of beliefs an agent possesses"ideology" because we imagine it as a strategy a human may employ in order to succeed in complex tasks such as survival and reproduction.

4.1.2 The agents

At every step, agents perform the following set of actions:

Copy a random set of a neighbour's beliefs. This step is optional and the exact way in which it is performed depends on the agent's genotype, as we will see next. The number of beliefs that will be copied in any case is, by default, 1/3 of the total food types. Also, the region "visible" by an agent, the agent's neighbourhood, is an hexagon of 18 squares centred at this agent.

A minimum degree of reciprocity is implemented: after copying, agents invite their model to eat first. The effect of this kind of deference is not very important and as we will see, removing it completely does not alter the results produced, in any significant way.

Consume the food that exists in the square they occupy (if not empty), provided that they *believe* that it is nutritious and that they have not reached the maximum level of energy (100).

Reproduce asexually. The probability of breeding heavily depends on the parent's energy: the fitter the agent the more probable giving birth and agents below certain threshold (50) cannot breed. Twenty per cent of the parent's energy is transferred to the offspring that is placed in the same square as the parent, when born.

Move in a random manner to an adjacent square

Die if their energy level is below 0 or their age is above a certain limit (50 by default).

Additionally, a unit of each agent's energy is subtracted and it age incremented at every step.

Each agent is born with a set of randomly generated beliefs about the edibility of each food type. This ideology is not at all related with its parent's beliefs. Since, as mentioned before, foods are abstractions for complex cognitive entities, we suppose that there is no genetic, no Lamarckian inheritance of beliefs. The only way for agents to change their beliefs about the edibility of food resources is by communicating with neighbouring agents. The copying behaviour of agents discriminates them into three categories:

1. Agents that never copy other agents (represented as white circles in figure 4.1).

2. Agents that, at every step, copy some of the beliefs of a random agent in their environment (represented as grey circles).

3. Agents that copy the most prestigious agents among their neighbours (represented as blue circles). Determining the most prestigious is a crucial function in our model and is analysed in depth in section 4.2.

Given that, as described above, there are on average more nutritious foods in the environment and that they add more energy than harmful foods subtract, agents who eat most or everything they will, in most cases, survive. Therefore, however complex the environment is, agents of the first genotype, who, on average, eat half the food types, will, usually, survive unless outcompeted by another type of agents.

The copying behaviour remains static throughout an agent's life and it is the only thing that is inherited during reproduction. When the simulation starts, the world is populated with an equal number of agents of each genotype.

Individual learning is only indirectly implemented in our model: we may imagine that immediately after the agents are born, they are experimenting

with food types, in order to acquire their initial beliefs about food edibility. This is similar to the method presented in Čače's and Bryson's model (2005). In all the graphs that appear in this text, we can observe that the average number of correct beliefs at the beginning is approximately half the number of different foods, as expected from the random distribution.

Apart from inheriting the copying behaviour, agents have no other special relationship with their parents, other than the probability that their parent may be considered as more prestigious and thus is more likely to be copied, as long as they both live in the same neighbourhood. However, no special care has be taken to ensure that. Contrary to what is commonly believed, research by behavioural geneticists indicates that correlations between parents and offspring are mainly the result of inherited genes and not infocopying (Richerson and Boyd, 2005, p.36).

4.1.3 Experiments and graphs

In the following sections, we analyse the experiments conducted to investigate prestige-biased infocopying and the results produced by our model. The exact experiment conditions can be found in appendix C. In the main text, the results are presented in the form of graphs. These graphs are representative of each experiment. The exact numbers and the graphs produced by each of the ten simulations for each experiment can be found in the accompanying CD-ROM in the folder "ResultData". There, in the subfolder "Data" there is a set of .xls files, one for each experiment, where, apart from the data, there are also calculated mean values and standard deviations.

The graphs that will be presented in the main text are mainly of two kinds:

- "Number of agents per type": A plot with three curves, one for each genotype, showing the number of agents per time step.
- "Average correct beliefs per type": A plot with three curves, one for each genotype, showing the mean number of beliefs that are correct for all agents of each genotype. A correct belief is either the belief according to which a nutritious food item should be consumed or the one according to which a harmful food item should be avoided. The rapid fluctuations observed before an agent type becomes extinct are insignificant and are due to the very small number of agents of that genotype that are left in the simulation.

4.2 Finding the prestigious

The process by which prestige is attributed and prestige hierarchies are built within a group is itself Darwinian in nature. We have:

- Variation: every individual is potentially a model to be copied with different knowledge, beliefs, attitudes and behaviours.
- Competition: Only one of the models will be selected by the other agents at each step.
- Selection: The copier selects the model to imitate, based on various heuristics such as health, age, sex, physical appearance etc.

The last point, selecting the model to copy, is a complicated issue. It is basically based on the success an individual *demonstrates* to the community as well as the pre-existing beliefs. For example, the hunter who repeatedly returns with the most numerous prey, will gain more prestige and thus her knowledge and behaviour will be more likely to be copied by others, especially in societies where hunting is the primary source of food.

A variety of heuristics are used to attribute prestige. Berger et al. (1980) argue, based on various experiments, that individuals that first meet each other, quickly decide on the prestige order that is "correlated with external status differences" such as sex, occupation, race, physical appearance, education etc.

In our model, agents of the third genotype attempt to determine and compare the prestige of their neighbours and copy the beliefs of the most prestigious among them. Determining the most prestigious is accomplished by a function that takes into account

- the energy level,
- the age and
- the number of offspring

of the agent under consideration.

The degree to which each of the above properties is relied upon depends on three constants E, O and A, each associated with a property. Thus, the function used to determine prestige is:

prestige = E * (energy / max_Energy) + A * (age / max_Age)
+ O * offspring

which, given the default values used in all the experiments, becomes

prestige = E * (energy / 100) + A * (age / 50) + O * offspring

We have conducted a series of experiments with different combinations of the above constants in order to determine the importance of each of the attributes that reveal prestige.

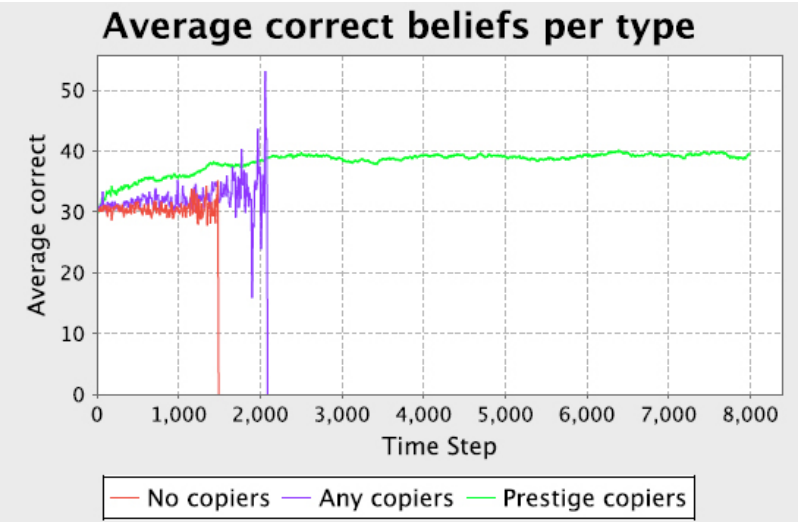


Figure 4.2: Energy as the only indicator of prestige. The plot presents the mean correct beliefs for each genotype per time step.

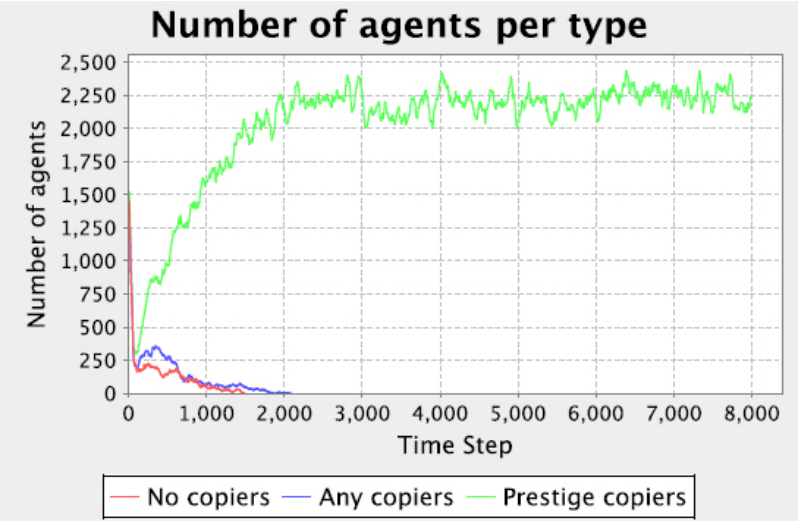


Figure 4.3: Energy as the only indicator of prestige. The plot presents the number of agents of each genotype per timestep.

Energy

Energy is the only direct and usually objective means to evaluate an agent's success. Graphs 4.2 and 4.3 show that agents that copy the neighbours with the highest energy value, manage to prevail over other genotypes. However, in real circumstances, detecting an individual's energy, i.e. health or fitness, is rarely a straight-forward process and, even in our model, there are cases

when judging an agent from its energy level alone is not the optimum strategy. We will see later that combining energy with other indicators produces better results, than relying on energy alone.

Offspring

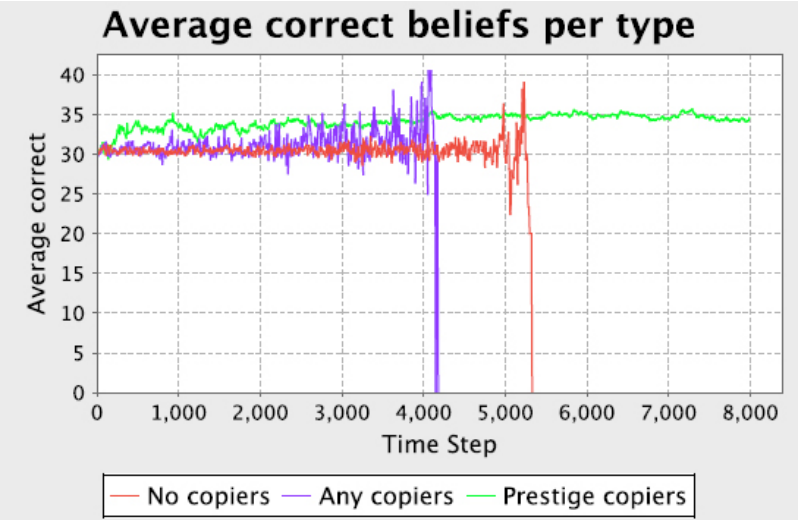


Figure 4.4: Offspring as the only indicator of prestige

The number of offspring an agent has produced, is, at least in the way implemented in our model (where an agent breeds only when its energy level is above a certain limit), an indicator similar to energy. In fact, compared to energy, it is an indirect and less reliable method to assess success. The only important fact that may be inferred is that an agent with many children manages to sustain a constant level of energy above a certain threshold, but that heavily depends on the energy required for breeding. We could, of course, change the parameters that affect breeding to require higher level of energy and rely less on chance. We predict that, in that case, copying agents with the most children would essentially have the same (positive) results as combining energy and age when attributing prestige (see figure 4.7(a)).

However, relying only on the number of offspring, even if it is less successful than relying on energy, it still allows agents to outcompete agents of other genotypes and survive, as can be seen in graph 4.4. Moreover, graph 4.5 shows that the performance of agents who assess prestige based equally on energy and on the number of offspring is similar to that based only on energy, as expected according to our analysis.

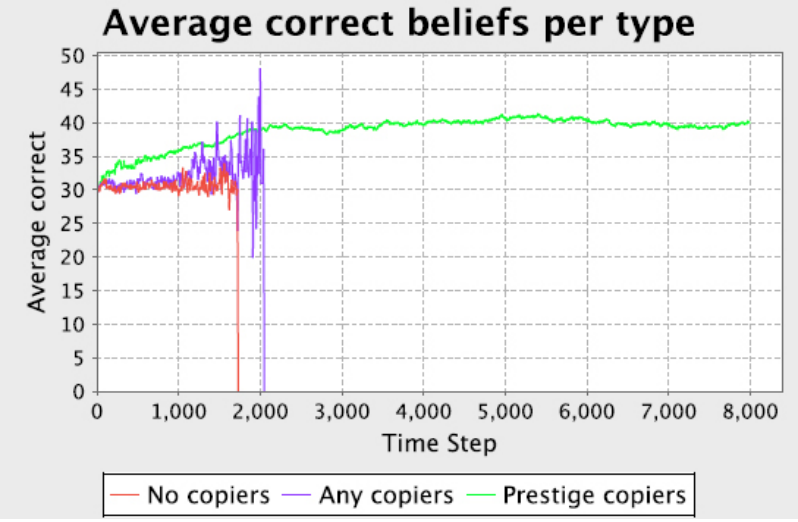


Figure 4.5: Energy and offspring as a combined indication of prestige

Age

Age is also taken into account when assessing potential models. Older individuals are usually attributed higher prestige. That is often true even if when the skills of the elders are only communicated and not actually demonstrated. Henrich and Gil-White (2001) refer to a survey in 1945 by L.W. Simmons according to which in the majority of the simple societies studied, individuals show respect, deference or obeisance towards the elderly and in some cases "the high status of the elderly can be inferred", despite their physical weakness. This is probably due not only to the fact that older individuals have managed to survive for a relatively longer period of time but also to the inferred experience individuals have accumulated at late stages of their lives, leading them, thus, to possess knowledge of improved quality.

In our experiments, age, has proven to be the most interesting of the indicators studied. Agents that live for a relatively large number of steps — years, are likely to have a set of beliefs that even if it doesn't maximise their energy, it still allows them to survive. However, as can be seen in graph 4.6, judging success on age alone is not an adaptive strategy. Copying the beliefs of the oldest individuals, even if it is a rational strategy, for reasons already explained, it turns out to be so inaccurate that it finally does not cancel the costs of infocopying and fails for the same reasons that random copying fails (see 4.5).

Is, therefore, the tendency to listen to the elders a maladaptive strategy? We have conducted another set of experiments to determine the adaptiveness of taking age into account in combination, this time, with the other indicators.

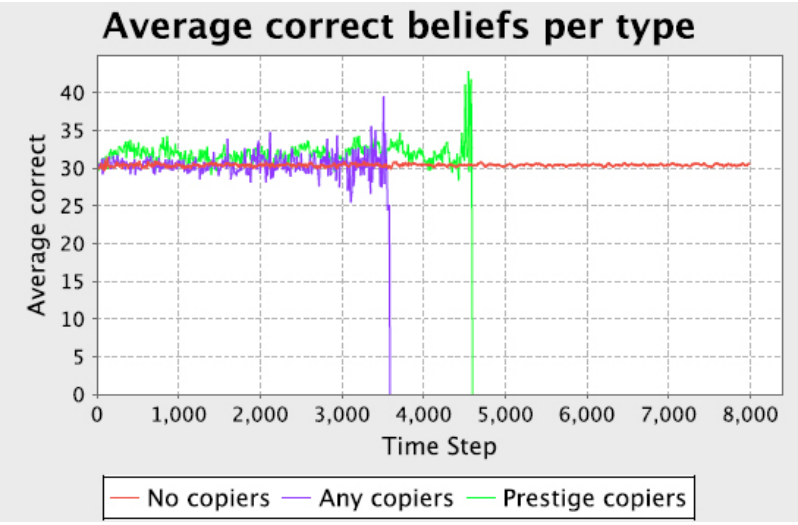
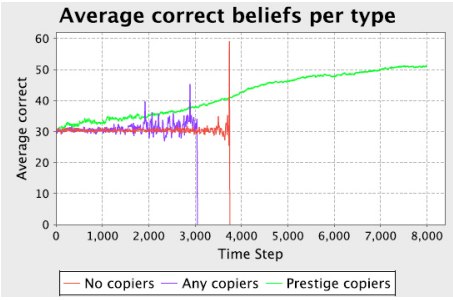
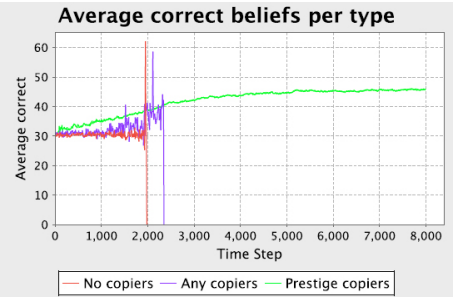


Figure 4.6: Age as the only indicator of prestige



(a) Age and energy



(b) Age and number of offspring

Figure 4.7: Age as a second indicator of prestige. When combined with either energy or number of offspring, taking age into account is improving the average quality of knowledge (compare with figure 4.6). The step in which other genotypes become extinct is volatile and it is due to the various sources of randomness and noise in the model.

