Evolutionary Mechanisms

Edmund Chattoe and Bruce Edmonds

Introduction

There are now many simulations of complex social phenomena that have structures or component processes analogous to biological evolution (see Arifovic 1994, Chattoe 2006a, Dosi et al 1999, Lomborg 1996, Nelson and Winter 1982, Oliphant 1996, Windrum and Birchenhall 1998 to get a flavour of the diversity of approach and applications). Clearly the process of biological evolution is complex and has resulted in the development of complex (and in several cases social) systems. However biological evolution follows very specific mechanisms and is clearly not strictly isomorphic with social processes. For a start biological evolution occurs over larger time spans than most social processes. Further, it is unlikely, as sociobiology (Wilson 1975) and evolutionary psychology (Buss 2004) are sometimes supposed to imply, that the domain of social behaviour will actually prove reducible to genetics. Thus it is not immediately apparent *why* evolutionary ideas have had such an influence upon the modelling of social processes. However simulations of social phenomena have been strongly influenced by our understanding of biological evolution and this has occurred via two main routes: through *analogies* with biological evolution and through computer science approaches.

In the first case, conceptions of evolution have been used as a way of understanding social processes and then simulations have been made using these conceptions. For example (Nelson and Winter 1982) modelled growth and change in firms using the idea of random variation (new products or production processes) and selective retention (whether these novelties in fact sustain profitability – the survival requirement for firms – in an environment defined by what other firms are currently doing).

In the second case computer science has taken up the ideas of evolution and applied it to engineering problems. Most importantly in Machine Learning, ideas from biological evolution have inspired whole families of techniques in what has become known as “Evolutionary Computation”. The most famous of these techniques are Genetic Algorithms (Holland 1975, Mitchell 1996) and Genetic Programming (Koza 1992a, 1994) discussed below. These algorithms have then been applied in social simulations with different degrees of adaption; from using them unchanged as “off the shelf” plug-ins (for example to model learning processes) to specifying simulation processes that use the core evolutionary idea but are completely re-engineered for a particular modelling purpose or domain. There is no *a priori* reason to suppose that a particular technique from computer science will be the most appropriate algorithm in a social simulation (including those with a biological inspiration) as we shall see below, but it certainly presents a wealth of evolutionary ideas and results that are potentially applicable in some form. Like any theory, the trick is to use good judgement and a clear specification in applying an algorithm to a particular social domain (Chattoe 1998, 2006b).

What is certainly the case is that biological evolution offers an example of how complex and self-organised phenomena can emerge from randomness, so it is natural to look to this as a possible conceptual framework with which to understand social phenomena with similar properties. (In particular, while it may be reasonable to assume deliberation and rationality in some social contexts,

it is extremely unlikely to apply to all social structures and phenomena. As such, some kind of blind variation and retention – evolution – is probably the only well defined theoretical alternative.) The extent to which evolution-like processes are generally applicable to social phenomena is unknown (largely because this foundational issue has not received much attention to date), but these processes certainly are a rich source of ideas and it may be that there are some aspects of social complexity that will prove to be explicable by models thus inspired. It is already the case that many social simulation models have taken this path and thus have the potential to play a part in helping us to understand social complexity (even if they only serve as horrible examples).

This chapter looks at some of the most widely used approaches to this kind of modelling, discusses others, gives examples and critically discusses the field along with areas of potential development.

An Abstract Description of Biological Evolution

We will not provide full details of biological evolution as currently understood in the Neo-Darwinian synthesis.[1] Rather we will take from this a generalised model of evolution that will potentially cover a variety of social processes. This description will then be used to discuss an example from Evolutionary Game Theory (Vega-Redondo 1996). This will unpack the implications of the abstract description and demonstrate its generality. This generalisation is a preliminary to discussing evolutionary simulations of social phenomena based on the abstract description as a framework.

The Four Process Description

The basic components in the biological theory are the genotype (the set of instructions or genome) and the phenotype (the "body" which the genotype specifies) in which these instructions are embedded. The phenotype is constructed using "instructions" encoded in the genotype. The phenotype has various capabilities including reproduction. Maintenance of the phenotype (and the embedded genotype) requires a number of potentially scarce inputs (food, water). The phenotypic capabilities include management of inputs and outputs to the organism. Poor adaptation of these capabilities with respect to either external or internal environment will result in malfunction and consequent death. The death of a particular phenotype also ends its reproductive activity and removes the corresponding genotype from the population. Variation occurs by mutation and during reproduction, giving rise to novel genotypes (and hence subsequent phenotypes) in the resulting offspring. Genotypic variations are not selected directly by the environment but according to the overall capabilities of the phenotype. In biology, phenotype alterations cannot be transmitted to the genotype for physiological reasons but in social systems this "Lamarkian" adjunct to evolution (which is not, however, adequate to explain change in its own right) is both possible and plausible. In particular it allows for combinations of evolutionary learning and the social level and deliberate action at the individual level (Chattoe 2006a).

A full specification of an evolutionary model requires descriptions of the following processes:

1) Generation of phenotypes: A specification of the genotypes and the phenotypes these correspond to. This may not specify a 1-1 mapping between genotypes and phenotypes but describe the process by which phenotypes are actually constructed from genotypes. This is necessary when genotypes cannot be enumerated.

[1] For details about this see any good textbook on biology (e.g. Dobzhansky et al 1977).

2) Capabilities of the phenotypes: A specification of ways in which phenotypes may use their capabilities to affect the internal and external environment, including the behaviour and numbers of other phenotypes. Lamarckian systems include the capability to modify the genotype using environmental feedback during the lifetime of the phenotype.
3) Mechanisms of reproduction and variation: A specification of the process by which phenotypes reproduce including possible differences between ancestor and successor genotypes resulting from reproduction. Reproduction may involve a single ancestor genotype (parthenogenesis) or a pair (sexual reproduction). In principle, multiple parents could be modelled if appropriate for particular social domains (like policies decided by committees) though this approach has not been used so far.
4) Mechanism of selection: A specification of all the processes impinging on the phenotype and their effects. This is the converse of the second mechanism, the capabilities of one phenotype form part of the selection process for the others. Some processes, such as fighting to the death, can be seen as directly selective. However, even indirect processes like global warming may interact with phenotypic capabilities in ways that affect fitness.