The results, as can be seen, were enlightening. If, age is combined with one of the other indicators, either energy or offspring, it produces the more accurate results than any of the indicators alone. Age means adaptiveness for a number of years and, as an indicator of prestige, it cancels out the possible errors from assessing prestige based on energy (directly or indirectly) alone. Thus, age, although very error prone as an indicator when blindly trusted, it reveals the long-term performance of an agent and thus improves decisions when combined with other indicators of prestige.

4.2.1 Conformity in prestige hierarchies

Whenever information required to assess the skills of a potential model are not available, it has been argued by many researchers that a successful alternative is to "follow the majority", to copy the one that is considered the most prestigious by the rest of the community. "...the size and lavishness of a given model's clientele (the prestige) provides a convenient and reliable proxy for that person's information quality"(Henrich and Gil-White, 2001).

Models who seek more clients, due to reasons that will be analysed later (see 3.5), will try to broadcast the fact that they are considered prestigious by various means. Medals and awards that are publically displayed are such means of broadcasting status. Honorary titles, such as "Doctor" or "Professor" can also be thought as ways to broadcast prestige.

In order to investigate the role of conformity in model selection, we have added an additional parameter, a conformity field. Each time an agent is being copied, the model's conformity field is being incremented. Then, when another agent is assessing the model's prestige, it takes into account the number of times this model has been copied by other agents. Thus, the function to determine prestige now becomes:

prestige = E * (energy / max_Energy) + A * (age/max_Age) +
O * offspring + C * *timesCopied*

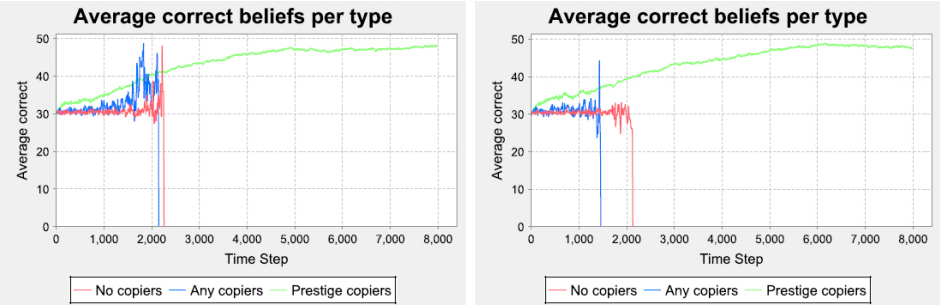
where C is a constant that determines how much agents rely on conformity when attributing prestige.

We have conducted a set of experiments to determine the effect of conformity bias in selecting the model to copy. In each experiment conformity was relied upon at a different degree. Graphs 4.8 show the results of those experiments.

As we can see, contrary to what we may have expected, our results show that conformity bias in model selection is either insignificant when associated with a very low constant or it is, in fact, deteriorating the average quality of knowledge.

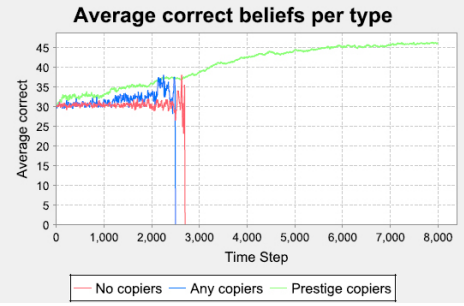
We could be tempted to support the seemingly absurd argument, that conformity is maladaptive. Of course, this is not the case. Apart from common sense, there are numerous field observations and experimental results that indicate that "copying the majority" is an adaptive strategy. In addition, here, we are referring to conformity in model selection and there are other uses for conformity in social learning, e.g. copying the mean behaviour or belief of a population. Our result is interesting, however, because it indicates that conformity bias in model selection is adaptive under certain circumstances that the construction of our model does not satisfy.

We believe that the most important explanation for the above result is that, in the way our model is constructed, *better indicators of prestige always exist*: As we said, following the majority is a heuristic useful when no other

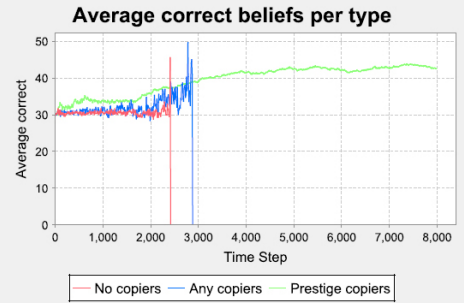


(a) No conformity bias

(b) Conformity = 0.005



(c) Conformity = 0.008



(d) Conformity = 0.2

Figure 4.8: Conformity bias in model selection. In our model, conformity bias deteriorates the performance of agents who copy the prestigious.

reliable information is present. In our model however, any decision based on the available indicators (energy, offspring and age in any combination) is relatively more accurate than a conformity biased decision would be. An indication that supports this argument is that, as can be seen in graph 4.9, combining the age of an agent with the times this agent has been copied provides better results than relying on age only, that as we saw earlier (see graph 4.6) is not adaptive. Thus when age is the only information available, conformity is, in fact, improving the quality of the decision. In real circumstances, conformity bias is adaptive because assessing a model's success is usually a very complicated issue, for which little information is available and conformity is, then, a useful guide.

Since conformity bias is not the subject of our research, we will simply indicate that conformity is simply a rule of thumb, useful under certain conditions but misleading under others. Referring to an example used earlier (see 3.4.2), entering a crowded restaurant may be a reasonable choice only if we can safely assume that we and the other clients have no fundamental

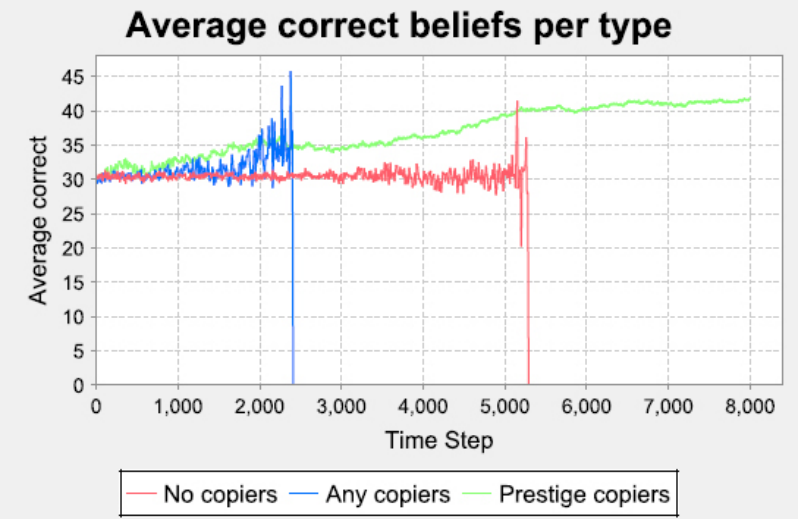


Figure 4.9: Age and conformity combined to attribute prestige. Although conformity is usually deteriorating the performance of agents, in this case, it produces better results than relying only on age (compare with graph 4.6)

differences regarding our eating habits, otherwise a vegetarian could easily end up in a steak house.

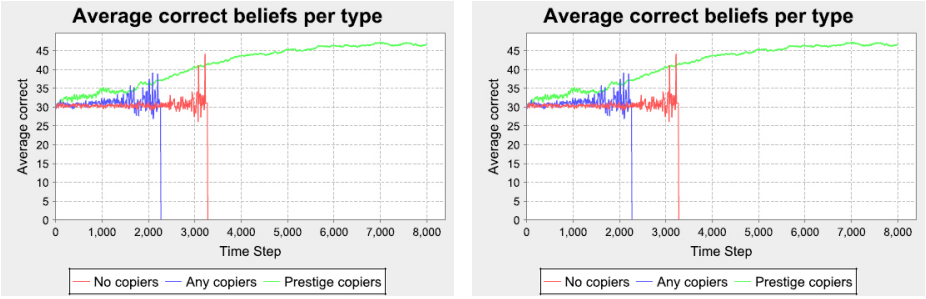
4.3 Prestige speeds up information propagation

A very important side-effect of prestige bias is the fact that cultural items, memes, are propagated very quickly, especially when they improve the survival and reproductive chances of their carriers. However, since content is not taken under consideration here, the selection and propagation of information is achieved in an indirect manner. Agents copy beliefs, not depending on their quality but on the agents that hold them. This indirectness is useful, especially when the task of assessing the quality of information is very costly or very complicated. In section 4.8.2 we expand on the issue of information propagation and also analyse the problems related to acquiring beliefs and attitudes in such an indirect manner.

4.4 Prestige bias is adaptive

The results of our experiments, as the graphs in the previous sections illustrate, indicate that prestige bias is adaptive in most cases. Agents that use any rational means to attribute prestige and copy the beliefs of the most prestigious possess better than average knowledge, increase their survival and reproductive chances and, finally, manage to outcompete agents

of other genotypes.



(a) With deference

(b) Without deference

Figure 4.10: Prestige-biased infocopying is adaptive even when the copiers do not reciprocate

Additionally, similar to Čače's and Bryson's model (2005), prestige-biased infocopying is adaptive even without deference. Despite the fact that in most of the simulations conducted so far there is a minimum degree of reciprocity, since the copier allows his model to eat first, we argue, in accordance with Čače and Bryson, that infocopying is adaptive even when the copiers do not reciprocate at all. A comparison of the graphs in figure 4.10, where deference has been switched off, is another indication supporting this argument. We can see clearly that our minimal inclusion of in our model has minimal or no impact on the results produced.

4.5 Can random copying be adaptive?

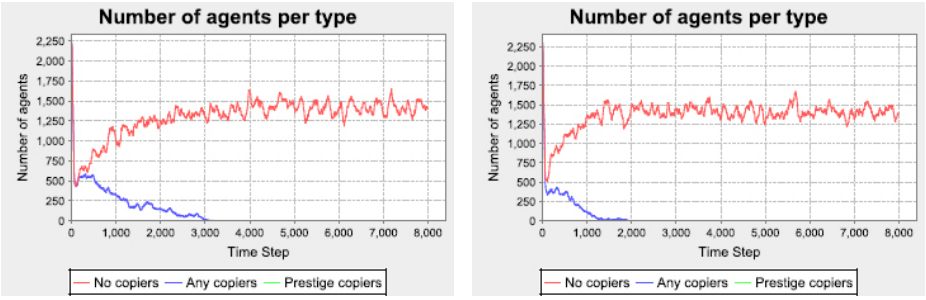
In section 3.4 we discarded the idea that infocopying is random, on the grounds that if random choices were the norm, the average fitness would not increase and, therefore, infocopying would not be adaptive.

In order to verify the above, in our model, agents of the second genotype will, at every step, randomly choose one of their neighbouring agents and copy its beliefs regarding food edibility. We are interested in the average performance of this type of agents compared to those that copy the most prestigious neighbour and more interestingly to those agents that never copy other agents.

In the first case, the results of all the experiments set forth so far, are not at all surprising. Agents who use any rational function to determine prestige (such as taking into account the energy, age and/or number of offspring) will in most cases outcompete agents who copy randomly. Regardless of the indicators used to evaluate their neighbours, it is certain that as long as those indicators have any degree of rationality they will always lead to knowledge

improvement compared to copying without making any assumptions about the quality of the knowledge copied.

Also interesting, though, is the comparison between random_copiers and no_copiers. Agents who never copy will, on average, always be correct exactly half the times. In every graph shown so far, we can see that no_copiers are always correct about half the food items, on average. Therefore, the comparison between agents who copy randomly and those that never copy is similar to the question about whether random copying can ever lead to better than average quality of knowledge.



(a) With deference

(b) Without deference

Figure 4.11: Number of agents per genotype, without the presence of prestige_copiers

From the experiments conducted, it turns out that random copying is in most cases less successful strategy than never copying. For example, in graphs 4.6, 4.13(d) and 4.14(d), where agents who copy the prestigious became extinct, it was the no_copiers who finally prevailed and not the random_copiers or even a mixture of both. To further illustrate this result we have conducted another set of experiments where prestige_copiers have been removed. Graph 4.11(a) shows again that random_copiers are outcompeted by no_copiers. Perhaps, we would expect this not be true, since randomly copying anyone should on average lead at least to the same outcome as never copying. This is not the case, due to two factors:

Copying does not come absolutely for free: Although there are various views regarding the benefits for the model, it is true that, at least to some degree, the model enjoys preferential treatment from those that copy her. This preferential treatment in our model is implemented by allowing the model to eat before the copier. Agents who never copy will never suffer the cost of allowing someone to eat first and will thus have a relative advantage compared to agents who do copy.

The copier will anyway eat second: Even if we remove the implemen-

tation of deference, mentioned above, as we can see in graph 4.11(b), random_copiers will still be outcompeted worse than no_copiers. This is because the copier will always learn something that the model already knew and probably already exploited. In other words, copying means that we always come second. If we learn that we should be eating a type of food, there is an increased chance that the agent who taught us about this food has already consumed it, at least in the immediate environment.

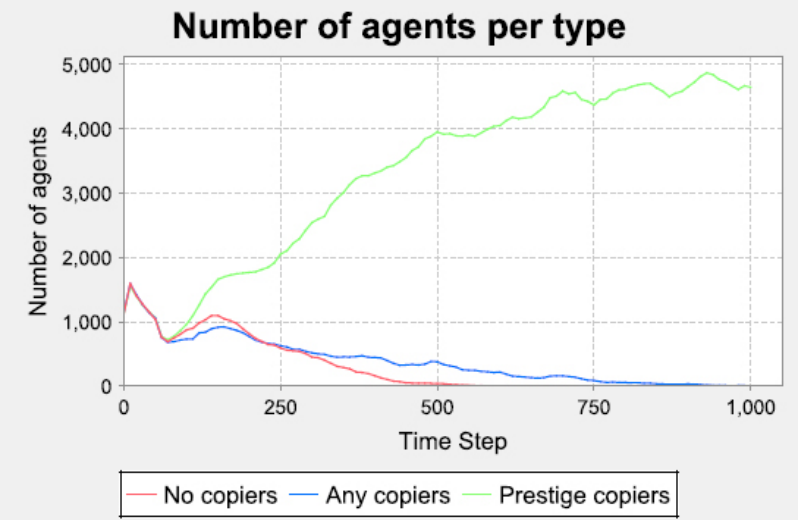


Figure 4.12: Performance of agents who copy randomly in "simple" environments. Random_copiers outcompete no_copiers only when there is very high quality of average knowledge in the environment

There is, on the other hand, a case in which agents who copy randomly will outcompete those who never copy, even if the costs mentioned above are still present. In order to study this possibility we have conducted another experiment where only 10 different food types can be found in the environment.

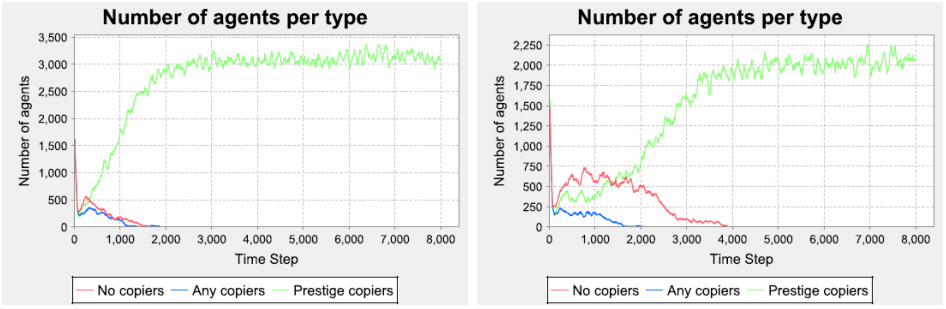
Graph 4.12 shows that in this case, random_copiers manage to outcompete agents who never copy. This is not happening because copying randomly is more successful in simple environments. Random copiers outcompete no_copiers in numbers and possess improved knowledge, only when prestige copiers clearly constitute the majority of the population and their average knowledge is close to perfection, i.e. the majority of the population is correct about what foods are safe to consume. This means that only when the average quality of knowledge in the environment is high enough, can copying randomly be a successful strategy. In those circumstances, an agent who copies randomly has an increased chance to copy from an agent

who possesses better than average knowledge.

However, in real circumstances, it is very difficult if not impossible to define "perfect" knowledge and even more difficult to determine when this perfection has been reached. Therefore, contrary to what some researchers seem to suggest (Bentley et al., 2005; Hahn and Bentley, 2003), and as analysed earlier (see 3.4), random copying can actually never be a successful strategy.

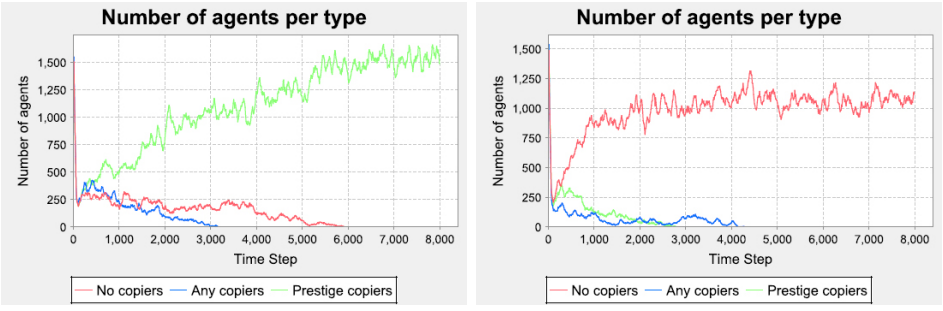
4.6 Prestige bias in complex environments

Agents who copy randomly are, of course not the only ones to be affected when the environment becomes more complicated. We have conducted another set of experiments, each with a different number of food items, to determine the adaptivity of prestige bias in increasingly complicated environments.



(a) Food types = 40

(b) Food types = 80



(c) Food types = 120

(d) Food types = 160

Figure 4.13: Performance of prestige-biased infocopiers in environments with variable complexity

As can be seen from graphs 4.13, agents who copy the prestigious, per-

form well in most environments. As expected, the "simplest" the environment the faster they manage to outcompete other genotypes. However, as shown in the graph 4.13(d), in especially complicated environments (more than 160 different food types), agents living for only 50 steps do not manage to acquire the required knowledge in order to adapt and are outcompeted by agents who never copy[1].

4.7 Prestige bias in variable environments

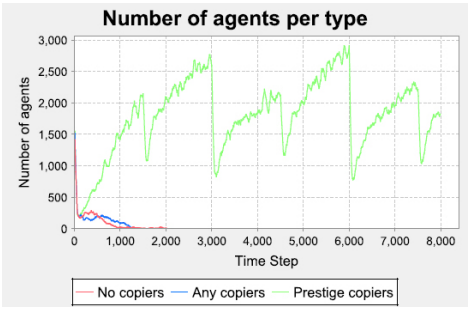
Another interesting question about prestige bias, is its adaptiveness in constantly changing environments. The issue of infocopying in variable environments has been a subject of wide research. As we saw in section 3.3, Alan Rogers has shown that in changing environments individual learning is especially crucial because without it, infocopiers cannot keep track of the changed environment. Moreover, accumulation of knowledge is advantageous only if the behaviours and attitudes that are culturally inherited are still as useful for the new generation. If the latter inhabits a fundamentally different environment then the accumulated knowledge is not only useless but, in many cases, misleading and potentially dangerous.

In this set of experiments, every n number of steps all the harmful food items become nutritious and 1/3 of the nutritious foods become unhealthy to consume.

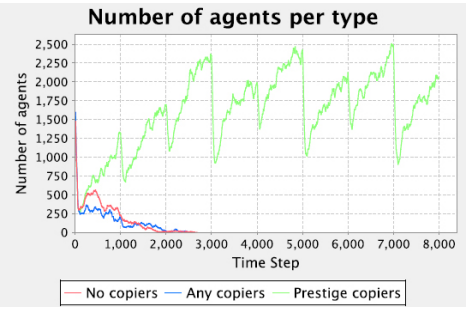
We can see in all the graphs, that each time the environment switches state, the number of agents who copy the prestigious is greatly reduced. This happens exactly for the reason mentioned above: the new environment renders the accumulated knowledge essentially misleading. Thus, for a significant number of steps, agents of the third genotype continue copying agents whose beliefs are utterly wrong in the new environment, even if their energy and other indicators used to assess prestige are still relatively high.

However, given that, individual learning is, even indirectly, present in our model (each agent is born with a random set of beliefs - see 4.1.2), if there is enough time until the next environment change, prestige bias (or infocopying, in general) proves to be an adaptive strategy. Only when the environment switches state extremely often (more than once every 2-3 generations — see graph 4.14(d)) do the the misleading inherited knowledge combined with costs related to infocopying, in general, outweigh the benefits and finally lead prestige copiers to extinction.

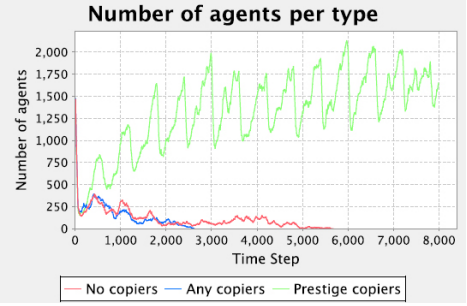
[1] As already mentioned in section 4.1.2, however complicated the environment in our model is, nutritious food items are always more plentiful than harmful ones and provide more energy when eaten, compared to energy subtracted by unhealthy foods. Therefore, agents who acquire a random distribution of beliefs about food types and never copy other agents manage to survive in complicated environments because two out of three times they choose to eat, they will be correct and additionally they are spared from the costs of copying



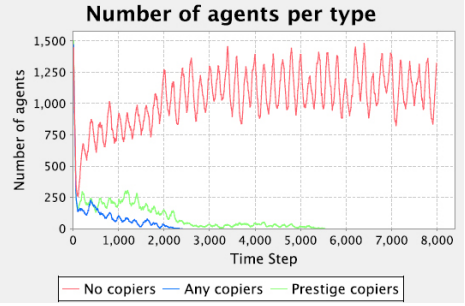
(a) Switch state every 1500



(b) Switch state every 1000



(c) Switch state every 300



(d) Switch state every 100

Figure 4.14: Performance of agents in variable environments.

In the next section we will see that, even in stable environments, prestige bias has flaws that provoke certain undesired effects.

4.8 Prestige bias is not perfect

"Copying the successful", similar to conformity bias, is a rule of thumb, a heuristic that is not always returning optimum results. Infocopiers do not directly evaluate the quality of the information acquired. They simply assume that someone with high levels of energy, who has managed to survive for many years and/or has produced many offspring is highly likely to be worth copying.

If the required knowledge was about something as simple as various types of foods (other than mushrooms!), a more direct method of making decisions such as content bias or even trial and error might prove to be more efficient than prestige bias. However, as we saw earlier (see 4.1.1), food types in our model are abstractions for more complex entities. In a complex environment, adopting the most successful attitude is most often a complicated task and,

thus, direct assessment is very costly and heuristics such as prestige bias are very useful.

This indirect method of acquiring knowledge is, of course, not flawless by any means. Next, we will see two important problems that affect prestige bias and are related both to model selection and the cultural items that are copied.

4.8.1 Model selection is imperfect

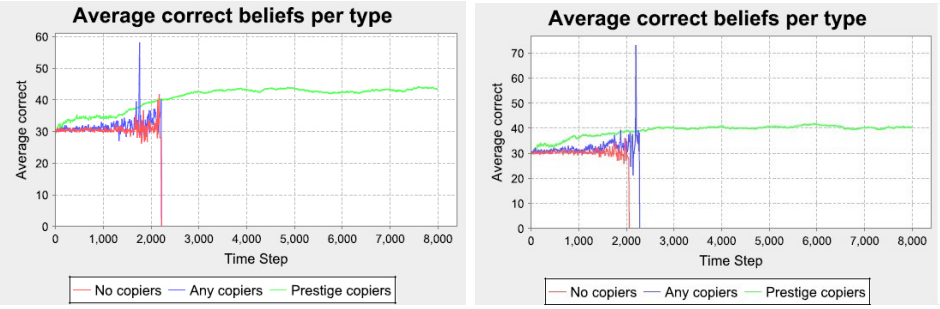
As we saw, the effectiveness of prestige bias in acquiring the correct information depends on the ability to successfully determine the most prestigious individual in the region. In our model, this detection depends on three parameters but this is just a simplification, that facilitates our study. Under real circumstances, the ability to detect the prestigious is a very important trait, depending on much cultural information that someone may or may not possess. According to Henrich and Gil-White: "natural selection favors improved learning efficiencies, such as abilities to identify and preferentially copy models who are likely to possess better than average information. Moreover, selection will favor behaviors in the learner that lead to better learning environments, e.g. gaining greater frequency and intimacy of interaction with the model, plus his/her cooperation"(Henrich and Gil-White, 2001).

In our model, for example, it is quite often that someone copies an agent possessing less than average quality of knowledge, judging mainly from its energy, that may be high if the model is recently born and inherited this high energy from its parent. The opposite, i.e. failing to detect a prestigious individual, can of course also occur, if, for example, the prestigious is, for various reasons, failing to demonstrate his superior quality of knowledge.

Another problem with prestige bias, is the possible existence of defectors, individuals who may manage to imitate the characteristics of those genuinely prestigious. Cheating may achieved, for example, by lying or presenting false artifacts such as forged medals, awards and other items that usually denote prestige. Cheating, in this case, may be difficult, most of the times costly and since the benefits to the prestigious are not direct, as both Čače's and Bryson's (2005) and our model indicate, then cheating is not even beneficial.

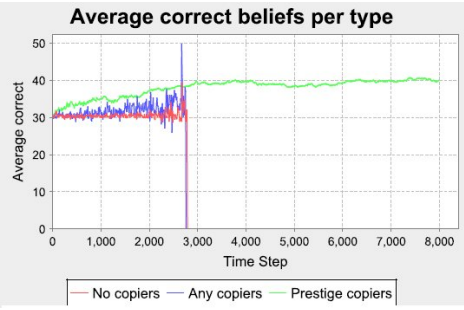
Whenever it occurs, however, copying from a cheater or someone mistakenly believed as more prestigious will usually lead to deterioration of the quality of information someone possesses. Although it is difficult to determine exactly how often and to which extent this takes place, we have added some noise in the function used to determine the most prestigious.

As we can see, noise at any rational level, does not change the fact that prestige biased infocopying, even with the danger to fail, from time to time, to determine the prestige level of the other agents, is nevertheless overally an effective strategy, especially in complex environments. Of course,

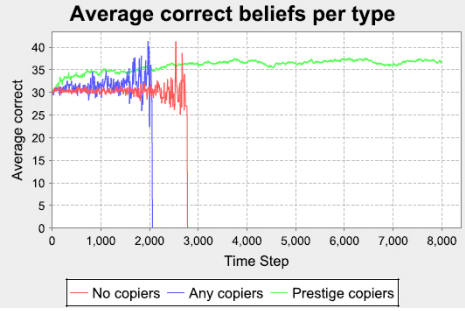


(a) Noise = 0.1

(b) Noise = 0.3



(c) Noise = 0.5



(d) Noise = 1

Figure 4.15: Performance of prestige-biased infocopiers using a "noisy" function to determine prestige (the average prestige level is normally between 0 and 1)

the better the detection function the faster prestige_copiers dominate their environment. We can see in the graphs 4.15, that as the noise level increases, the performance of the agents that copy the prestigious decreases from an average of approximately 43 correct beliefs down to approximately 36 (out of 60, in total) but, in any case, they manage to outcompete agents of other genotypes.

4.8.2 Generality bias

Apart from the problems related to selecting the person to copy, another important question is what exactly should be copied. The problem is that deciding which of the properties of the model are responsible for her success in a domain is most often a very complicated issue.

In our model, when an agent encounters another agent with high levels of energy or many offspring, it assumes that it holds correct beliefs, on

average, but it possesses no means to decide what to copy, what exactly are the beliefs that lead to the observed success. In a foraging society, the hunter returning with the most numerous prey is usually the most prestigious but, similarly, it is very difficult to know which exactly are the skills required for successful hunting. Good eyesight, speed, aiming but also tracing and knowledge of animal habits are all candidates. It gets even more complex, when considering what is required to acquire these and other, crucial for hunting, skills, such as eating carrots, for example, to improve our eyesight.

Therefore, usually the best solution the copier has to the above problem is to try and copy every behaviour and knowledge of his model, even if not obviously connected to the latter's domain of expertise. "Because figuring out which combinations of ideas, beliefs and behaviors make someone successful is costly and difficult, selection favored a general copying bias" (Henrich and Gil-White, 2001). For example, people may copy the hair style, the outfit etc. of famous singers or sports stars despite the fact that these external characteristics are not related to their ability to sing or compete in a sports court. This fact is recognised by advertisers who contract popular models to advertise products often irrelevant to their activities. The soccer player Ronaldinho is shown, in a TV commercial, to be enjoying a specific potato chips brand. Given that the relationship between his scoring abilities and the snacks he prefers is anything but clear, the advertisers are, in this case, based on general prestige bias: In the minds of Ronaldinho's fans, the snack is related to their model, and copying his behaviour, may in this case, include eating this type of chips.

A very interesting experiment illustrating generality bias was conducted by Ryckman et al. in 1972 and mentioned by Henrich and Gill-White (2001). The researchers asked the opinion of students about student activism. Before expressing their opinion, the students listened to experts both on relevant and on totally irrelevant subjects. The results were impressing: 80% of the students shifted their opinion towards to that of prestigious experts, even in the cases where their area of expertise was as irrelevant as the subject of Ming Dynasty.

4.8.3 Prestige bias allows maladaptive information to propagate

Generality bias has two important side-effects. First of all, it further increases the speed with which information is propagated. As mentioned in section 4.3, infocopying, in general, is the vehicle with which memes are travelling through people's minds. Given that copiers do not just copy the information relevant to the model's expertise, a very large amount of the knowledge, the beliefs and the behaviours of the model will be transferred to all his clients. Furthermore, the existence of prestige hierarchies means that this process is recursive: models are copied by their clients who them-

selves function as models to other individuals and so on. The result is that the cultural variants possessed by the most prestigious will rapidly spread throughout the social group at increasing speeds.

The second important implication of the general prestige bias, is that it is possible for less adaptive or even maladaptive cultural variants to spread as well. Although the copier may possess the capacity of assessing the information to be copied, the costs and the complexity related with this process suggest that evaluating information via individual trial and error will not always be performed. As a result it is highly probable that maladaptive traits will be copied as well, and by the process described above, spread quickly.

We have conducted a number of experiments to evaluate the two side-effects of general prestige bias. In this set of experiments a new factor is added: a very small number of super-agents (about 1% of the total population) in injected in the simulation at a point when equilibrium has been reached, at step 4000 (agents who copy prestigious have totally prevailed and the average number of "correct" beliefs is at maximum possible - usually 2/3 to 3/4 of the number of food types, depending on the constants used to attribute prestige). The super-agents live twice as much as the normal agents (to render their effect more obvious) and are injected with a very high level of energy (250). Most interestingly, super-agents are correct about what to eat about all the food resources but a single one. In the graph below, we can see what happens to the average ideology of all the agents as well as to their beliefs about the food type about which the super-agents are wrong.

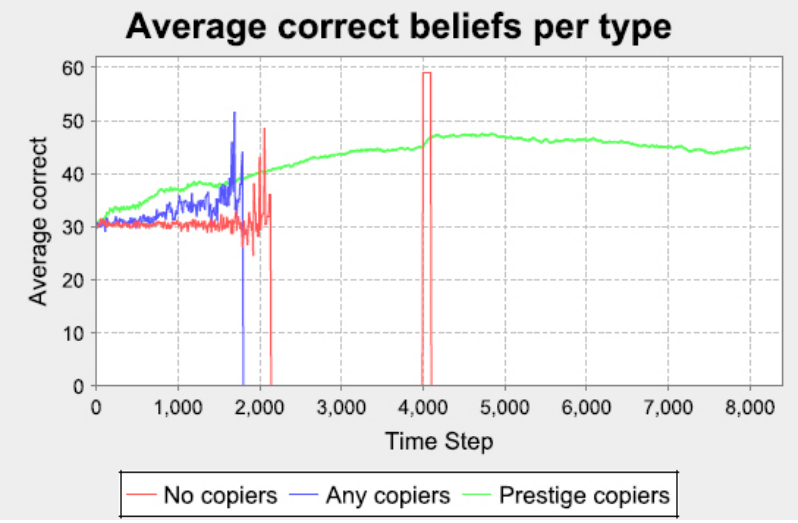


Figure 4.16: Effect of the injection of agents with better than average knowledge in the population. At step 4000, a small number of "super-agents" are injected in the simulation.