In these process specifications it may be convenient to distinguish (and model separately) the "environment" as the subset of objects impinging on phenotypes which display no processes of the first three types. Whether a separate representation of the environment is useful depends on the process being modelled. At one extreme, a person in a desert is almost exclusively dealing with the environment. At the other, rats in an otherwise empty cage interact almost entirely with each other.

Obviously, some of these specifications could be extremely complex depending on the system being modelled. The division into system components is necessarily imprecise but not arbitrary. It is based on the considerable observed integrity of organisms relative to their environment. (This integrity is also observed in social "organisms" like firms which have clearly – and often legally – defined boundaries.) The first and third specifications involve processes internal to the organism while the second and fourth represent the organisms effect on the external world and the converse.

Of course, social processes, even "evolutionary social processes" are not constrained by the above specification, for example what most closely corresponds to the genotype might not be separable from what corresponds to the phenotype. Nevertheless, however, for a very broad class of evolutionary simulations it will be necessary to implement something very similar to the above four categories.

Illustrative Example: A Simple Evolutionary Game

Despite the potential complexity of specifying complete models for biological systems, this description can also be used to clarify and analyse relatively simple evolutionary systems. In this section we shall provide a description for an evolutionary game. The purpose is not to comment on Evolutionary Game Theory *per se* but to show how the description raises issues relevant to our understanding of evolutionary models.

For each agent, the genotype is one of a set of finite state automata producing a single action in each period, for example the complete set of one and two-state automata leading to the actions "Co-operate" and "Defect" in a Prisoner's Dilemma (see, for example, Lomborg 1996). The action is compared with the action of a co-player (another agent) and the result is an adjustment to the

“energy level” for each agent depending on the game payoffs and chosen strategies. If agents reach a certain energy level, they produce an exact copy. (This model dispenses with variation and involves asexual reproduction.) If the energy level of any agent reaches zero, it dies and is removed from the environment. Reproduction reduces the energy level considerably. Merely existing also does so but at a much lower rate.

With some qualifications, this is an example of a complete description discussed in the last section. It reveals some interesting things about the process of constructing such descriptions.

Firstly, this model involves a very attenuated environment compared to real social systems. Agents have a single external capability involving one of two actions and thus affecting the energy levels of their co-players. The effect of these actions is also the only environmental process that impinges on agents. The model of the environment just consists of mechanisms for deciding when and which agents will play, administering actions and energy changes, producing copies and removing dead agents. In real social systems, exogenous events (both social and environmental) are likely to be very important.

Secondly, the discussion of energy levels still sounds biological but this is simply to make the interpretation of the example more straightforward in the light of the recent discussion. As I shall show subsequently, the survival criterion can just as easily be profitability or organisation membership levels.

Thirdly (and perhaps most importantly) there has been sleight of hand in the description of the model. I have already described Lamarckism (genotype modification by the phenotype during the organism’s lifetime) and the construction of the phenotype by the genotype during gestation is a fundamental part of the evolutionary process. But in this example the genotype is effectively “reconstructing” the phenotype every time the finite state automaton generates an action. There is nothing observable about a particular agent, given the description above, except the sequence of actions they choose. There is no way for an agent to establish that another is actually the same when it plays D on one occasion and C on another, or that two plays of D in successive periods actually come from two different agents. In fact, this is the point at which models of social evolution develop intuitively from the simple description of biological evolution used so far. The capabilities of social agents (such as consumers, families, churches and firms) include the “senses” that give them the ability to record actions and reactions in memory. Furthermore, they have mental capabilities that permit the processing of sense data in various ways, some subset of which we might call rationality. The simple automata described above are reactive, in that their actions depend in systematic ways on external stimuli, but they can hardly be said to be rational or reflective in that their “decision process” involves no choice points, internal representations of the world or “deliberation”. Such distinctions shelve into the deep waters of defining intelligence, but the important point is that we can make useful distinctions between different kinds of adaptability based on the specifications of process we use in our models without compromising the evolutionary framework we have set up. It is in this way that the complex relationship between selection and reasoned action may begin to be addressed.

Thus, even in this simple example, one can see not only the general evolutionary structure of the simulation but also that the conception differs in significant ways from the corresponding biological process.

Evolutionary Ideas in the Social Sciences

From early on, since the publication of “The Origin of Species” (Darwin 1859), Darwin’s ideas of evolution have influenced those who have studied social phenomena. In 1884 Gabriel Tarde published a paper (Tarde 1884) discussing “natural” and “social” Darwinism. This marked a shift from looking at the social organisation of individuals to the patterns of social products (fashions, ideas, tunes, laws and so on). Tarde (1903 p.74) put it like this:

"but self-propagation and not self-organisation is the prime demand of the social as well as of the vital thing. Organisation is but the means of which propagation, of which *generative* or *imitative* imitation, is the end."

However it was from the latter half of the 20th Century that the full force of the analogy with biological evolution (as understood in the Neo-Darwinian Synthesis) was felt in the social sciences. There were those who sought to understand the continuous change in cultural behaviours over long time-scales in this way, e.g. (Boyd and Richerson 1984, Campbell 1965, Cavalli-Sforza and Feldman 1973, Cloak 1975, Csanyi 1989). Richard Dawkins coined the term 'meme' as an discrete and identifiable unit of cultural inheritance corresponding to the biological gene (Dawkins 1976, 1982), an idea which has influence a stream of thinkers including (Costall 1991, Lynch 1996, Dennett 1990, Heyes and Plotkin 1989, Hull 1982, 1988, Westoby 1994). Another stream of influence has been the Philosophy of Science via the idea that truth might result from the evolution of competing hypotheses (Popper 1979), a position known as Evolutionary Epistemology since (Campbell 1974). The ultimate reflection of the shift described by Tarde above is that the human mind is “merely” the niche where memes survive (Blackmore 1999) or exploit (as “viruses of the mind” – Dawkins 1993) – the human brain is programmed by the memes, rather than using them (Dennett 1990). This fits in with the idea of the Social Intelligence Hypothesis (Kummer et al 1997) that the biological reason the brain evolved is because it allows specific cultures to develop in groups giving specific survival value with respect to the ecological niches they inhabit (Reader 1970). All of these ideas hinge on the importance of imitation (Dautenhahn and Nehaniv 2002), since without this individual memes, ideas or cultural patterns would be quickly lost.