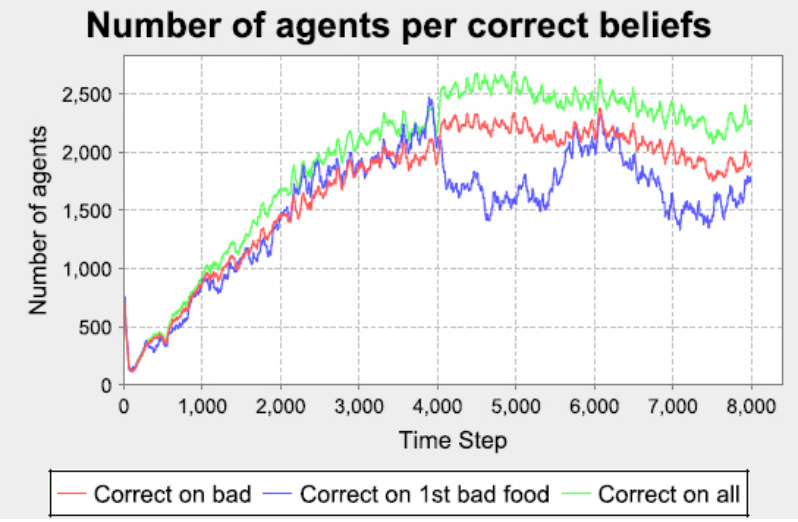


Figure 4.17: Number of agents who are correct about all food types (green curve), unhealthy food types (red curve) and the food type about which super-agents are wrong (blue curve)

Graphs 4.16 and 4.17[2] confirm our assumptions:

- Prestige bias speeds up propagation of knowledge. At the point when super-agents appear, the average number of items on which agents are correct is approximately 45. The propagated correct beliefs of the super-agents very quickly increase this number up to 48 (out of 60).
- Wrong beliefs are propagated as well: As we can see in graph 4.17 the number of agents who are correct about the food item about which super-agent are wrong (blue curve) is significantly decreased, when super-agents are injected (step 4000). Even more illustrating is the distance in this graph between the number of agents who are correct on all (green curve) or harmful (red curve) food types and the food type about which super-agents are wrong. Although these three curves are usually close to each other, when super-agents are injected the distance becomes more than obvious. This means that prestige bias (without any individual testing) leads the agents into acquiring maladaptive attitudes together with adaptive ones, provided that the source of the maladaptive information has on average knowledge of superior quality.

[2]We may observe that, on average, less agents are correct about the harmful food types than about all the food types, which means that agents are usually more correct about the nutritious food types (not shown here for clarity) than harmful ones. This is not a significant result and happens only because nutritious food types are more and add more energy. Therefore, in this specific model, it is more crucial for survival to know what to eat rather than what to avoid.

- Wrong information is sustained in the environment for a very long period of time: Again in graph 4.17 we can see that the curves return to the relationship they had before super-agents were injected in no less than approximately 1000-1500 steps, which corresponds to about 20-30 generations of agents. This means that prestige-biased infocopying allows for both adaptive and maladaptive information to remain in the environment for extended periods of time.

This last set of results about the propagation of maladaptive information is very interesting. It means that, under certain circumstances, it is possible for agents who are considered prestigious to have a beneficial effect to their social groups but at the same time to allow for dangerous beliefs to spread and be sustained. Of course, in order for the latter to occur, the source of maladaptive information must, otherwise, hold adaptive beliefs, useful for those who will copy her.

This side-effect of general prestige-biased copying can be greatly increased by something not illustrated in our model, the capability of certain agents to address to and thus be copied by larger number of clients.

We can think of a number of interesting cases in which the above result may apply. For example, this partially explains the capability of the mass media, especially those that are considered the most prestigious, to increase the average quality of knowledge of their clients, on the one hand, and at the same time to deliberately or not, allow for false or maladaptive information to spread.

Chapter 5

Conclusions and further work

In overall, we believe that this project has been a success. The model we have constructed served very well in providing evidence for the main hypotheses, the adaptiveness of prestige biased infocopying and the fact that it allows maladaptive information to spread, as well. Moreover, given that the main purpose of this project was an *investigation* of the mechanisms that govern prestige bias, the fact that some results of the experiments were unforeseen, such as the effectiveness of age as a success indicator, was, nevertheless, welcomed. These unexpected results, together with the adequate explanations, we believe we have provided, contributed equally to a better insight of the processes studied.

Summarising, we believe that the most important results produced by our model are the following:

- Prestige bias infocopying is adaptive, even in complicated or variable environments, and even if the function used to determine the successful suffers from realistic levels of noise.
- Sharing information, in general, does not require deference to become adaptive.
- Determining the prestigious is a very complicated issue. Age is a useful indicator only when combined with other evidence.
- Conformity bias in model selection can be helpful only when no other reliable information about the model is available.
- Random copying of cultural items is adaptive only in the usually unrealistic condition of environments where better than average quality of information is available.
- General prestige bias speeds up the propagation of information

- General prestige bias allows maladaptive information to spread and persist for a relatively long period of time, provided that the majority of the beliefs of prestigious models are of better than average quality.

Apart from the above, and although, the results produced by any mathematical, software-based or other model should always be under the most careful scrutiny before adopted, we believe that our model manages to adequately simulate certain aspects of reality. We feel that our model is successful because, given the perceived realism of all the results it produced, it can be reused, with or without extensions and modifications, in order to study other issues related to cultural evolution.

The only major problem was the lack of time and expertise required to produce graphs with mean values and variations, instead of simply representative plots. However, all the results that are presented in the text have been adequately tested, with more than ten experiments for each set of conditions.

In what follows, we discuss some ideas about future work, inspired by this project.

5.1 Future work

5.1.1 Conformity bias in cultural evolution

As seen in section 4.2.1 our model illustrates that conformity bias in model selection is adaptive only when no better information is available. However, copying the majority is considered by many as one of the most important heuristics used in infocopying (Henrich and McElreath, 2003; Boyd and Richerson, 2005b).

It would be interesting, therefore, either to extend the existing model or to build a similar one to investigate under exactly which circumstances is conformity biased infocopying adaptive. More interestingly, we predict that conformity bias will also allow maladaptive behaviours to spread and persist, even to a greater extent than prestige bias, given that the averaged beliefs of a population constitute a single source of information.

5.1.2 Dominance and prestige

Henrich and Gill-White (2001) provide an extensive and very interesting comparison between prestige and dominance. While both constitute the basis for the construction of social hierarchies, the authors make some critical distinctions, most important of which is the fact that dominance is based on force or force threat. Dominance is originally based on physical strength while prestige on the possession of more successful than average beliefs and behaviours.

As cultural evolution renders cultural items such as knowledge more vital than genetic traits such as physical strength, it may be that dominance is becoming less adaptive than prestige. At the same time, it is possible that the prevalence of more democratic social structures makes the retainment of power based on force more costly than the power based on prestige. We propose the construction of an agent based model to evaluate this prediction and also to further investigate the adaptive differences between dominance and prestige.

Bibliography

Bentley, R. A., Lipo, P. C., Herzog, A. H., and Hahn, W. M.: 2005, *Constant fashion change under random cultural drift*, Submitted to Behavioral and Brain Sciences

Berger, J., Rosenholtz, S. J., and Zelditch, M. J.: 1980, Status organizing processes, *Annual Review of Sociology* **6**, 479–508

Blackmore, S.: 1999, *The Meme Machine*, Oxford University Press, Oxford

Blackmore, S.: 2000, in R. Augner (ed.), *Darwinizing Culture: The Status of Memetics as a Science*, pp 25–42, Oxford University Press, Oxford

Boesch, C. and Tomasello, M.: 1998, Chimpanzee and human cultures, *Current Anthropology* 39

Bowles, S. and Gintis, H.: 2004, The evolution of strong reciprocity:cooperation in heterogeneous populations, *Theoretical Population Biology* **65**, 17–28

Boyd, R. and Richerson, P. J.: 1985, *Culture and the evolutionary process*, University of Chicago Press, Chicago

Boyd, R. and Richerson, P. J.: 1995, Why does culture increase human adaptability?, *Ethology and Sociobiology* **16**, 125–143

Boyd, R. and Richerson, P. J.: 1996, in *Why culture is common, but cultural evolution is rare*, pp 77–93, Proceedings of the British Academy, Hatfield UK

Boyd, R. and Richerson, P. J.: 2000, in R. Augner (ed.), *Darwinizing Culture: The Status of Memetics as a Science*, pp 143–162, Oxford University Press, Oxford

Boyd, R. and Richerson, P. J.: 2005a, in P. Carruthers, S. Stich, and S. Laurence (eds.), *The Innate Mind: Culture and Cognition*, Oxford University Press, Oxford

Boyd, R. and Richerson, P. J.: 2005b, *The Origin and Evolution of Cultures*, Oxford University Press, Oxford

Brodie, R.: 1996, *Virus of the Mind: The new science of the meme*, WA, Integral press, Seattle

Bryson, J. B. and Wood, M.: 2005, in *Learning Discretely: Behaviour and Organisation in Social Learning*, Proceedings of the Third International Symposium on Imitation in Animals and Arifacts

Carley, K. M.: 2000, in *Computational social science: Agents, interaction*

and dynamics, pp 3–14, Proceedings of the workshop on Simulation of Social Agents: Architecures and Institutions, Chicago USA

Chomsky, N.: 1988, *Language and Problems of Knowledge: the Managua Lectures*, MIT Press, Cambridge

Conte, R.: 2000, in R. Augner (ed.), *Darwinizing Culture: The Status of Memetics as a Science*, pp 83–120, Oxford University Press, Oxford

Darwin, C.: 1859, *The origin of species: Or the Preservation of Favoured Races in the Struggle for Life*, John Murray, London, reprinted in 1981 by Penguin Books

Dawkins, R.: 1976, *The selfish gene*, Oxford University Press, Oxford

Dennett, D. C.: 1995, *Darwin's dangerous idea: Evolution and the meanings of life*, Simon and Schuster, New York

Dennett, D. C.: 2002, in M. Pagel (ed.), *The Encyclopedia of Evolution*, pp E83–E92, Oxford University Press, Oxford

Dessalles, J.-l.: 2000, in C. Knight, J. R. Hurford, and M. Studdert-Kennedy (eds.), *The evolutionary emergence of language: social function and the origin of linguistic form*, pp 62–79, Cambridge University Press, Cambridge

Eiseley, L.: 1958, *Darwin's Century: Evolution and the Men who Discovered It*, Doubleday, New York, cited by (Wagoner, 2004)

Fowler, H., Fowler, F., and Pearsall, J. (eds.): 2004, *The Concise Oxford English Dictionary*, Oxford University Press, Oxford, 11 edition

Gilbert, N.: 1994, Computer simulation of social processes, *Social Research Update* 6, Available from http://www.soc.surrey.ac.uk/sru/SRU6.html, Last accessed 29/04/06

Gintis, H., Bowles, S., Boyd, R., and Fehr, E.: 2003, Explaining altruistic behavior in humans, *Evolution and Human Behavior* **24**, 153–172

Hahn, W. M. and Bentley, R.: 2003, in *Drift as a mechanism for cultural change: an example from baby names*, pp 120–123, Proceedings of the Royal Society B: Biological Sciences, London

Hemelrijk, K. C.: 2000, Towards the integration of social dominance and spatial structure, *Animal Behaviour* **59**, 1035–1048

Henrich, J. and Gil-White, F. J.: 2001, The evolution of prestige. freely conferred deference as a mechanism for enhancing the benefits of cultural transmission, *Evolution and Human Behavior* **22**, 165–196

Henrich, J. and McElreath, R.: 2003, The evolution of cultural evolution, *Evolutionary Anthropology* **12**, 123–135

Heylighen, F.: 1997, Objective, subjective and intersubjective selectors of knowledge, *Evolution and Cognition* **3**, 63–67

Heylighen, F.: 1998, in *What makes a meme succesful? Selection criteria for cultural evolution*, Proceedings of the 15th International Congress on Cybernetics, Namur, Available from http://cogprints.org/1132/00/MemeticsNamur.html, Last accessed 25/03/06

Luke, S., Balan, G., Panait, L., Cioffi-Revilla, C., and Paus, S.: 2003, in *Mason: a jave multi-agent simulation library*, pp 49–64, Proceedings of the conference on challenges in social simulation, Chicago USA

Luke, S., Cioffi-Revilla, C., Panait, L., and Sullivan, K.: 2004, in *MASON: A New Multi-Agent Simulation Toolkit*, Proceedings of the 2004 SwarmFest Workshop, Michigan USA

Noble, J. and Todd, P.: 2006, in K. Dautenhahn and C. Nehaniv (eds.), *Imitation in animals and artifacts*, Cambridge, MA: MIT Press, Cambridge

Packard, A.: 1901, *Lamarck, the Founder of Evolution: His Life and Work*, Longmans, Green, and Co., New York, cited by (Wagoner, 2004)

Railsback, F. S., Lytinen, L. S., and Jackson, K. S.: 2005, *Agent-based simulation platforms: Review and development recommendations*, Submitted to Simulation, Available from http://www.humboldt.edu/ ecomodel/documents/ABMPlatformReview.pdf, Last accessed 29/04/06

Richerson, P. J. and Boyd, R.: 2005, *Not By Genes Alone: How Culture Transformed Human Evolution*, The University of Chicago Press, Chicago

Rogers, R. A.: 1988, Does biology constrain culture?, *American Anthropologist, New Series* **90**, 819–831

Sapienza, M.: 2003, Do real options perform better than net present value? testing in an artificial financial market, *Journal of Artificial Societies and Social Simulation* 6

Schelhorn, T., O'Sullivan, D., Hacklay, M., and Thurstain-Goodwin, M.: 1999, Streets: An agent-based pedestrian model, *CASA Center for Advanced Spatial Analysis Working Papers Series* 9, Available from http://www.casa.ucl.ac.uk/streets.pdf, Last accessed 29/04/06

Sperber, D.: 2000, in R. Augner (ed.), *Darwinizing Culture: The Status of Memetics as a Science*, pp 163–174, Oxford University Press, Oxford

Tobias, R. and Hofmann, C.: 2004, Evaluation of free java-libraries for social-scientific agent based simulation, *Journal of Artificial Societies and Social Simulation* 7

Tooby, J. and Cosmides, L.: 1989, Evolutionary psychology and the generation of culture: Part i theoretical considerations, *Ethology and Sociobiology* **10**, 29–49

Čače, I. and Bryson, J. B.: 2005, in *Why Information can be free*, pp 12–15, Proceedings of the Second International Symposium on the Emergence and Evolution of Linguistic Communication (EELC'05), Hatfield UK

Waggoner, B.: 2004, *Jean-Baptiste Lamarck (1744–1829)*, webpage, available at http://www.ucmp.berkeley.edu/history/lamarck.html

Wooldridge, M.: 2002, *An introduction to multiagent systems*, John Wiley & Sons, Chichester, England

Appendix A

An overview of Agent Based Modelling

Agent based modelling (ABM) is a research and experimental tool that is becoming increasingly popular, in various domains. It is used to model interactions between individuals (agents) who are placed in a environment and interact with each other, according to certain rules. As the social psychologist Rosaria Conte puts it "the simulation based study of social phenomena has already proved fruitful in promoting both the methodological development of social sciences and a cross-fertilization between agent theory and social theory" (Conte, 2000, p.87)

ABM is another abstraction: we build a model of the target that we want to study, that is effectively much smaller and more accessible than the target (Gilbert, 1994) itself. We aim, however, to retain the similarity between the model and the target in a degree sufficient to make the conclusions drawn from the model, applicable to the target.

Agent based modelling has been applied in various domains such as animal behaviour (Hemelrijk, 2000), economics (Sapienza, 2003) , sociology (Gintis et al., 2003), traffic and vehicle simulations (Schelhorn et al., 1999) etc.

Advantages

Agent based modelling "...enable the researcher to examine the relations among groups and individual agents and to witness the way in which these relations enable, constrain, and affect agent and group behavior, as well as emergent phenomena" (Carley, 2000). It allows the study of complicated systems, by defining a relatively small set of properties of the environment and of the agents that "inhabit" this environment.

One of the most important characteristics of ABM is that it allows "artificial" experimentation when actual experiments cannot take place due to physical or other constraints (e.g. examining very complicated phenomena

or the very long term effects of a phenomenon, or examining phenomena of the distant past etc). For example, the EOS project, undertaken in the University of Essex (Wooldridge, 2002), investigated the sudden rise in the complexity of societies in Upper Palaeolithic France. .

As Nigel Gilbert argues (1994), one of the main advantages of a computer simulation is that all parameters, attributes and relationships must be clearly specified, meaning that they must be thought through very carefully. The nature of ABM, thus, prohibits a vague specification and that way many mistakes can be avoided. This also means, on the other hand, that complex systems are more difficult to be implemented.

Problems

Perhaps the most important problem with agent based modelling, is the difficulty to validate the models developed. According to Gilbert "Ideally, a simulation should produce outputs which match those of the target for all possible inputs which can be envisaged to occur in reality, and should fail to produce output in all other circumstances" (Gilbert, 1994). However, it is not possible to test all possible inputs, in order to verify the above requirement. Therefore, in most cases an approximation is used. The goal is to design and implement a model that produces results as close to reality as possible.

Apart from the verification problem, the fact that, in ABM, very small amounts of code are responsible for large parts of functionality, hides the danger of tiny mistakes leading to serious output errors. Therefore, very careful programming and reflection on the output is required.

Available platforms

A number of software tools for agent based modelling have been developed over the past few years. The main purpose of these platforms is to relieve the researcher from the need to program the model from scratch. They provide higher level tools that can be used with minimum knowledge of programming principles. Below we briefly review some of the most popular such platforms.