Evolutionary theories are applied in a wide variety of disciplines. As mentioned above, evolutionary theories are applied to culture and anthropology, as in the work of Boyd and Richerson, Cavalli-Sforza and Feldman and Csanyi. The evolution of language can be seen as an analogy to biological evolution, as described by (Hoenigswald and Wiener 1987). In computer science, Genetic Programming and Genetic Algorithms (as well as the more rarely used Classifier Systems) are descendants of the evolutionary view as well, for example in the work of several individuals at the Santa Fe Institute (Holland 1975, Kauffman 1993). Learning theories of humans, applied to individuals, groups and society can be tied to evolutionary theory, as shown in the work of Campbell (1965, 1974). The work of several philosophers of science also shows an evolutionary perspective on knowledge, as in Popper's (1979) and Kuhn's (1970) work. In addition, these views have impact on evolutionary epistemology, and are analogical to biological evolution. Evolutionary theories have been described to account for brain development by Gerald Edelman (1992), and extended to the msec-to-minutes time scale of thought and action by William Calvin (1996a, 1996b). Evolutionary theory (and in some cases, explicit modelling) is present in economics, often tied to the development of technology, as in the work of Nelson and Winter (1992) or to the evolution of institutions and practices as in the work of Dosi et al (1999), Hodgson (1993) and North (1990).

Sociology too has used evolutionary ideas and simulations to understand the evolution of social order (Lomborg 1996, Macy 1996), changing populations of organisations (Hannan and Freeman 1993) and the survival of so-called "strict" churches (Chattoe 2006).

Interestingly, however, against these creative approaches must be set forces in particular social sciences that have slowed or marginalised their adoption. In sociology, the conversion of functionalism (potentially a form of social evolution) into a virtual religion was followed by a huge backlash against untestable grand theory which made these ideas virtually beyond the pale for 20 years or so (Chattoe 2002, Runciman 1998). It is quite likely that confused associations with Social Darwinism, eugenics and even Nazism have not helped the use of biological analogies in social science from the forties until quite recently. In economics, the focus on deliberate rationality and well defined equilibria has meant that evolutionary approaches are judged *ad hoc* unless they can be reinterpreted to suport the core assumptions of economics. (This can be observed, for example, in evolutionary approaches to equilibrium selection where the object is not to understand the dynamics of the system but to support the claim that particular equilibria are robust.) In psychology, while there seems to be no overt objection to evolutionary approaches, it seems to be the case (perhaps for historical reasons) that the main interest in these ideas is to explain behaviour using genetic accounts of cognitive structure rather than using evolutionary analogies.

In subsequent sections, having shown that interest in evolutionary ideas is widespread, we turn to technical details of various kinds of evolutionary algorithm, their strengths, weaknesses and social applicability so the reader is able to evaluate their use and consider applications in their own areas of research interest. We start with the Genetic Algorithm, which is easiest to describe, then move to Genetic Programming and the (more rarely used but in some sense more satisfactory as an analogy) Classifier Systems. The final example doesn't rely directly on the use of an evolutionary algorithm but clearly attempts to model a social process using a biological analogy.

The Basic Genetic Algorithm

This section describes the basic operation and limitations of the Genetic Algorithm. This leads to a description of ways in which the Genetic Algorithm can be generalised and a detailed discussion of one specific way of generalising it (Genetic Programming) in the subsequent section.

What is a Genetic Algorithm?

The Genetic Algorithm is actually a family of programmes developed by John Holland (1975) and his co-workers at the University of Michigan. The following algorithm describes the structure of a typical Genetic Algorithm. It is the different ways in which various parts of the algorithm can be implemented which produces the wide variety of Genetic Algorithms available. Each part of the algorithm will be discussed in more detail in a subsequent section. For the purposes of illustration, consider an attempt to solve the notorious Travelling Salesman Problem that involves producing the shortest tour of a set of cities at known distances visiting each once only (Grefenstette et al 1985).

1) Represent potential solutions to the problem as data structures.
2) Generate a number of these solutions/structures and store them as a composite data structure called the Solution Pool.
3) Evaluate the "fitness" of each solution in the Solution Pool using a Fitness Function.

4) Make copies of each solution in the Solution Pool, the number of copies depending positively on its fitness according to a Reproduction Function. These copies are stored in a second (temporary) composite data structure called the Breeding Pool.
5) Apply Genetic Operators to copies in the Breeding Pool chosen as "parents" and return one or more of the resulting "offspring" to the Solution Pool, randomly overwriting solutions which are already there. Repeat this step until some proportion of the Solution Pool has been replaced.
6) Repeat steps 3, 4 and 5 until the population of the Solution Pool satisfies a Stopping Condition. One such condition is that the Solution Pool should be within a certain distance of homogeneity.

There is an obvious parallel between this algorithm and the process of biological evolution that inspired it. The string representing a solution to a problem corresponds to the genotype and each element to a gene. The Fitness Function represents the environment that selects whole genotypes on the basis of their relative performance. The Genetic Operators correspond to the processes causing genetic variation in biology that allow better genes to propagate while poorer ones are selected out. This class of Genetic Algorithms has a number of interesting properties (for further discussion see Goldberg 1989).

1) It is evolutionary. Genetic Operators combine and modify solutions directly to generate new ones. Non-evolutionary search algorithms typically generate solutions "from scratch" even if the location of these solutions is determined by the current location of the search process. The common Genetic Operators are based on biological processes of variation. Genetic Operators permit short subsections of parent solutions to be propagated unchanged in their offspring. These subsections (called schemata) are selected through their effect on the overall fitness of solutions. Schemata that produce high fitness for the solutions in which they occur continue to be propagated while those producing lower fitness tend to die out. (Note that while it is not possible to assign a meaningful fitness to single genes, it is possible to talk about the relative fitness of whole genotypes differing by one or more genes. By extension, this permits talk about successful "combinations" of genes.) The Genetic Operators also mix "genetic material" (different solutions in the Breeding Pool) and thus help to ensure that all the promising areas of the Problem Space are explored continuously. These ideas clearly resonate with the social production of knowledge, in science for example.
2) It is non-local. Each solution is potentially exploring a different area of the Problem Space although solutions can "cluster" in promising areas to explore them more thoroughly. This allows for societies to be "smarter" than their members.
3) It is probabilistic. The Fitness Function ensures that fitter solutions participate in Genetic Operators more often because they have more copies in the Breeding Pool and are thus more likely to propagate their useful schemata. However, it sometimes happens that a solution of low overall fitness contains useful schemata. The probabilistic replacement of only a proportion of the Solution Pool with new solutions means that a small number of poor solutions will survive for sufficient generations that these schemata have a good chance of being incorporated into fitter solutions. This probabilistic approach to survival (when coupled with non-locality and the use of Genetic Operators) means that the Genetic Algorithm avoids getting stuck on non-optimal peaks in the Problem Space. Consider a

problem space with two peaks, one higher than the other. A simple hill climbing algorithm, if it happens to start "near" the lower peak, will climb up it and then be stuck at a non optimal position. By contrast, there is nothing to prevent the Genetic Operators from producing a new solution somewhere on the higher peak. Once this happens, there is a possibility of solutions fitter than those at the top of the lower peak and these will come to dominate the population. The search process can thus "jump" from one peak to another which most variants of hill climbing don't do.

4) It is implicitly parallel. In contrast with the behaviour of serial search algorithms that operate on a single best solution and improve it further, the Genetic Algorithm uses a population of solutions and simultaneously explores the area each occupies in the Problem Space. The results of these explorations are repeatedly used to modify the direction taken by each solution. The parallelism arises because the "side effects" of exploring the area surrounding each solution affect all the other solutions through the functioning of Genetic Operators. The whole is thus greater than the sum of its parts.
5) It is highly general. The Genetic Algorithm makes relatively few assumptions about the Problem Space in advance. Instead, it tries to extract the maximum amount of information from the process of traversing it. For example, non-evolutionary heuristic search algorithms use features like the gradient (first differential) which may not be calculable for a highly irregular Problem Spaces. By contrast, in the Genetic Algorithm all operations take place directly on a representation of the potential solution. The Fitness Function also evaluates fitness directly from solutions rather than using derived measures. Although no search technique escapes the fact that all such techniques exploit *some* properties of the problem space they are applied upon, in practice Genetic Algoritms are good at finding acceptable solutions to hard problems (which, in some cases, defeat other methods), albeit not always the best solution. Ironically, social evolutionary learning may be *better* at finding the solutions to difficult problems than rationality which struggles without high levels of knowledge about environmental structure.

The Problem Representation and Initial Population

The most important step in developing a Genetic Algorithm also requires the most human ingenuity. A good representation for solutions to the problem is vital to efficient convergence. Some solutions have more obvious representations than others do. In the Travelling Salesman Problem, for example, the obvious representation is an ordered list of numbers representing cities. For example, the solution (1 4 3 2) involves starting at city 1, then going to city 4 and so on. Once a representation has been developed, a number of solutions are generated and form the initial population in the Solution Pool. These solutions can be generated randomly or they may make use of some other ("quick and dirty?") algorithm producing better than random fitness. The optimum size of the initial population depends on the size of the Problem Space. A population of almost any size will ultimately converge. But the efficiency of the Genetic Algorithm relies on the availability of useful genetic material that can be propagated and developed by the Genetic Operators. The larger the initial population, the greater the likelihood that it will already contain schemata of an arbitrary quality. This must be set against the increased computational cost of manipulating the larger Solution Pool. The initial population must also be sufficiently large that it covers the Problem Space adequately. One natural criterion is that any given point in the Problem Space should not be more than a certain "distance" from some initial solution. A final requirement for a good solution representation is that some

"genes" should not be too much more important to overall fitness than others. Equivalent variations at different positions should have a broadly similar effect on overall fitness. In the Travelling Salesman Problem all the positions in the list are equivalent. They all represent cities. The efficiency of the Genetic Algorithms relies on the exponential propagation of successful schemata and this efficiency is impaired if schemata differ too much in importance as the system then becomes "bottlenecked" on certain genes.

The Fitness Function

The Fitness Function is at least as important as the solution representation for the efficiency of the Genetic Algorithm. It assigns fitness to each solution by reference to the problem that solution is designed to solve. The main requirement for the Fitness Function is that it must generate a fitness for any syntactically correct solution. (These are commonly referred to as "legal" solutions.) In the Travelling Salesman Problem, an obvious Fitness Function satisfying this requirement would be the reciprocal of the tour length. The reciprocal is used because the definition of the problem involves finding the shortest tour. Given this goal we should regard shorter tours as fitter. More complicated problems like constrained optimisation can also be handled using the Fitness Function. One approach is simply to reject all solutions that do not satisfy the constraints. This involves assigning them a fitness of 0. However, where solutions satisfying the constraints are sparse, a more efficient method is to add terms to the Fitness Function reflecting the extent of constraint satisfaction. These "penalty terms" lower the fitness of solutions that fail to satisfy the constraints but do not necessarily reduce it to zero.

The Process of Reproduction

Reproduction is sometimes classified as a Genetic Operator in that it takes a number of solutions (the Solution Pool) and produces a new set (the Breeding Pool). However, it is a Genetic Operator of a special type in that it uses additional information (the fitness of solutions and the Reproduction Function) in generating that population. The Reproduction Function links the fitness of individual solutions and the number of copies they produce. This process mimics the reproductive success of fitter organisms in biological systems. The number of copies depends on the type of Genetic Algorithm. Typically the fittest solutions in the Solution Pool may produce two or three copies while the worst may produce none. In order that potentially useful "genetic material" be retained, it is important that fitter solutions do not proliferate too rapidly, nor less fit solutions die out too fast. Despite their low fitness, poor solutions may contain useful schemata that need to be incorporated into better solutions. Ensuring "adequate" survival for instrumental efficiency is a matter of trial-and-error and depends on the problem and the type of Genetic Algorithm being used. There are two main types of reproduction strategies.

In the first, the Holland-Type algorithm (Holland 1975), the copies of each solution make up the Breeding Pool as described above. The Breeding Pool thus contains more copies of fitter solutions. There are two main sorts of Reproduction Function. The first is proportional fitness: here the number of copies produced for each solution is equal to the size of the Solution Pool normalised by some function according to the "share of fitness" accruing to each particular solution. Fitter solutions, responsible for a larger share of total fitness, produce more copies. This system is similar to that used in Replicator Dynamics (Vega-Rendondo 1996): it is performance relative to the average that determines the number of offspring. The second possibility is rank-based fitness. In this case, the number of copies depends on fitness rank. For example, the fittest 5 solutions may receive 2

copies each, the least fit receive no copies and all others receive 1. Both types of function have probabilistic equivalents. Instead of determining the actual number of copies, the function can determine the probability of drawing each type. The reproduction operator is then applied repeatedly, drawing from the probability distribution until the Breeding Pool is full. Clearly, this will still result in a greater proportion of fitter solutions in the Breeding Pool. The Reproduction Function can be linear or arbitrarily complex. In practice, the "shape" of the Reproduction Function is chosen on the basis of experience to optimise the performance of the Genetic Algorithm.