- NetLogo: Perhaps the most widely used platform that owes its popularity to its especially easy to understand and use environment. It provides a graphical user interface, and a high level programming language (a variant of Logo) based on simple and comprehensive concepts. It also provides one of the most complete and comprehensive documentation.

- Swarm: Developed in Objective-C (although a Java variant exists), Swarm is a general-purpose toolbox for agent based modelling. A

model in Swarm comprises of sets of "swarms", i.e. groups of objects with associated functions to be executed.

- RePast: Initially started as a Java version of Swarm but it gradually became an independent platform. It was designed for use mainly in social sciences, so it include special tools for the domain.

Mason Multiagent Simulation Toolkit

This project is implemented using MASON, one of the most recent ABM platforms. MASON is a relatively smaller platform, compared to those reviewed above. It is simply a library of Java classes that provide all the required functionality to easily build an agent based model. MASON is especially powerful and easy to use, provided that the researcher has some familiarity and experience with the Java programing language. It is also one of the faster platforms according to comparison tests (Railsback et al., 2005). Other useful features of MASON include the capability to support a very large number of agents simultaneously (a feature especially important in social studies) and also to guarantee to production of identical results when executed with identical parameters, even when running on different operating systems or different hardware (Luke et al., 2003; Luke et al., 2004).

Perhaps the most serious drawback of MASON is the minimal documentation currently available. We hope that, with time, adequate documentation will be provided.

Appendix B

How to use the model

In the CD-ROM that accompanies this project, in the folder named "Executable", there are the files that can be used to run simulations of the model we have implemented.

By double–clicking the file called "prestige.jar" or by typing in a terminal

```
>java -jar prestige.jar
```

the main control panel of the simulation appears (see B.1).

In the tab called "Model", we can set all the values for the variables that determine the conditions of the simulation, such as the number of different food types, the initial number of agents, the constants that determine the prestige assessment function, the number of super-agents etc. In the tab called "Console" various other parameters affecting the simulation can be set, such as the step at which the simulation should automatically halt etc.

The simulation starts when we press the play button at the bottom left corner of the window. There are also control buttons that pause or stop the simulation.

Once the simulation is started, from the tab "Displays" we can show/hide various displays of the simulation. The first object, "The role of prestige in knowledge spread Display", is an illustration of the model, similar to the one shown in figure 4.1. The rest of the objects are bar charts and plots that we will briefly describe, starting from the most useful:

Average number of correct beliefs per agent type: A plot with three curves, one for each genotype, showing the mean number of beliefs that are correct for all agents of each genotype.

Number of agents per time step: A plot with three curves, one for each genotype, showing the number of agents per time step.

Number of agents, correct in bad/1st bad: A plot with the number of agents who are correct about the first unhealthy food type (the

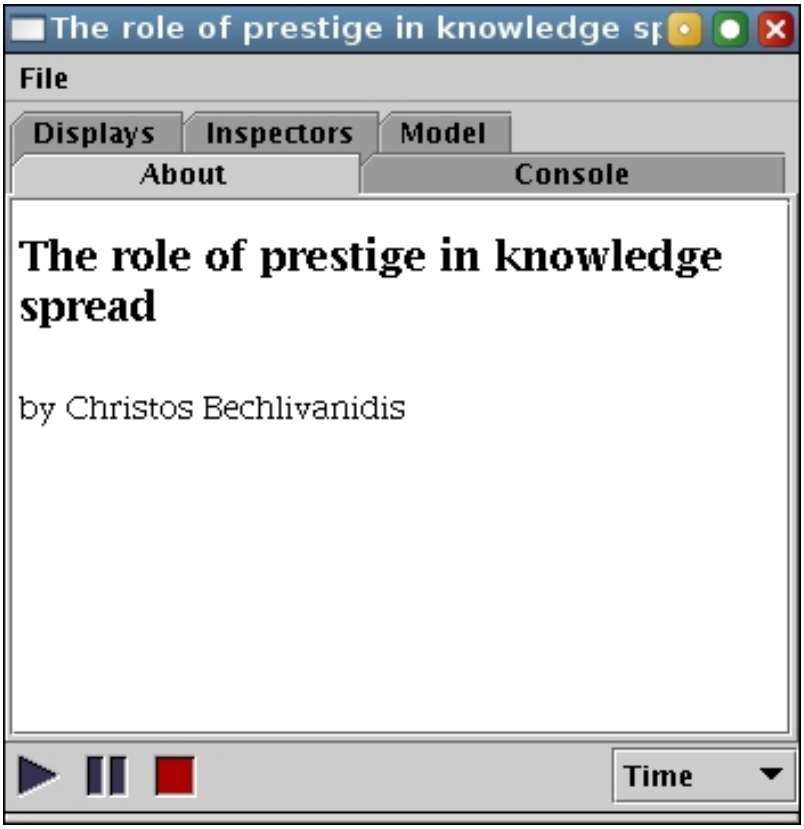


Figure B.1: The main control panel of the implemented software based on MASON toolkit

one about which super-agents are wrong), those that are correct on all unhealthy foods on average and those that are correct on all food types.

Number of agents per belief: A bar chart showing each food item and the number of agents believe it should be consumed.

Average energy per number of correct beliefs: A bar chart showing the average energy of the agents with 1,2,etc. correct beliefs.

Average energy per agent type: A bar chart displaying the average energy of each agent type.

Number of agents per correct beliefs A bar chart showing the number of agents who are correct about 1,2,etc. food items.

B.1 System Requirements

The only requirement in order to use our model is a system with Java Runtime Environment installed (available at http://java.sun.com/j2se/1.4.2/download.html). The requirements for J2SE can be found at http://java.sun.com/j2se/1.4.2/install-windows.html (MS Windows) and http://java.sun.com/j2se/1.4.2/install-linux.html (Linux).

Appendix C

Experiment conditions

What follows is a table with the values of all the variables used in experiments presented throughout this text, together with the graphs and the sections where the results of those experiments were presented. The values that appear in bold face illustrate the variables that constituted the major point of interest in each experiment. Wherever multiple values or a range of values are present, more information can be found in the corresponding graph. For more information about the way variables are used please refer to each corresponding section as well as the code and the inline comments presented in appendix D.

	1	2	3	4	5	6	7	8
Section	4.2	4.2	4.2	4.2	4.2	4.2	4.2.1	4.2.1
Experiment name	EnergyOnly	OffsprOnly	EnergyOffspr	AgeOnly	EnergyAge	OffsprAge	Conf(ormity)	AgeConf
Figure	4.2	4.4	4.5	4.6	4.7(a)	4.7(b)	4.8	4.9
Steps	8000	8000	8000	8000	8000	8000	8000	8000
Agents #	3000	3000	3000	3000	3000	3000	3000	3000
Super-agents #	0	0	0	0	0	0	0	0
Lifespan	50	50	50	50	50	50	50	50
Foods Types #	60	60	60	60	60	60	60	60
Switch world every	0	0	0	0	0	0	0	0
Show Deference	yes	yes	yes	yes	yes	yes	yes	yes
Constants for determining prestige:								
Energy (E)	**0.7**	0	**0.7**	0	**0.7**	0	0.7	0
Offspring # (O)	0	**0.7**	**0.7**	0	0	**0.7**	0	0
Age (A)	0	0	0	**0.7**	**0.7**	**0.7**	0.5	**0.7**
Times Copied (C)	0	0	0	0	0	0	**0-0.2**	**0.1**
Noise	0	0	0	0	0	0	0	0

	9	10	11	12	13	14	15	16	17
Section	4.4	4.4	4.5	4.5	4.5	4.6	4.7	4.8.1	4.8.3
Experiment name	allDef	allnoDef	noPrestigeDef	noPrestigeNoDef	10Foods	$_{F}oods$	Switch	Noise	Super
Figure	4.10(a)	4.10(b)	4.11(a)	4.11(b)	4.12	4.13	4.14	4.15	4.16 & 4.17
Steps	8000	8000	8000	8000	8000	8000	8000	8000	8000
Agents #	3000	3000	3000	3000	3000	3000	3000	3000	3000
Super-agents #	0	0	0	0	0	0	0	0	**30**
Lifespan	50	50	50	50	50	50	50	50	50
Foods Types #	60	60	60	60	**10**	**40-160**	60	60	60
Switch world every	0	0	0	0	0	0	**100-1500**	0	0
Show Deference	yes	**no**	yes	**no**	yes	yes	yes	yes	yes
Constants for determining prestige:									
Energy (E)	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.7
Offspring # (O)	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2
Age (A)	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
Times Copied (C)	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01
Noise	0.1	0.1	0.1	0.1	0	0	0	**0.1-1**	0.1

Appendix D

Code listings

The implementation of the model consists of the following four classes:

Model: The heart of the model. Sets up the "world", populates it with foods and agents and controls that various actions at each step of the simulation.

Agent: Represents an agent. There are three types of agents that move, copy each other, depending on their type and eat resources according to their "ideology". (see 4.1.2).

Viewer: Sets up the control panel, controls the simulation and creates the various frames to hold the charts.

Experiments: "Silently" runs a number of simulations with different values for some of the variables. At the end of each simulation all the charts are exported to a set of .pdf files.

DataWriter: Helper class to write data from the experiments in text files in a format suitable for inclusion in latex tables.