The second reproduction strategy, the GENITOR algorithm (Whitley 1989) does not involve a Breeding Pool. Instead of copying solutions into the Breeding Pool and then copying the results of Genetic Operators back again, the GENITOR takes parent solutions sequentially from the Solution Pool, applies Genetic Operators and returns the offspring immediately to the Solution Pool. The Solution Pool is kept sorted by rank and new solutions are appropriately placed according to fitness. A new solution either overwrites the solution with fitness nearest to its own or it is inserted into the Solution Pool so that all solutions with lower fitness move down one place and the solution with the lowest fitness is removed altogether. The GENITOR algorithm ensures that fitter solutions are more likely to become parents by using a skewed distribution to select them.

The differences between these strategies are instructive. The GENITOR algorithm is more similar to the interaction of biological organisms. The parents produce offspring that are introduced into a population that probably still contains at least one parent. Fitness affects which parents will mate, rather than generating offspring from all individuals in the Solution Pool. Even the "pecking order" interpretation of the introduction of offspring seems relatively intelligible. By contrast, the Breeding Pool in the Holland-Type algorithm seems to be an abstraction with little descriptive plausibility. The Holland-Type algorithm effectively splits the process of reproduction into two parts: the proliferation of fitter individuals and the subsequent generation of variation in their offspring. In biological systems, both processes result from the "same" act of reproduction. Furthermore, the differential production of offspring emerges from the relative fitness of parents. It is not explicitly designed into the system. In functional terms, both types of algorithm promote the survival of the fittest through variation and selective retention. In instrumental terms, one is sometimes more suitable than the other for a particular Problem Space. In descriptive terms, the GENITOR algorithm seems more appropriate to biological systems. (It can also be given a more plausible behavioural interpretation in social contexts.)

The Genetic Operators

There are two main types of Genetic Operator that correspond to the biological phenomena of recombination and mutation. These are the original Genetic Operators developed by Holland (1975). Recombination Genetic Operators involve more than one solution and the exchange of genetic material to produce offspring. The commonest example is the Crossover Operator. Two solutions are broken at the same randomly selected point (n) and the "head" of each solution is joined to the "tail" of the other to produce two new solutions. Here a_i identifies an ordered set of k genes from one parent and b_i identifies those from another.

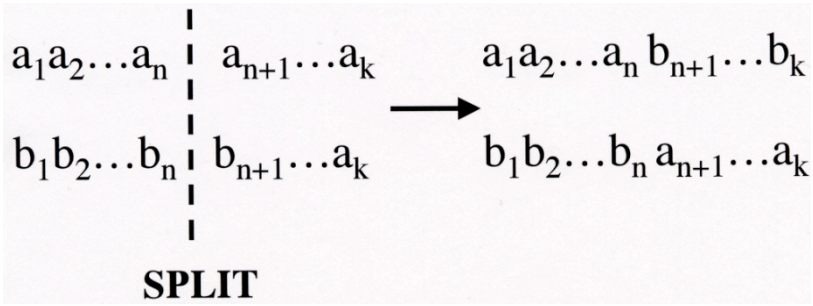

Figure 1. The Crossover Operator

One of the two new solutions is then chosen with equal probability as the offspring to be placed in the Solution Pool.

Mutation Genetic Operators involve a single solution and introduce new genetic possibilities. The two main sorts of mutation Genetic Operators used in Genetic Algorithms correspond to so called "large scale" and "point" mutation. In the Mutation Operator (corresponding to point mutation) one gene is altered to another value from the legal range (selected with equal probability) and shown in the diagram as h_n.

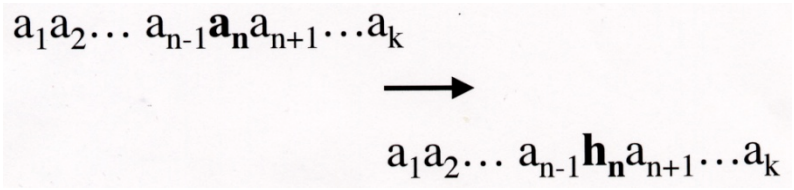

Figure 2. The Mutation Operator

One commonly used Genetic Operator corresponding to large scale chromosomal mutation is the Inversion Operator which involves reversing the order of a set of genes between two randomly selected points (n and m) in the genotype.

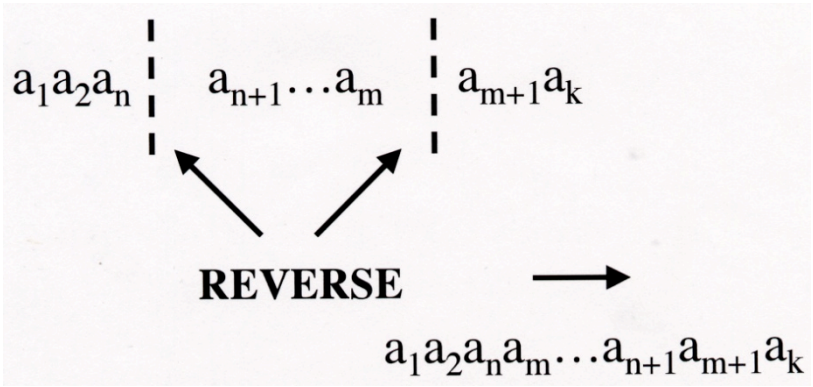

Figure 3. The Inversion Operator

The Inversion Operator provides an opportunity to discuss positional effects in solution representations although these can arise in all Genetic Operators except point mutation. It is not problematic to invert (reverse) the order in which a section of a city tour takes place in the Travelling Salesman Problem. However, there may be problem representations for which we have no reason to expect that the Inversion Operator will generate solutions that are even syntactically correct let

alone fit. There are two solutions to the production of illegal solutions by Genetic Operators. One is to use the penalty method. The other is simply to avoid unsuitable Genetic Operators by design. Positional effects pose particular problems if genes have different meanings, some representing one sort of object and some another. In this case, inverted solutions are almost certain not to be legal. This difficulty will be addressed further in the section on developing the Genetic Algorithm.

In biological systems, recombination ensures that the genes of sexually reproduced offspring are different from those of both parents. Various forms of mutation guarantee that entirely new genetic possibilities are also being introduced continuously into the gene pool. In the Genetic Algorithm, the Genetic Operators perform the same function, but the probability with which each is applied has to be tuned to ensure that useful genetic material can be properly incorporated before it is lost. Typically the Crossover Operator is applied with a high probability to each solution and the Mutation Operator with a low probability to each gene leading to a moderate probability of some mutation occurring in each solution. Other Genetic Operators are applied with intermediate probability. These probabilities are intended to reflect very approximately the relative importance of each process in biological systems.

For instrumental uses of the Genetic Algorithm, the setting of probabilities is a matter of experience. If the probabilities of application are too low, especially for the Mutation Operator, there is a danger of premature convergence on a local optimum followed by inefficient “mutation only” search. (In such cases, the advantages of parallel search are lost and the Genetic Algorithm effectively reverts to undirected serial search.) By contrast, if the probabilities of application are too high, excessive mixing destroys useful schemata before they can be combined into fit solutions.

There is a wide variety of other Genetic Operators discussed in the literature (Goldberg 1989, Mitchell 1996), some developed descriptively from biological systems and others designed instrumentally to work on particular problems. The descriptive use of Genetic Operators in the example provided here means that although it is important to bear the instrumental examples in mind, they should not be regarded as definitive. The processes of variation that affect the economic analogues of genotypes should be established empirically just as they were for biological genes.

Convergence

Because the Genetic Algorithm is a powerful technique, many of the problems it is used to solve are very hard to tackle by other means. Although it is possible to test the Genetic Algorithm by comparison with other techniques for simple problems, there is a danger that conclusions about performance will not scale to more complex cases. One consequence of this is the difficulty of defining satisfactory conditions for convergence. Provided the Problem Space is suitable, a non-evolutionary algorithm will find the best solution within a certain time. In the same time, the Genetic Algorithm is only statistically likely to converge (though, in practice, it will actually do so for a far larger class of problems). As a result, unlike some iterative procedures, the Genetic Algorithm cannot simply be stopped after a fixed number of generations. Instead, the properties of the Solution Pool must be analysed to determine when the programme should stop. The simplest method involves stopping when the fittest solution is “good enough”. Clearly, this involves a value judgement external to the definition of the problem. Another possibility is to stop the programme when the rate of change in best solution fitness drops below a specified level. Unfortunately the behaviour of the Genetic Algorithm means that improvements in fitness are often “stepped” as the Genetic

Operators give rise to whole ranges of new possibilities to be explored. For this reason more sophisticated approaches analyse the Solution Pool continuously and measure fitness in the whole population. Another advantage of this technique is that it allows for the fact that convergence is never total because of the Mutation Operator. There is always a certain amount of "mutation noise" in the Solution Pool even when it has converged.

Developing the Genetic Algorithm

The previous section was intended to provide a summary of the main aspects of design and a feel for the operation of a typical instrumental Genetic Algorithm (one that is supposed to solve a pre-defined problem as efficiently as possible). In the next two subsections, I describe a variety of generalisations that move the Genetic Algorithm away from the instrumental interpretation and towards the possibility of realistic *description* of certain social processes. This involves enriching the syntax for solution representations, developing formal techniques for analysing the behaviour of evolutionary models and making various aspects of the evolutionary process endogenous. The fact that these generalisations develop naturally from previous discussions suggests that a suitably sophisticated Genetic Algorithm might serve as a framework for evolutionary models of (carefully chosen) social phenomena. I shall try to show that Genetic Programming (as an extension of Genetic Algorithms) is particularly suitable for this purpose.

Generalising the Solution Representation

In the simplest Genetic Algorithm, the solution representation is just a list of numbers with a fixed length. Each gene (number) in the genotype (list) represents an object like a city in the Travelling Salesman Problem. But there is no reason why the Genetic Algorithm should be limited to solving problems using such a restricted representation. The enrichment of the syntax for solution representations has proceeded in three overlapping domains: the computational improvement of programmes implementing Genetic Algorithms, the incorporation of useful insights from biology and the study of theoretical requirements for the use of different solution representations.

Developments of the first sort are those which broaden the capabilities of the Genetic Algorithm itself. Instead of solutions of fixed length "hard coded" by the programmer, Goldberg et al (1990) have developed a "messy" Genetic Algorithm. This evolves an encoding of optimal length by varying the lengths of potential solutions as well as their encoding interpretations. Schraudolph and Belew (1992) have also addressed this problem, developing a technique called Dynamic Parameter Encoding that changes the solution encoding in response to an analysis of the current Solution Pool. (This technique avoids the loss of efficiency that results from premature convergence and the consequent failure of parallel search.) Finally, Harvey (1993) has stressed the importance of variable length genotypes in systems that are to display genuine increases in behavioural complexity.

Developments of the second sort have arisen from the study of biological systems. Smith et al (1992) have developed a Genetic Algorithm that produces a diverse coexistent population of solutions in "equilibrium" rather than one dominated by a single "optimal" solution. In this way, the coexistent population is capable of generalisation. This approach also forms the basis of the Classifier Systems discussed in Forrest (1991). Here groups of "IF [condition] THEN [action]" rules form coexistent data structures that can jointly perform computational tasks. Belew (1989, 1990) has developed this notion further by considering models in which the solutions themselves take in information from the environment and carry out a simple form of learning. Koza (1992b, 1992c) considers the possibility

of co-evolution. This is a process in which the fitness of a solution population is not defined relative to a fixed environment or Fitness Function but rather in terms of another population. He applies this technique to game strategy learning by Genetic Programmes. Clearly this development is important to models of social systems where we can seldom define, let alone agree, a clear objective ranking of alternative social arrangements. In a sense, it is the existence of a Fitness Function that identifies instrumental (rather than descriptive) applications of Evolutionary Algorithms. The exception might be a model in which different solutions to a problem were created "subconsciously" in the style of a Genetic Algorithm but were then evaluated "rationally" by an agent. For an example see Chattoe and Gilbert (1997).

Developments of the third sort involve the adaptation of formal systems such as grammars to serve as solution representations. Antoinisse has developed a representation and set of Genetic Operators that can be used for any problem in which legal solutions can be expressed as statements in a formal grammar (Antoinisse 1991). Koza (1992a, 1994) has developed a similar though far more general representation involving the syntax of computer languages. This approach (called Genetic Programming) will received detailed discussion in its own section shortly.