D.1 Model

```
package prestige;

//for the simulation
import sim.engine.*;
import sim.field.grid.*;
import sim.util.*;
import ec.util.*;
//for the charts
import org.jfree.data.xy.XYSeries;
import org.jfree.data.xy.XYSeriesCollection;
import org.jfree.chart.*;
import org.jfree.chart.plot.*;
import org.jfree.data.category.*;
//for the pdf files
import com.lowagie.text.*;
import com.lowagie.text.pdf.*;
import java.io.*;
import java.awt.*;
import java.awt.geom.*;

/**
 *
 * <p>Title: Model</p>
 *
 * <p>Description: Sets up the world, populates it with agents,
 * controls food growth and agent actions
 * </p>
 *
 *
 * @author Christos Bechlivanidis
 */

public class Model extends SimState {

  /*******************Information about the world******************/

//the grid where the agents will interact
public SparseGrid2D grid;
//the grid holding the patches of food (0 is empty)
public IntGrid2D patches;
//the size of the world
public final int gridWidth = 200;
public final int gridHeight = 200;
//How often to change the world (change the type of 1/3 of the foods)
//(use a very large constant, to have a static world
public int changeWorldEvery = 100000;

/*****************************************************************/

/*******************Information about the food*******************/

//The total number of food types
public static int noThings = 60;
//the chances an empty patch has to be filled with some type of food
public final int replaceRate = 30;
//the number of food types that decrease energy when eaten
public static int badThings;
//the number of food types that increase energy when eaten
public static int goodThings;
//the ratio between nutritious and unhealthy food types
public static float badGoodRatio = 0.3f;
//nutritious value of each type of food source
public static int badValue = -6;
public static int goodValue = 2 * ( -badValue);
//whether the first n food types are unhealthy foods or not
//(used to determine whether a food type is good or bad)
//(it is usually true, unless the world switches state
public boolean badAreFirst = true;
/*****************************************************************/

/***********************Information about the agents*******************/
```

```
//number of agents that will initially populate the world
public int noAgents = 3000;
//Holds the total number of agents for each type
//(no copy, copy anyone, copy prestigious)
public int agentTypes[] = new int[3];
//the number of times the agents communicate per step
public static int talkTime;
//the region within which agents communicate
public static int comRegion = 3;
//The length of an agent's step
public static int stepLength = 1;
//number of agents born
public int born;
//number of agents died
public int died;
//total agents at one point in the program
public int totalagents;
//number of agents that died for exceeding age limit
public static int naturalDied = 0;
//number of agents that died due to starvation
public static int starvedDied = 0;

//number of super agents to inject
public int superAgents = 30;
//when to inject super agents
//(use negative value for no injection)
public int injectSuperAt = -1;

/*********************Constants used when assessing prestige********/
public double ageImportance = 0.5;
public double energyImportance = 0.7;
public double offspringImportance = 0.2;
public double conformImportance = 0.01;
//error in assessing prestige
public double error = 0.1;
//Whether the agent allows the model to eat first
public boolean deference = true;
```

```
/*****************************************************************/


/*************************Information about experiments***************/
//experiment number (used to name pdf files produced)
public int expNo = 1;
//Experiments object (used to notify it when a simulation
// has finished)
public Experiments exp;
//The name of the experiment (used by Experiments)
public String expName;
//a flag used to know if the user wishes the results (graphs)
//to be written to a set of PDF files
public boolean writePdf = true;
//how often should data be written to files
public static int writeDataEvery = 300;
//The writers used to write data to files (if started from Experiments)
public DataWriter beliefsWriter;
public DataWriter numbersWriter;
/*****************************************************************/


/*************************Charts*************************************/
//Average correct beliefs
public XYSeries avgCorrectNoCopy = new XYSeries(
    "No copiers");
public XYSeries avgCorrectCopyAny = new XYSeries(
    "Any copiers");
public XYSeries avgCorrectCopyPrestige = new XYSeries(
    "Prestige copiers");

//Number of agents, per type
public XYSeries noCopyNum = new XYSeries("No copiers");
public XYSeries copyAnyNum = new XYSeries("Any copiers");
public XYSeries copyPrestigeNum = new XYSeries(
    "Prestige copiers");
```

```
//Number of agents, who are correct about all food types
public XYSeries avgCorrect = new XYSeries(
    "Correct on all");
//Number of agents, who are correct about unhealthy food types
public XYSeries avgCorrectOnBad = new XYSeries(
    "Correct on bad");
//number of agents who are correct about 1st bad food
//(the one about which super-agents are wrong
public XYSeries avgCorrect1Bad = new XYSeries(
    "Correct on 1st bad food");

//Beliefs per food type
public DefaultCategoryDataset beliefStats = new
                                        DefaultCategoryDataset();
//Energy per agent type
public DefaultCategoryDataset energyPerTypeStats = new
    DefaultCategoryDataset();
//Average energy
public DefaultCategoryDataset avgEnergyStats = new
                                       DefaultCategoryDataset();
//Correct beliefs per agent
public DefaultCategoryDataset agentsPerCorrectStats = new
    DefaultCategoryDataset();

//The charts that will be displayed
public static JFreeChart beliefBarChart;
public static JFreeChart energyPerTypeBarChart;
public static JFreeChart agentsPerCorrectBarChart;
public static JFreeChart avgCorrectChart;
public static JFreeChart numberPerTimeChart;
public static JFreeChart avgEnBarChart;
public static JFreeChart avgCorrectGoodBadChart;

/************************************************************/

/**
 * constructor
 */
public Model(long seed) {
  super(new MersenneTwisterFast(seed), new Schedule(3));
}

/**
 * Cosntructor to be used when experiments are run
 * (sim starts from an Experiments object)
 * @param exp Experiments The object that starts the sim
 */
public Model(long seed, Experiments exp) {
  super(new MersenneTwisterFast(seed), new Schedule(3));
  beliefsWriter = exp.beliefsWriter;
  numbersWriter = exp.numbersWriter;
  this.exp = exp;
}

/**
 * Sets up the grid/world:
* places the initial agents and the patches,
 * ready for the beginning of the simulation.
 *
 */
public void setGrid() {
  //initialise the grid (for food sources)
  for (int i = 0; i < gridHeight; i++) {
    for (int j = 0; j < gridWidth; j++) {
      patches.field[i][j] = 0;
    }
  }

  //simulate food growth (a number of times so
  //that it seems like there is some growth)
  for (int k = 0; k < 10; k++) {
    growFood();
  }
```

```
  //put all agents on the grid
  placeAgents(noAgents);

}

/**
 * Grow food to patches
 */
private void growFood() {
  for (int i = 0; i < gridHeight; i++) {
    for (int j = 0; j < gridWidth; j++) {
      if (random.nextFloat() * 1000 < replaceRate) {
        //grow food of type [1,noThings] (0 is empty patch)
        patches.field[i][j] = random.nextInt(noThings) +
                              1;
      }
    }
  }
}

/**
 * A helping method that creates the agents
 * (to be used only at the beginning of
 * the simulation) and places them on the grid.
 * (allows multiple agents to be at the same location)
 * @param int num specifying how many agents to place
 */
private void placeAgents(int num) {
  Agent temp;
  for (int i = 0; i < num; i++) {
    //Create a new agent with random age and energy
    temp = new Agent(random.nextInt(Agent.lifespan),
                     random.nextInt(Agent.maxEnergy - 1) +
                     1);
    //populate agent's beliefs randomly
    for (int j = 0; j < noThings; j++) {
      temp.ideology[j] = random.nextBoolean();
    }
    //Randomly define the type of the agent
    temp.type = random.nextInt(agentTypes.length);
    //Update the number of agents of this type in the world
    agentTypes[temp.type]++;

    //finds a random location on the grid and puts the agent there
    grid.setObjectLocation(temp, random.nextInt(gridWidth),
                           random.nextInt(gridHeight));
    temp.location = grid.getObjectLocation(temp);
  }
}

/**
 * Inject super-agents
 * @param number int When should super-agents be injected
 */
private void injectSuperAgents(int number) {
  for (int n = 0; n < number; n++) {
    //Create a super-agent with age 0 and energy 250
    Agent superAgent = new Agent(0, 250);
    //super-agents are wrong on first harmful food item
    //and correct on everything else
    for (int i = 0; i < noThings; i++) {
      //the belief on first harfmul is wrong
      if (badAreFirst && i == 0) {
        superAgent.ideology[i] = true;
      } else if (!badAreFirst && i == goodThings) {
        superAgent.ideology[i] = true;
      }
      //everything else is correct
      else {
        superAgent.ideology[i] = isGood(i) ? true : false;
      }
    }
    //Super-agents don't copy other agents
    superAgent.type = Agent.NOCOPY;
```

```
    //So that the agent performs a super-step
    superAgent.isSuper = true;
    //place super agent randomly somewhere
    grid.setObjectLocation(superAgent,
                           random.nextInt(gridWidth),
                           random.nextInt(gridHeight));
    superAgent.location = grid.getObjectLocation(
        superAgent);
  }
  agentTypes[Agent.NOCOPY] += number;
}

/**
 * Starts the simulation. Updates statistics and includes
 * to steppable inner classes to control simulation
 */
public void start() {
  super.start();
  //*************reset the statistical information
   born = 0;
  died = 0;
  totalagents = noAgents;
  for (int i = 0; i < agentTypes.length; i++) {
    agentTypes[i] = 0;
  }
  //*******************************************

   //****Set bad-good things and communication time
    badThings = (int) (noThings * badGoodRatio);
  goodThings = noThings - badThings;
  talkTime = noThings / 3;
  //*******************************************

   //*****************************resets the charts
    beliefStats.clear();
  avgEnergyStats.clear();
  energyPerTypeStats.clear();
  agentsPerCorrectStats.clear();
  avgCorrectNoCopy.clear();
  avgCorrectCopyAny.clear();
  avgCorrectCopyPrestige.clear();
  noCopyNum.clear();
  copyAnyNum.clear();
  copyPrestigeNum.clear();
  avgCorrectOnBad.clear();
  avgCorrect1Bad.clear();
  avgCorrect.clear();
  //*******************************************

   //creates the grid/patches that the agents will
   //populate
   patches = new IntGrid2D(gridWidth, gridHeight);
  grid = new SparseGrid2D(gridWidth, gridHeight);
  setGrid();

  //creates a thread that updates the patches each timestep
  Steppable patchUpdater = new Steppable() {
    public void step(SimState state) {
      growFood();
    }
  };
  //starts the updating thread: start at EPOCH and run once each timestep
  schedule.scheduleRepeating(Schedule.EPOCH, 1,
                             patchUpdater, 1);

  //creates another thread that makes the agents step at each timestep
  //and updates statistics and charts, at the same time
  Steppable agentUpdater = new Steppable() {
    public void step(SimState state) {
      //the current step
      long currentStep = state.schedule.getSteps();
      //the current time
      double currentTime = state.schedule.time();
```

```
//switch world
if ((currentStep + 1) % changeWorldEvery == 0) {
  changeWorld();
}
//inject super-agents
if (currentStep == injectSuperAt) {
  injectSuperAgents(superAgents);
}

//**********************************for charts
 //used to collect agent beliefs per agent type
 int[][] beliefs = new int[agentTypes.length][
                   noThings];
//used to collect agent correct beliefs per agent type
int[][] correctBeliefs =
    new int[agentTypes.length][noThings + 1];
//average energy per number of correct beliefs
int[] avgEn = new int[noThings + 1];
//average energy per agent type
int[] energyType = new int[agentTypes.length];
//correct beliefs per agent type
int[] correctsPerType = new int[agentTypes.length];
//number of correct beliefs about unhealthy foods
int correctsOnBad = 0;
//number of correct beliefs about 1st unhealthy food
//(about which super-agents are wrong
int corrects1Bad = 0;
//**********************************************


 //puts all the agents who are on the grid into a bag
 Bag agents = grid.getAllObjects();
Agent temp;

//Call "step" for every agents and update statistics
//based on every agent's beliefs
for (int i = 0; i < agents.numObjs; i++) {
temp = (Agent) agents.objs[i];
temp.step(state);
//number of food types about which this agent is correct
int corrects = 0;
//the current energy of the agent
energyType[temp.type] += temp.energy;

//cycle through all food types to collect agent's beliefs
//about each
for (int j = 0; j < noThings; j++) {
  //if the agent believes jth food it nutritious
  if (temp.ideology[j]) {
    beliefs[temp.type][j]++;
    //and if the food type is indeed nutritious
    if (isGood(j)) {
      corrects++;
    }
  }
  //if jth food is unhealthy and agent believes it's unhealthy
  else if (!isGood(j)) {
    if (temp.type == Agent.COPYPRESTIGIOUS) {
      correctsOnBad++;
      //if it is the first bad food
      //(about which super-agents are wrong)
      if ((badAreFirst && j == 0) ||
          (!badAreFirst && j == goodThings)) {
        corrects1Bad++;
      }
    }
    corrects++;
  }
}

//Increment the number of agents who have k number of correct beliefs
correctBeliefs[temp.type][corrects]++;
//Add the energy to the energy of those agents with
//given number of correct beliefs
```

```
  avgEn[corrects] += temp.energy;
  //Add this agent's number of correct beliefs
  //to the number of correct beliefs of his genotype
  correctsPerType[temp.type] += corrects;
}
//Output the data to files (if started from Experiments class)
if (exp != null && currentStep != 0 &&
    (currentStep % writeDataEvery == 0)) {
  String beliefsData = "";
  String numbersData = "";

  for (int i = 0; i < agentTypes.length; i++) {
    numbersData += agentTypes[i] + "/";
    String floatFormat = new Float(agentTypes[i] !=
                                   0 ? (float)
                                   correctsPerType[
                                   i] /
                                   agentTypes[i] :
                                   0).toString();
    beliefsData +=
        floatFormat.substring(0,
                              floatFormat.
                              indexOf(".") + 2) +
        "/";

  }
  numbersWriter.writeData(numbersData);
  beliefsWriter.writeData(beliefsData);

}

//Update charts (every ten steps to make sim run faster)
if (currentStep % 10 == 0) {
  //energy per agent type
  energyPerTypeStats.addValue(agentTypes[0] != 0 ?
                              energyType[0] /
                              agentTypes[0] : 0,
                              "Agents",
                              "No copy");
energyPerTypeStats.addValue(agentTypes[1] != 0 ?
                              energyType[1] /
                              agentTypes[1] : 0,
                              "Agents",
                              "Copy anyone");
energyPerTypeStats.addValue(agentTypes[2] != 0 ?
                              energyType[2] /
                              agentTypes[2] : 0,
                              "Agents",
                              "Copy prestige");

//Average correct beliefs per time step
avgCorrectNoCopy.add((double) currentTime,
                     agentTypes[Agent.NOCOPY] > 0 ?
                     (double) correctsPerType[
                     Agent.NOCOPY] /
                     agentTypes[Agent.NOCOPY] : 0);

avgCorrectCopyAny.add((double) currentTime,
                      agentTypes[Agent.COPYANY] !=
                      0 ?
                      (double) correctsPerType[
                      Agent.COPYANY] /
                      agentTypes[Agent.COPYANY] :
                      0);

avgCorrectCopyPrestige.add((double) currentTime,
                           agentTypes[Agent.
                           COPYPRESTIGIOUS] != 0 ?
                           (double)
                           correctsPerType[Agent.
                           COPYPRESTIGIOUS] /
                           agentTypes[Agent.
                           COPYPRESTIGIOUS] : 0);
```

```
//Number of agents per time step
noCopyNum.add((double) currentTime,
              agentTypes[Agent.NOCOPY]);
copyAnyNum.add((double) currentTime,
               agentTypes[Agent.COPYANY]);
copyPrestigeNum.add((double) currentTime,
                    agentTypes[Agent.
                    COPYPRESTIGIOUS]);

//Average Correct on unhealthy foods + first unheathy food
avgCorrectOnBad.add((double) currentTime,
                    (double) correctsOnBad /
                    badThings);
avgCorrect1Bad.add((double) currentTime,
                   (double) corrects1Bad);
avgCorrect.add((double) currentTime,
               (double) correctsPerType[Agent.
               COPYPRESTIGIOUS] / noThings);

for (int i = 0; i < noThings; i++) {
  //number of agents believing the nutritious
  //nature of each food type
  beliefStats.addValue(beliefs[Agent.NOCOPY][i],
                       "No copy", "" + i);
  beliefStats.addValue(beliefs[Agent.COPYANY][i],
                       "Copy any", "" + i);
  beliefStats.addValue(beliefs[Agent.
                       COPYPRESTIGIOUS][
                       i], "Copy prestige",
                       "" + i);

  //average energy per correct beliefs
  avgEnergyStats.addValue(correctBeliefs[0][i] !=
                          0 ?
                          (double) avgEn[i] /
                          correctBeliefs[0][i] :
                          0,
                          "No copy",
                          "" + (i - badThings));
  avgEnergyStats.addValue(correctBeliefs[1][i] !=
                          0 ?
                          (double) avgEn[i] /
                          correctBeliefs[1][i] :
                          0,
                          "Copy any",
                          "" + (i - badThings));
  avgEnergyStats.addValue(correctBeliefs[2][i] !=
                          0 ?
                          (double) avgEn[i] /
                          correctBeliefs[2][i] :
                          0,
                          "Correct Beliefs",
                          "" + (i - badThings));

  //number of agents per correct beliefs
  agentsPerCorrectStats.addValue(correctBeliefs[
                                 Agent.NOCOPY][i],
                                 "No copy",
                                 "" + i);
  agentsPerCorrectStats.addValue(correctBeliefs[
                                 Agent.COPYANY][i],
                                 "Copy any",
                                 "" + i);
  agentsPerCorrectStats.addValue(correctBeliefs[
                                 Agent.
                                 COPYPRESTIGIOUS][
                                 i],
                                 "Copy prestige",
                                 "" + i);
    }
  }
 }
};
```

```
  //starts the agents thread: starts at EPOCH
  // and runs once each timestep (runs before the patch updater)
  schedule.scheduleRepeating(Schedule.EPOCH, 0,
                             agentUpdater, 1);
}

/**
 * Create all the charts
 */
public void createCharts() {
  beliefBarChart = ChartFactory.createBarChart
                  ("Number of Agents per belief", // Title
                   "Different type of foods", // X-Axis label
                   "Number of agents", // Y-Axis label
                   beliefStats, // Dataset
                   PlotOrientation.VERTICAL,
                   true, // Show legend?
                   true, // show tooltips?
                   false); // make URL?

  energyPerTypeBarChart = ChartFactory.createBarChart(
      "Average energy per agent type", // Title
      "Types of agents", // X-Axis label
      "Average energy", // Y-Axis label
      energyPerTypeStats, // Dataset
      PlotOrientation.VERTICAL,
      true, // Show legend?
      true, // show tooltips?
      false); // make URL?

  agentsPerCorrectBarChart = ChartFactory.createBarChart
                             (
                                 "Number of agents per correct beliefs", // Title
                                 "Number of correct beliefs", // X-Axis label
                                 "Number of agents", // Y-Axis label
                                 agentsPerCorrectStats, // Dataset
                                 PlotOrientation.VERTICAL,
                                 true, // Show legend?
                                 true, // show tooltips?
                                 false); // make URL?

  XYSeriesCollection avgCorrectPlot = new
                                      XYSeriesCollection();
  avgCorrectPlot.addSeries(avgCorrectNoCopy);
  avgCorrectPlot.addSeries(avgCorrectCopyAny);
  avgCorrectPlot.addSeries(avgCorrectCopyPrestige);

  avgCorrectChart = ChartFactory.createXYLineChart(
      "Average correct beliefs per type", // the title of the chart
      "Time Step", // the label for the X axis
      "Average correct", // the label for the Y axis
      avgCorrectPlot, // the dataset for the chart
      PlotOrientation.VERTICAL, // the orientation of the chart
      true, // show legend?
      true, // show tooltips?
      false); //make URL?

  XYSeriesCollection numberPlot = new XYSeriesCollection();
  numberPlot.addSeries(noCopyNum);
  numberPlot.addSeries(copyAnyNum);
  numberPlot.addSeries(copyPrestigeNum);

  numberPerTimeChart = ChartFactory.createXYLineChart(
      "Number of agents per type", // the title of the chart
      "Time Step", // the label for the X axis
      "Number of agents", // the label for the Y axis
      numberPlot, // the dataset for the chart
      PlotOrientation.VERTICAL, // the orientation of the chart
      true, // show legend?
      true, // show tooltips?
      false); //make URL?

  XYSeriesCollection avgGoodBad = new XYSeriesCollection();
```

```
  avgGoodBad.addSeries(avgCorrectOnBad);
  avgGoodBad.addSeries(avgCorrect1Bad);
  avgGoodBad.addSeries(avgCorrect);

  avgCorrectGoodBadChart = ChartFactory.createXYLineChart(
      "Number of agents per correct beliefs", // the title of the chart
      "Time Step", // the label for the X axis
      "Number of agents", // the label for the Y axis
      avgGoodBad, // the dataset for the chart
      PlotOrientation.VERTICAL, // the orientation of the chart
      true, // show legend?
      true, // show tooltips?
      false); //make URL?

  avgEnBarChart = ChartFactory.createBarChart(
      "Average Energy per Number of correct beliefs", // Title
      "Number of correct beliefs", // X-Axis label
      "Average Energy", // Y-Axis label
      avgEnergyStats, // Dataset
      PlotOrientation.VERTICAL,
      true, // Show legend?
      true, // show tooltips?
      false); // make URL?

}

/************************Helper methods********************/

/**
 * Whether a food type is nutritious
 * @param foodType int the food type to check
 * @return boolean true if it is nutritious
 */
public boolean isGood(int foodType) {
  if ((badAreFirst && foodType >= badThings) ||
      (!badAreFirst && foodType < goodThings)) {
    return true;
  } else {
    return false;
  }
}

/**
 * The energy value (positive or negative) of a food type
 * @param foodType int The food type to check
 * @return int
 */
public int getFoodValue(int foodType) {
  return isGood(foodType) ? Model.goodValue :
      Model.badValue;
}

/**
 * Switch the world state (unhealthy become healthy
 * and 1/3 of nutritious foods become unhealthy
 */
public void changeWorld() {
  badAreFirst = !badAreFirst;
}

/**
 * Export a chart to a pdf file
 * @param chart JFreeChart The chart to export
 * @param width int The width of the exported image
 * @param height int The height of the exported image
 * @param fileName String The name of the pdf to create
 */
public void printChartToPDF(JFreeChart chart, int width,
                            int height,
                            String fileName) {
  try {
    Document document = new Document(new com.lowagie.text.
                                     Rectangle(
```

```
                                        width, height));

    PdfWriter writer = PdfWriter.getInstance(document,
                                             new
                                             FileOutputStream(
        fileName));
    document.addAuthor("Prestige model");
    document.open();
    PdfContentByte cb = writer.getDirectContent();
    PdfTemplate tp = cb.createTemplate(width, height);
    Graphics2D g2 = tp.createGraphics(width, height,
                                      new
                                      DefaultFontMapper());
    Rectangle2D rectangle2D = new Rectangle2D.Double(0, 0,
        width,
        height);
    chart.draw(g2, rectangle2D);
    g2.dispose();
    cb.addTemplate(tp, 0, 0);
    document.close();
  } catch (Exception e) {
    e.printStackTrace();
  }
}

/**
 * Export all the chart to separate pdf files
 */
public void printCharts() {
  printChartToPDF(avgCorrectChart, 400, 260,
                  expName + "avgCorrect" + ".pdf");
  printChartToPDF(avgCorrectGoodBadChart, 400, 260,
                  expName + "avgCorrectGoodBad" + ".pdf");
  printChartToPDF(energyPerTypeBarChart, 400, 260,
                  expName + "energyPerType" + ".pdf");
  printChartToPDF(beliefBarChart, 400, 260,
                  expName + "beliefs" + ".pdf");
  printChartToPDF(agentsPerCorrectBarChart, 400, 260,
                  expName + "agentsPerCorrect" + ".pdf");
  printChartToPDF(numberPerTimeChart, 400, 260,
                  expName + "numberPerTime" + ".pdf");
  printChartToPDF(avgEnBarChart, 400, 260,
                  expName + "avgEnergy" + ".pdf");
}

/**
 * is invoked whenever we press stop on a simulation
 * we output the all plots graph to a set of pdf files
 * with the name of the current experiment
 */
public void finish() {
  super.finish();
  //if not running experiments, set exp name to be the
  //expNo counter
  if (expName == null) expName = "" + expNo;

  if (writePdf) printCharts();

  //Notify the experiments class
  //that we 've finished (if running experiments)
  if (exp != null) {
    exp.expFinished();
  }
  //increment the experiment number (name)
  expNo++;
}

/***************Getter and setter methods to*****************
 *************** be used by the inspector*****************/

//***************Get and set number of food types
 public static int getnoThings() {
   return noThings;
 }
```

```
public static void setnoThings(int noThings) {
  Model.noThings = noThings;
}

//****************Get and set agents' lifespan
 public static void setLifespan(int lifespan) {
   Agent.lifespan = lifespan;
 }

public static int getLifespan() {
  return Agent.lifespan;
}

//*********Get and set initial number of agents
 public void setnoAgents(int noAgents) {
   this.noAgents = noAgents;
 }

public int getnoAgents() {
  return noAgents;
}

//***************Get the current number of agents
 public int getTotalagents() {
   return totalagents;
 }

//**************Get the number of agent per genotype
 public int getNoCopiers() {
   return agentTypes[Agent.NOCOPY];
 }

public int getAnyCopiers() {
  return agentTypes[Agent.COPYANY];
}
```

```
public int getPrestigeCopiers() {
  return agentTypes[Agent.COPYPRESTIGIOUS];
}

//**Get the number of agents who died since the sim has started
 public int getDied() {
   return died;
 }

//***********Get the number of agents who died from starvation
 public static int getStarvedDied() {
   return starvedDied;
 }

//************Get the number of agents who died of age
 public static int getNaturalDied() {
   return naturalDied;
 }


//*********Get and set the number of times agents communicate (copy)
 public int gettalkTime() {
   return talkTime;
 }

public void settalkTime(int times) {
  talkTime = times;
}

//***********Get and set the radius within which agent communicate
 public int getcomRegion() {
   return comRegion;
 }

public void setcomRegion(int communicationRegion) {
  comRegion = communicationRegion;
}
```

```
//**********Get and set whether to output charts to pdf upon stop
 public boolean getwritePdf() {
   return writePdf;
 }

public void setwritePdf(boolean write) {
  writePdf = write;
}

//********************Get the number of nutritious food types
 public static int getGoodThings() {
   return goodThings;
 }

//********************Get the number of unhealthy food types
 public static int getBadThings() {
   return badThings;
 }

//**********Get and set the number of super-agents to be injected
 public int getSuperAgents() {
   return superAgents;
 }

public void setSuperAgents(int num) {
  superAgents = num;
}

//*****Get and set the step in which super-agents are to be injected
 public int getInjectSuperAt() {
   return injectSuperAt;
 }

public void setInjectSuperAt(int time) {
  injectSuperAt = time;
}
```

```
//*******Get and set the frequency in steps that the world switches state
 public int getChangeWorldEvery() {
   return changeWorldEvery;
 }

public void setChangeWorldEvery(int every) {
  changeWorldEvery = every;
}

/*****Getter and setter methods for the constants ************
 *************** used to determine prestige******************/

public double getAgeImportance() {
  return ageImportance;
}

public double getEnergyImportance() {
  return energyImportance;
}

public double getOffspringImportance() {
  return offspringImportance;
}

public double getConformImportance() {
  return conformImportance;
}

public void setAgeImportance(double value) {
  ageImportance = value;
}

public void setEnergyImportance(double value) {
  energyImportance = value;
}
```

```
  public void setOffspringImportance(double value) {
    offspringImportance = value;
  }

  public void setConformImportance(double value) {
    conformImportance = value;
  }

  //*********Get and set the error (noise) in determining prestige
   public double getError() {
     return error;
   }

  public void setError(double error) {
    this.error = error;
  }

  //*********Get and set whether agents allow their models to eat first
   public boolean getDeference() {
     return deference;
   }

  public void setDeference(boolean def) {
    deference = def;
  }


}
```