Making the Process of Evolution Endogenous

So far, most of the Genetic Algorithm generalisations discussed have been instrumental in their motivation and use. The abstractions and limitations in the simple Genetic Algorithm have not been viewed as unrealistic but merely unhelpful (since they are engineering solutions rather than attempts to describe and understand complex social behaviour). The interesting question from the perspective of this chapter is how it is possible to develop simulations based on Evolutionary Algorithms which are not just instrumentally effective (allowing firms to survive by learning about their market situation for example) but actually provides a convincing ("descriptive") insight into their decision processes and the complexity of the resulting system. At the same time, the powerful self-organising capabilities of Evolutionary Algorithms may serve to provide an alternative explanation of observed stability (and instability) in social systems which do not (or cannot) involve a high level of individual rationality. Despite the instrumental nature of most current developments in Genetic Algorithms, the trend of these developments suggests an important issue for the design of descriptive models.

Most of the developments discussed above can be characterised as making various aspects of the process of evolution endogenous. Instead of exogenous system level parameters that are externally "tuned" by the programmer for instrumental purposes, various parts of the evolutionary process become internalised attributes of the individual solutions. They need not be represented in the solution explicitly as numerical parameters. They are parameters in the more general sense that they alter the process of evolution and may be adjusted by the programmer. For example, the level of mutation may emerge from some other process (such as endogenous copying of information through imitation) rather than being "applied" to the solutions. Co-evolution provides a good example of this approach. In the instrumental Genetic Algorithm, the Fitness Function is specified by the programmer and applied equally to all solutions, producing an answer to some question of interest. To follow an old Darwinian example, this is equivalent to the deliberate breeding of particular dog breeds. In co-evolving Genetic Algorithms, as in biological evolution, there is no fixed Fitness Function. Fitness can only be measured relative to the behaviour of other agents that constitute an important part of the environment. This is equivalent to the production of the dog

species by biological evolution. Another example is provided by the Classifier Systems briefly discussed above. The simple Genetic Algorithm assumes that the fitness of an individual solution is independent of the fitness of other solutions. In practice, the fitness of one solution may depend on the existence and behaviour of other solutions. In biology, this is acknowledged in the treatment of altruism (Becker 1976, Boorman and Levitt 1980) and of group selection (Hughes 1988).

The use of solutions that are syntactically identical also abstracts from another important feature of evolution. Because the solutions only differ semantically there is no sense in measuring the relative "cost" of each. By contrast, when solutions differ syntactically, selection pressure may operate to produce shorter solutions as well as better ones. In descriptive models, "fitness" no longer measures an abstract quantity but describes the efficient scheduling of all scarce resources used including time. The less time is spent making decisions (provided they are sensible) the more time can be spent on other things. To put this point in its most general terms, organisms (and firms) are dynamic solutions to a dynamic environment while the simple Genetic Algorithm is a static solution to a static environment. Since social environments are dynamic, one way in which social agents can evolve or adapt is by evolving or adapting their models of that environment. Thus, an important way in which descriptive models can make the evolutionary process endogenous is by simulating agents that develop and test their own interpretations of the world in an evolutionary manner rather than being "gifted" with a fixed set of interpretations or decision processes by the modeller (Dosi et al 1999).

The value of making parts of the process specification endogenous can only be assessed in specific cases using descriptive plausibility as the main criterion. For example, if the rate of mutation can realistically be treated as fixed over the lifetime of a given evolutionary process it makes little practical difference whether it is represented as an extra global parameter or as part of the representation for each solution. In such cases, instrumental considerations such as computational efficiency may as well decide the matter. By contrast, making fitness endogenous will probably have a major effect on the behaviour of the system. In particular, there will be a tension between the descriptive plausibility of this change and the "instrumental" desirability of convergence to a unique optimum facilitated by an external Fitness Function.

This aspect of Genetic Algorithm design provides a new insight into the distinction between instrumental and descriptive models. Instrumental models are those allow the programmer to achieve her goals whatever they are. By contrast, the only goal that is permitted to shape a descriptive model is that of effective description as determined by empirical evidence. What determines the extent to which mutation should be modelled as a process inhering in agents is the extent to which the mutation process inheres in agents. Only once it has been shown that the mutation rate does not vary significantly across agents should it be represented as an environmental variable.

To sum up then, Genetic Algorithms constitute a broad class of powerful evolutionary search mechanisms with an active research agenda. Some (but not all) of the subsequent developments to the basic Genetic Algorithm are valuable to the descriptive modelling of social systems. (In addition some developments may have value in the characterisation of models. In the long term it may be possible to prove formal convergence results for descriptively realistic systems.) I now turn to a discussion of Genetic Programming, a significant variant of the Genetic Algorithm based on the idea

of “evolving” computer programmes which can both solve instrumental problems *and* represent sets of practices agents use to address the problems their environment creates for them.

Genetic Programming

The fundamental insight of Genetic Programming (Koza 1992a, 1994) is that Evolutionary Algorithms do not need to be limited to static representations or adaptation in a static environment. The approach originated in an instrumental concern, the possibility of evolving efficient computer programmes rather than having to design them explicitly (Koza 1991). However, it rapidly became clear that the power of the technique could be extended to any process which could be represented as an algorithm provided the fitness of different solutions could be measured (Koza 1992d). The possibility of developing descriptive models of agents was also considered early on (Koza 1992c). In most models of this kind, however, the fitness of the programme representing an agent is assessed by its ability to fulfil exogenous goals. Agents typically “compete” against the environment on an equal footing rather than constituting that environment.

The potential of such an approach is tremendous. It involves the possibility of an evolutionary process that operates on the richest representation language we can envisage: the set of computable functions. These functions can model the capability to collect, abstract, store and process data from the environment, transfer it between agents and use it to determine action. Furthermore, we know that (in principle at least) languages within the class of computable functions can also represent important features of human consciousness like self-awareness and self-modification of complex mental representations (Kampis 1991, Metcalfe 1994, Fagin et al 1995).