D.2 Agent

```
package prestige;

import sim.engine.*;
import sim.util.*;
import java.awt.*;

/**
 *
 * <p>Title: Agent</p>
 *
 * <p>Description: Three different types of agents that move,
 * copy each other, depending on their type and eat resources
 * according to their "ideology" </p>
 *
 * @author Christos Bechlivanidis
 */


public class Agent extends sim.portrayal.simple.
    OvalPortrayal2D implements
    Steppable {

  /**************general information common to all agents*************/
  //the maximum level of energy
  public static final int maxEnergy = 100;
  //the age after which the age dies
  public static int lifespan = 50;
  /*****************************************************************/


  /**************the three types of agents***************************/
  //the agent does not copy other agents' beliefs
  public static final int NOCOPY = 0;
  //the agent copies beliefs from any agent in its environment
  public static final int COPYANY = 1;
  //the agent copies beliefs only from the prestigious agents
  public static final int COPYPRESTIGIOUS = 2;
  /*****************************************************************/


  /**************personal properties of the agent********************/
  //age calculated in timesteps
  public int age;
  //energy of the agent. cannot be lower than 0 or higer than the maximum
  public double energy;
  //the number of offspring this agent has
  public int offspring = 0;
  //the parent of this agent
  public Agent parent;
  //position/location of the agent on the grid
  public Int2D location;
  //the set beliefs the agents holds, i.e. which foods considers to be
  //nutritious (true) and which unhealthy
  public boolean[] ideology = new boolean[Model.noThings];
  //the times this agent has been copied by other agents
  public double timesCopied = 0;
  //the copying behaviour of the agent
  public int type;
  //whether the agent is a super-agent (lives longer)
  public boolean isSuper = false;


  /**
   * constructor, used only when the world is being populated (beggining):
   * (agents have no parents - age and energy determined by model)
   * @param curAge the initial age of the agent
   * @param curEnergy the initial energy of the agent
   */
  public Agent(int curAge, int curEnergy) {
    age = curAge;
    energy = curEnergy;
    parent = null;
```

```
}

/**
 * constructor used only when using the agent
 * as a portrayal... (to draw itself)
 */
public Agent() {
  super();
}

/**
 * constructor, used when agents are born.
 * does not place the agent in the world (grid), as parent does this.
 * @param SimState the simulation it is born into
 * @param Agent the parent
 */
public Agent(SimState state, Agent parent, double energy) {
  Model world = (Model) state;
  age = 0;
  this.energy = energy;
  this.parent = parent;
  //inherit the copying behaviour of the parent
  type = parent.type;

  //No lamarckian inheritance. Acquire knowledge randomly
  getBeliefs(world);

  //update statistics
  world.agentTypes[type]++;
  world.born++;
  world.totalagents++;
  location = world.grid.getObjectLocation(this);
}

/**
 * Give random beliefs to agents (a truth value for each food type)
 */
public void getBeliefs(Model world) {
  for (int k = 0; k < Model.noThings; k++) {
    ideology[k] = world.random.nextBoolean();
  }
}

/**
 * Draw the agent in the world. Give different colour
 * depending on the agent type
 */
public void draw(Object object, Graphics2D graphics,
                 sim.portrayal.
                 DrawInfo2D info) {

  switch (((Agent) object).type) {
  case NOCOPY:
    super.paint = Color.white;
    break;
  case COPYANY:
    super.paint = Color.gray;
    break;
  case COPYPRESTIGIOUS:
    super.paint = Color.blue;
    break;
  }
  super.draw(object, graphics,
             info);
}


/**
 * specifies actions to be taken at each time step.
 * @param SimState the current state
 */
public void step(SimState state) {
  //we cast the state to be a model, so that we can pass it to
  //the different methods. World is then passed around to the different
```

```
//methods so that the stats and the position can be updated
Model world = (Model) state;

//if it's a super-agent, execute other set of action and exit
if (isSuper) {
  superStep(world);
  return;
}

//first, depending on its type the agent
//may learn from someone else
Agent copied = copy(world);

//If he copied someone and they're
//both at the same place (and the model allows deference)
//, invite him to eat first
if (copied != null && copied.location.equals(location)
                                        && world.deference) {
  copied.eat(world);
}

//then agent eats
eat(world);

//then breeds with a certain probability
//provided he has enough energy
if (energy > 0 &&
    state.random.nextInt((int) energy + 1) > 50) {
  breed(world);
}

//Then moves (randomly)
move(world);

//updates the global variable location
location = world.grid.getObjectLocation(this);
  //loses some energy
  energy--;

  //then grow older
  age++;

  //finally agent dies if it reaches the life limit or if it starves
  if (age > lifespan || (energy <= 0)) {
    die(world);
  }
}

public void superStep(Model world) {
  //then agent eats
  eat(world);

  //loses some energy
  energy--;

  //Then moves (randomly)
  move(world);
  //updates the global variable location
  location = world.grid.
             getObjectLocation(this);

  //the grow older
  age++;

  //finally agent dies if it reaches the life limit or if it starves
  if (age > 100 || (energy <= 0)) {
    die(world);
  }

}


/**
```

```
 * Simulates the death of an agent. Updates the statistics of
 * the world and takes the agent out of the world's grid.
 * @param world
 */
public void die(Model world) {
  world.died++;
  world.totalagents--;
  world.agentTypes[type]--;
  if (energy < 0) {
    Model.starvedDied++;
  } else {
    Model.naturalDied++;
  }
  world.grid.remove(this);
}

/**
 * Simulates the communication betweem agents.
 * An agent depending on its type
 * may copy a number of beliefs from a neighbouring agent.
 * @param world
 * return the agent that was copied or null
 */
private Agent copy(Model world) {
  //The agent from which beliefs will be copied
  Agent toCopy = null;

  if (type == COPYPRESTIGIOUS) {
    //find the neighbour with the highest prestige
    toCopy = getMostPrestigious(world);

    //Add to the agent to copy the difference between his prestige
    //and this agent's prestige
    if (toCopy != null) {
      toCopy.timesCopied++; //(toCopy.getPrestige(world) -
      //getPrestige(world));
    }
  } else if (type == COPYANY) {
    //randomly copy one of the neighbouring agents
    toCopy = getRandomNeighbour(world);
  }
  if (toCopy != null) {
    //For the duration of talkTime randomly select a belief
    //and copy it
    for (int i = 0; i < Model.talkTime; i++) {
      int k = world.random.nextInt(Model.noThings);
      ideology[k] = toCopy.ideology[k];
    }
  }
  return toCopy;
}

/**
 * Find the neighbour with the highest prestige
 * @return the neighbour with the highest prestige or null if no neighbours
 * are there or if no neighbour has higher prestige than our agent
 */
private Agent getMostPrestigious(Model world) {
  //to hold all neighbours
  Bag n = new Bag();

  //get all neighbours in a given radius (com_region)
  world.grid.getNeighborsHexagonalDistance(location.x,
                                           location.y,
                                           Model.
                                           comRegion, true,
                                           n, null, null);

  //the highest prestige value among the neighbours. Instantiate with
  //this agent's prestige
  double top_prestige = getPrestige(world);
  //The agent with the highest prestige among the neighbours
  Agent best = null;
  //a neighbour to consider
```

```
  Agent a;
  //the prestige of the neighbour under consideration
  double prestige;
  while (!n.isEmpty()) {
    a = (Agent) n.pop();
    //possible mistake in error in evaluating prestige
    double error = world.random.nextDouble() *
                   world.error;
    //whether the agent will be over or under-estimated (random)
    boolean plus = world.random.nextBoolean();
    prestige = a.getPrestige(world) +
               (plus ? error : -error);

    if (prestige > top_prestige) {
      top_prestige = prestige;
      best = a;
    }
  }
  return best;
}


/**
 * Find a random agent in the neighbourhood
 * @param world Model The world this agent inhabits
 * @return Agent A random neighbour
 */
private Agent getRandomNeighbour(Model world) {
  //to hold all neighbours
  Bag n = new Bag();

  //get all neighbours in a given radius (comRegion)
  world.grid.getNeighborsHexagonalDistance(location.x,
                                           location.y,
                                           Model.
                                           comRegion, true,
                                           n, null, null);

  Agent neighbour = null;

  //Randomly select one of the neighbours (provided there are some)
  if (!n.isEmpty()) {
    neighbour = (Agent) n.get(world.random.nextInt(n.
        size()));
  }
  return neighbour;
}

/**
 * Simulates the agent feeding on the patches in the world.
 * Agents will only eat patches of food that they believe
 * to be good for them (and on which they are standing).
 * @param world
 */
private void eat(Model world) {
  if (energy < maxEnergy) {
    //get the food type of whatever is on the patch
    int foodType = world.patches.field[location.x][
                   location.y];

    //if there is food on the patch...
    if (foodType != 0) {
      //...and believes that it's nutritious, eat it and add
      // (or subract) energy
      if (ideology[foodType - 1]) {
        energy += world.getFoodValue(foodType);

        //empty the patch from food
        world.patches.field[location.x][location.y] =
            0;
      }
    }
  }
}
```

```
/**
 * Simulates the agent's breeding. Adds another agent to the model, with
 * the current agent as a parent
 * (child is the same genotype as the parent
 * but does not inherit any beliefs).
 * Agent loses energy when breeding, it is shared 80%:20%.
 * The child is placed on the same patch as its parent.
 * @param world
 */
private void breed(Model world) {
  //the energy to be given to the offspring
  double energyTransferred = energy * 0.2;
  //lose 20% because of giving birth
  energy -= energyTransferred;
  //update the number of offsprings
  offspring++;
  //create a new agent
  Agent child = new Agent(world, this,
                          energyTransferred);

  //child placed in same patch
  world.grid.setObjectLocation(child, location);
  child.location = location;
}


/**
 * Simulates the agent's movements at each time step. Movement is random
 * @param world
 */
private void move(Model world) {
  //the distance agents will travel
  int dist = Model.stepLength;
  //the angle of the movement (as an integer between 0 and 8)
  int angle = (int) world.random.nextInt(360) / 45;
  //the relative distance moved on the x axis
  //and on the y axis from current position

  double dx = 0; //horizontal movement
  double dy = 0; //vertical movement
  switch (angle) {
  case 0:
    dx = 1;
    dy = 0;
    break;
  case 1:
    dx = 1;
    dy = 1;
    break;
  case 2:
    dx = 0;
    dy = 1;
    break;
  case 3:
    dx = -1;
    dy = 1;
    break;
  case 4:
    dx = -1;
    dy = 0;
    break;
  case 5:
    dx = -1;
    dy = -1;
    break;
  case 6:
    dx = 0;
    dy = -1;
    break;
  case 7:
    dx = 1;
    dy = -1;
```

```
    break;
  }
  //sets the new position of the agent on the grid.
  //Take care to wrap around, and look out for negative indecies.
  int x, y;
  x = (location.x + (int) (dx * dist)) %
      world.gridWidth;
  y = (location.y + (int) (dy * dist)) %
      world.gridHeight;
  x = (x < 0) ? x + world.gridWidth : x;
  y = (y < 0) ? y + world.gridHeight : y;
  world.grid.setObjectLocation(this, x, y);
}


/**
 * The prestige of the agent depends on its age,
 * energy and number of offspring. Also depends on the number of
 * times this agent has been copied by others
 * @return double the agent's prestige
 */
public double getPrestige(Model model) {
  return model.ageImportance *
      ((double) age / (double) lifespan) +
      model.energyImportance * (energy / maxEnergy) +
      model.offspringImportance * offspring +
      model.conformImportance * timesCopied;
}


/**********************************************************************
 *
 * **********getter methods used for the inspector******************
 *
 **********************************************************************/
```

```
/**
 * @return Returns the age.
 */
public int getAge() {
  return age;
}

/**
 * @return Returns the energy.
 */
public double getEnergy() {
  return energy;
}


/**
 * What the agent believes its nutritious
 * @return boolean[] a truth/false value for each food type
 */
public boolean[] getIdeology() {
  return ideology;
}

/**
 * The way this agent copies beliefs from its environments
 * @return String A string to be used in the inspector
 */
public String getType() {
  switch (type) {
  case NOCOPY:
    return "No copy";
  case COPYANY:
    return "Copy anyone";
  case COPYPRESTIGIOUS:
    return "Copy prestigious";
  default:
    return "wrong type";
```

```
    }
  }

  /**
   * The number of children this agent has given birth to
   * @return int number of children
   */
  public int getOffsrping() {
    return offspring;
  }

  /**
   * The number of times this agent has been copied
   * @return double
   */
  public double getTimesCopied() {
    return timesCopied;
  }


}
```


D.3 Viewer

```
package prestige;

import sim.engine.*;
import sim.display.*;
import sim.portrayal.grid.*;
import sim.util.gui.SimpleColorMap;
import java.awt.*;
import javax.swing.*;
import org.jfree.chart.ChartPanel;

/**
 *
 * <p>Title: Viewer</p>
 *
 * <p>Description: Creates the control panel,
 * starts the simulation and creates the frames
 * to display the charts</p>
 *
 * @author Christos Bechlivanidis
 */


public class Viewer extends GUIState {

  //The control panel
  public Display2D display;
  //the frame to hold the panel
  public JFrame displayFrame;

  //The frames to show the charts
  public JFrame noAgentsBarchartFrame;
  public JFrame avgEnBarchartFrame;
  public JFrame enPerTypeBarchartFrame;
  public JFrame agentsPerCorrectBarchartFrame;
  public JFrame avgCorrectChartFrame;
  public JFrame numberChartFrame;
  public JFrame avgCorrectGoodBadChartFrame;

  //The 2D portrayal of the world
  FastValueGridPortrayal2D patchPortrayal = new
      FastValueGridPortrayal2D("Patches");
  SparseGridPortrayal2D gridPortrayal = new
      SparseGridPortrayal2D();
  //The model to run
  public Model model;

  /**
   * Constructor
```

```
 */
public Viewer() {
  super(new Model(System.currentTimeMillis()));
}

/**
 * Constructor
 * @param state SimState
 */
public Viewer(SimState state) {
  super(state);
}

/**
 * Information to be displayed by
 * the control panel
 */
public String getName() {
  return "The role of prestige in knowledge spread";
}

public String getInfo() {
  return
      "<H2>" + getName() + "</H2>" +
      "<p>by Christos Bechlivanidis";
}

/**
 * Sets up Portrayals
 * */
public void setupPortrayals() {
  // tell the portrayals what to portray and how to portray them
  patchPortrayal.setField(((Model) state).patches);
  //Colours of food patches
  Color colours[] = new Color[Model.noThings + 1];
  //empty patch
  colours[0] = Color.black;
  for (int i = 1; i < colours.length; i++) {
    if (i <= 3) {
      //unhealthy foods are red
      colours[i] = Color.red;
    } else {
      //nutritious foods are yellow
      colours[i] = Color.yellow;
    }
  }
  patchPortrayal.setMap(new SimpleColorMap(colours));

  gridPortrayal.setField(((Model) state).grid);
  gridPortrayal.setPortrayalForAll(new Agent());

  display.reset();
  display.repaint();
}

/*called when the simulation is started*/
public void start() {
  super.start();
  setupPortrayals();
}

/**
 * Called when frame is closed
 */
public void quit() {
  super.quit();

  if (displayFrame != null)
    displayFrame.dispose();
  displayFrame = null;
  display = null;
}

//Load the simulation
```

```
public void load(SimState state) {
  super.load(state);
  setupPortrayals();
  // make new frame and register it
  addChartPanel(controller);

}

/*used to control the simulation, to update the charts and
 also lets us show hide the frames that contain the charts*/
public void init(Controller c) {
  super.init(c);

  //making the 2D display
  Model st = (Model) state;
  display = new Display2D(st.gridWidth * 3,
                          st.gridHeight * 3, this, 1);
  displayFrame = display.createFrame();
  c.registerFrame(displayFrame);
  display.attach(gridPortrayal, "Language");
  display.setBackdrop(Color.black);
  display.attach(patchPortrayal, "Patches");
  display.attach(gridPortrayal, "Agents");
  addChartPanel(c);
}

/**
 * @return displays the charts
 */
public void addChartPanel(Controller c) {
  ((Model) state).createCharts();
  ChartPanel numAgentsBarchartPanel = new ChartPanel(((
      Model) state).
      beliefBarChart);
  ChartPanel avgEnBarchartPanel = new ChartPanel(((Model)
      state).
      avgEnBarChart);
```

```
ChartPanel enPerTypeBarchartPanel = new ChartPanel(((
    Model) state).
    energyPerTypeBarChart);

ChartPanel agentsPerCorrectBarchartPanel = new
                                          ChartPanel(((
    Model)
    state).agentsPerCorrectBarChart);
ChartPanel avgCorrectChartPanel = new ChartPanel(((
    Model) state).
    avgCorrectChart);
ChartPanel avgCorrectGoodBadChartPanel = new ChartPanel(((
    Model) state).
    avgCorrectGoodBadChart);
ChartPanel numberChartPanel = new ChartPanel(((Model)
                                           state).
                                          numberPerTimeChart);

enPerTypeBarchartFrame = new JFrame();
enPerTypeBarchartFrame.setResizable(true);
enPerTypeBarchartFrame.getContentPane().setLayout(new
    BorderLayout());
enPerTypeBarchartFrame.getContentPane().add(
    enPerTypeBarchartPanel,
    BorderLayout.CENTER);
enPerTypeBarchartFrame.setTitle(
    "Average energy per agent type");
enPerTypeBarchartFrame.pack();

noAgentsBarchartFrame = new JFrame();
noAgentsBarchartFrame.setResizable(true);
noAgentsBarchartFrame.getContentPane().setLayout(new
    BorderLayout());
noAgentsBarchartFrame.getContentPane().add(
    numAgentsBarchartPanel,
    BorderLayout.CENTER);
noAgentsBarchartFrame.setTitle(
```

```
    "Number of Agents per belief");
noAgentsBarchartFrame.pack();

avgEnBarchartFrame = new JFrame();
avgEnBarchartFrame.setResizable(true);
avgEnBarchartFrame.getContentPane().setLayout(new
    BorderLayout());
avgEnBarchartFrame.getContentPane().add(
    avgEnBarchartPanel,
    BorderLayout.CENTER);
avgEnBarchartFrame.setTitle(
    "Average Energy per Number of correct beliefs");
avgEnBarchartFrame.pack();

agentsPerCorrectBarchartFrame = new JFrame();
agentsPerCorrectBarchartFrame.setResizable(true);
agentsPerCorrectBarchartFrame.getContentPane().
    setLayout(new
              BorderLayout());
agentsPerCorrectBarchartFrame.getContentPane().add(
    agentsPerCorrectBarchartPanel, BorderLayout.CENTER);
agentsPerCorrectBarchartFrame.setTitle(
    "Number of agents per correct beliefs");
agentsPerCorrectBarchartFrame.pack();

avgCorrectChartFrame = new JFrame();
avgCorrectChartFrame.setResizable(true);
avgCorrectChartFrame.getContentPane().setLayout(new
    BorderLayout());
avgCorrectChartFrame.getContentPane().add(
    avgCorrectChartPanel,
    BorderLayout.CENTER);
avgCorrectChartFrame.setTitle(
    "Average number of correct beliefs per agent type");
avgCorrectChartFrame.pack();

numberChartFrame = new JFrame();
numberChartFrame.setResizable(true);
numberChartFrame.getContentPane().setLayout(new
    BorderLayout());
numberChartFrame.getContentPane().add(numberChartPanel,
                                      BorderLayout.
                                      CENTER);
numberChartFrame.setTitle(
    "Number of agents per time step");
numberChartFrame.pack();

avgCorrectGoodBadChartFrame = new JFrame();
avgCorrectGoodBadChartFrame.setResizable(true);
avgCorrectGoodBadChartFrame.getContentPane().setLayout(new
    BorderLayout());
avgCorrectGoodBadChartFrame.getContentPane().add(
    avgCorrectGoodBadChartPanel,
    BorderLayout.CENTER);
avgCorrectGoodBadChartFrame.setTitle(
    "Number of agents, correct in bad/1st bad");
avgCorrectGoodBadChartFrame.pack();

//registering the frames so that we can hide it
c.registerFrame(noAgentsBarchartFrame);
c.registerFrame(avgEnBarchartFrame);
c.registerFrame(enPerTypeBarchartFrame);
c.registerFrame(agentsPerCorrectBarchartFrame);
c.registerFrame(numberChartFrame);
c.registerFrame(avgCorrectChartFrame);
c.registerFrame(avgCorrectGoodBadChartFrame);

}

/**
 * returns the global inspector for the simulation.
 * @return the model's inspector
 */
public Object getSimulationInspectedObject() {
```

```
    return state;
  }

  /**
   * Used to tell the inspector for the model to update at each step,
   * change to false in order to speed up the simulation (but no
   * possibility of seeing statistics then).
   * @return true if the inspector updates at each step, false otherwise
   */
  public boolean isVolatile() {
    return true;
  }

  public static void main(String args[]) {
    Viewer v = new Viewer();
    Console c = new Console(v);
    c.setVisible(true);
  }

}
```

D.4 Experiments

```
package prestige;

import java.util.*;
import sim.display.Console;

/**
 * <p>Title: Experiments</p>
 *
 * <p>Description: This is a class used to  "silently"
 * run a number of simulations one after the other
 * and output the graphs with different names.
 * What is shown here is the set of values used to run
 * all the simulations 10 times each</p>
 *
 * @author Christos Bechlivanidis
 *
 */
public class Experiments {
  //How many times to run each
  private static int howMany = 10;
  //The current experiment number in the series
  int currentId = 0;
  //The names of the experiments
 private String names[] = new String[] {
      "EnergyOnly","OffspingOnly", "EnergyOffsping",
      "AgeOnly", "EnergyAge", "OffspringAge", "Conf0",
      "Conf005", "Conf08", "Conf2","AgeConf", "Food10",
      "Food40", "Food80", "Food120", "Food160","switch1500",
      "switch1000", "switch300","switch100",  "noise01",
      "noise03", "noise05", "noise1","Super", "NoDeference",
      "NoPrestigiousDef","NoPrestigiousNoDef"};

  //The values to be set for each set of experiments.
  private double values[][] = new double[][] {
  {0.7, 0, 0, 0, 0, 60, 10000000, -1, 1},  //en
  {0, 0.7, 0, 0, 0, 60,10000000, -1, 1}, //offsping
  {0.7, 0.7, 0, 0, 0, 60, 10000000, -1, 1},  //enOff
  {0, 0, 0.7, 0, 0, 60,10000000, -1, 1}, //age
  {0.7, 0,0.7, 0, 0, 60, 10000000, -1, 1},//enage
  {0, 0.7, 0.7, 0, 0, 60, 10000000, -1, 1}, //offage
  {0.7, 0.2, 0.5, 0, 0, 60, 10000000, -1, 1}, //con0
  {0.7, 0.2, 0.5, 0.005, 0, 60, 10000000, -1, 1},//con005
  {0.7, 0.2, 0.5, 0.08, 0, 60, 10000000, -1, 1}, //con08
  {0.7, 0.2, 0.5, 0.2, 0, 60, 10000000, -1, 1}, //con2
  {0.0, 0.0, 0.7, 0.1, 0, 60, 10000000, -1, 1}, //agecon
  {0.7, 0.2, 0.5, 0.01, 0, 10,10000000, -1, 1}, //food 10
  {0.7, 0.2, 0.5, 0.01, 0, 40, 10000000,-1, 1}, //food 40
  {0.7, 0.2, 0.5, 0.01, 0, 80, 10000000, -1, 1}, //food 80
  {0.7, 0.2, 0.5, 0.01, 0, 120,10000000, -1, 1}, //food 120
  {0.7, 0.2, 0.5, 0.01, 0, 160, 10000000,-1, 1}, //food 160
  {0.7, 0.2, 0.5, 0.01, 0, 60, 1500, -1, 1}, //switch 1500
  {0.7, 0.2, 0.5, 0.01, 0, 60, 1000,-1, 1}, //switch 1000
  {0.7, 0.2, 0.5, 0.01, 0, 60, 300, -1, 1}, //switch 300
  {0.7, 0.2, 0.5, 0.01, 0, 60, 100, -1, 1}, //switch 100
  {0.7, 0.2, 0.5, 0.01, 0.1,  60, 10000000, -1, 1}, //noise01
  {0.7, 0.2, 0.5, 0.01, 0.3, 60,10000000, -1, 1}, //noise 03
  {0.7, 0.2, 0.5, 0.01, 0.5, 60, 10000000,-1, 1}, //noise 05
  {0.7, 0.2, 0.5, 0.01, 1, 60, 10000000, -1, 1},//noise 1
  {0.7, 0.2, 0.5, 0.01, 0.1, 60,10000000, 4000, 1}, //super
  {0.7, 0.2,0.5, 0.01, 0.1, 60, 10000000,-1, 0} //noDef
  };

  //Writers to be used for writing data to txt files
  // with format suitable for a latex file
  static DataWriter beliefsWriter = new DataWriter(
      "beliefs.txt");
  static DataWriter numbersWriter = new DataWriter(
      "numbers.txt");

  TimerTask t = null;
  //The duration of each experiment
  public long duration = 8000;
```

```
//The current index of this set of sim
public int expIndex = 1;
//the counter of all the experiments run so far
public int counter = 1;
//The current experiment name
public String currentExpName;

//The model, viewer and console of the experiments
private Model m = new Model(System.currentTimeMillis(), this);
private Viewer v = new Viewer(m);
private Console c = new Console(v);

public static void main(String[] args) {
  new Experiments().startNextExp();
  beliefsWriter.newLine();
  numbersWriter.newLine();
}


public void startNextExp() {
  t = new TimerTask() {
    public void run() {
      if (c.getPlayState() == c.PS_STOPPED) {
        //Write data to files every n number of steps
        m.writeDataEvery = 400;
        //Set all the values for this experiment
        currentExpName = names[currentId] + "_" +
                         expIndex;
        beliefsWriter.startExp(currentExpName);
        numbersWriter.startExp(currentExpName);
        m.energyImportance = values[currentId][0];
        m.offspringImportance = values[currentId][1];
        m.ageImportance = values[currentId][2];
        m.conformImportance = values[currentId][3];
        m.error = values[currentId][4];
        m.noThings = (int) values[currentId][5];
        m.changeWorldEvery = (int) values[currentId][6];
        m.injectSuperAt = (int) values[currentId][7];
        m.deference = ((int) values[currentId][8]) == 0 ? false : true;

        m.expName = currentExpName;
        c.setWhenShouldEnd(duration);
        c.pressPlay();
        t.cancel();
      }
    }
  };
  new Timer().schedule(t, 0, 5);
}

/**
 * Called by model when a simulation is finished.
 * Outputs a relevant message and starts the
 * next experiment if any, otherwise exits
 */
public void expFinished() {
  System.out.println("Finished: " + currentExpName);
  beliefsWriter.endExp();
  numbersWriter.endExp();
  //Go to the next line of the tex table every 3 records
  if (counter % 3 == 0) {
    beliefsWriter.newLine();
    numbersWriter.newLine();
  }
  counter++;
  if (expIndex == howMany) {
    expIndex = 1;
    currentId++;
  } else {
    expIndex++;
  }

  if (names.length > currentId) {
    startNextExp();
```

```
    } else {
        beliefsWriter.endAll();
        numbersWriter.endAll();
        System.exit(0);
    }
}
}
```

D.5 DataWriter

```
package prestige;

import java.io.*;

/**
 * <p>Title: DataWriter</p>
 * <p>Description: A helper class
 * for writing data from experiments
 * into text files in a format
 * suitable for inclusion in latex tables</p>
 *
 * @author Christos Bechlivanidis
 *
 */

public class DataWriter {
  //The output stream (to a file)
  private OutputStreamWriter out;

  /**
   * Constructor
   * @param filename String The file where the
   */
  public DataWriter(String filename) {
    try {
      out = new OutputStreamWriter(new FileOutputStream(
          filename));
    } catch (FileNotFoundException ex) {
      System.out.println("Cannot open file \n" +
                         ex.getMessage());
    }
  }

  /**
   * Output a newline sequence, and add the header of the next line
   */
  public void newLine() {
    try {
      out.write("\\\\ \\hline ");
      out.write("Steps 400 800 1200 1600 2000 2400 2800 " +
                "3200 3600 4000 4400 4800 5200 5600 6000 6400 6800 7200 7600 &");
    } catch (Exception ex) {
      System.out.println("Cannot end experiment \n" +
                         ex.getMessage());
    }

  }

  /**
   * Denote the start of an experiment by simply writing its title
   * @param expName String
   */
  public void startExp(String expName) {
    try {
      out.write(expName + " ");

    } catch (Exception ex) {
      System.out.println("Cannot start experiment \n" +
                         ex.getMessage());
    }
  }

  /**
   * Output the sequence to go to the next column of the table
   */
  public void endExp() {
    try {
      out.write("&");
      out.flush();
    } catch (Exception ex) {
      System.out.println("Cannot end experiment \n" +
                         ex.getMessage());
```

```
  }

}

/**
 * Output data from the experiment
 * @param data String The data in a String
 */
public void writeData(String data) {
  try {
    out.write(data + " ");
  } catch (IOException ex) {
    System.out.println("Cannot write data \n" +
                       ex.getMessage());
  }

}
```

```
  /**
   * Close the output stream
   */
  public void endAll() {
    try {
      out.flush();
      out.close();
    } catch (IOException ex) {
      System.out.println("Cannot close file \n" +
                         ex.getMessage());
    }
  }


}
```