A simple example illustrates the most common solution representation used in Genetic Programming. This can be visualised as a tree structure and translates exactly into the set of “S-expressions” available in the LISP programming language (Friedman and Felleisen 1987). This is convenient for programming purposes because LISP comes already equipped to perform operations on S-expressions and can therefore easily and efficiently implement suitable Genetic Operators. The tree structure in Figure 4 is equivalent to the S-expression (OR (AND (NOT DO) (NOT D1))) (AND D0 D1)). This is the definition of the XOR (exclusive or) function. For obvious reasons, D0 and D1 are referred to as terminals and the set {AND, OR, NOT} are referred to as functions. The choice of a suitable set of functions and terminals (the equivalent of the solution representation in Genetic Algorithms) is a key part of Genetic Programming. Functions are by no means limited to the logical operators. They can also include mathematical operators and programming language instructions. Similarly, terminals can represent numerical (or physical) constants, a variety of “sensor” inputs from the environment (including the observable actions of other agents) and “symbolic” variables like “true” and “false”.

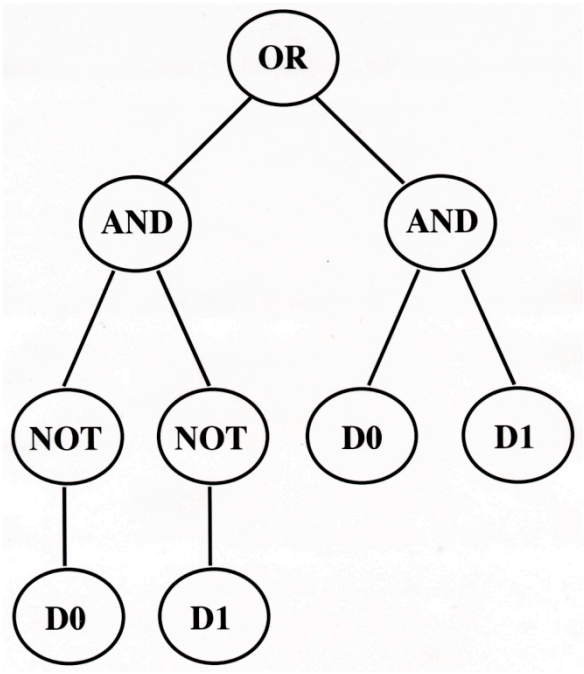

Figure 4. An S-Expression

The instrumental measurement of fitness involves providing the S-expressions with different "inputs", in this case truth values for D0 and D1 and assessing the extent to which the desired "output" results. For example, in Koza (1991) a programme to generate random numbers was tested by measuring the statistical properties of the number sequences it generated and rewarding such features as uncorrelated residuals. If S-expressions represent agents that are capable of action in an environment, success can be measured by the ability to modify the relationship between the agent and the environment in a certain way, for example by following a trail successfully. (The further along the trail an agent gets the fitter its programme.) It should be noted that the instrumental measurement of fitness requires a fairly precisely defined problem and solution grammar. On the other hand, the descriptive modelling of interaction need not. In order to do "well enough" in the market, a firm only needs to make some profit in every period sufficient to cover its costs. It may or may not have an internal goal to do better than this or even to make as much profit as it possibly can but this goal is not required for its survival (and may, in some cases, actually be counter-productive).

This discussion raises several potential difficulties with the descriptive use of Genetic Programming. However, these appear to recede on further consideration of the corresponding solutions to these problems in instrumental applications. The first difficulty is designing Genetic Operators that are guaranteed to produce meaningful offspring. In the S-expression representation, it is clear that a cut can be made at any point on the tree and the crossing of two such fragmented parents will always result in two legal offspring. However, the price to be paid for this advantage is that solutions must have a hierarchical form. More complicated function sets, mixing numerical and logical functions for example, must restrict crossover to prevent such outcomes as (+ 4 TRUE) or (NOT 6).

However, given the descriptive interpretation of Genetic Operators, it is plausible that agents should know the syntactic rules of combination for the set of terminals and operators they possess. As such the relevant "descriptive" Genetic Operators may execute rather more slowly than the simple instrumental ones but it is not unreasonable to suppose that only syntactically correct trees will

result. However, this raises another interesting possibility for using Genetic Operators. A good illustration is provided by a second difficulty with Genetic Programming, that of "bloating". This occurs because Genetic Programmes sometimes grow very large and contain substantial amounts of syntactically redundant material. (If a tree is trying to converge on a specific numerical value, for example, any sub trees evaluating to 0 are syntactically redundant.) Bloating produces a number of difficulties. Firstly, it slows down the evaluation of trees. Secondly, it becomes harder to interpret the trees and assess their behavioural plausibility. Finally, it is descriptively unsatisfactory. We do not expect real human decision processes to contain pointless operations (although bureaucratically specified production processes might, for example). Unfortunately, the obvious solution (the exogenous penalisation of long solutions) lacks precision. It is not possible to establish how long solutions to a particular problem "ought" to be without making arbitrary assumptions. The result is an ungrounded trade-off between length and quality. An interesting alternative is to introduce "purely syntactic" Genetic Operators. These take no account of tree fitness but simply look for redundant material within trees. For example, a Genetic Operator which replaced instances of the pattern (* constant 0) with 0 would be very simple to implement.

This approach allows firms (for example) to apply plausible syntactic knowledge to the structure of their decision processes ("rationalisation" in the non pejorative sense) without compromising the assumption (opposed to extreme economic rationality) that they cannot evaluate the fitness of a strategy without trying it in the market.

It also suggests a possible solution to another persistent problem with Genetic Programmes, that of interpretation. Even quite small trees are often hard to interpret and thus to evaluate behaviourally. Application of syntactic Genetic Operators may reduce the tree to a form in which it can be more easily interpreted. Another approach might be to use a Genetic Programming instrumentally to interpret trees, giving the greatest fitness to the shortest tree which can predict the output of a decision process tree to within a certain degree of accuracy. Thus, in principle at least, the Genetic Programming approach can be extended to include processes that are behaviourally similar to abstraction and refinement of the decision process itself.

As in the discussion of Genetic Algorithms above, I have kept this discussion of Genetic Programming relatively technical with some digressions about its general relevance to modelling social behaviour. In the final section of this chapter, I will present some evolutionary models in social science specifically based on Evolutionary Algorithms. This discussion allows us to move from general to specific issues about the applicability of biological analogies to social systems. In particular, I will try to show why models based on Genetic Programming and some Classifier Systems are more behaviourally plausible than those based on Genetic Algorithms.

Example using Genetic Algorithms: The Arifovic "cobweb" model

Arifovic (1994) is probably responsible for the best-known simulation of this type representing the quantity setting decisions of firms to show convergence in a cobweb model. She argues that the Genetic Algorithm both produces convergence over a wider range of model parameters than various forms of rational and adaptive learning, but also that it mimics the convergence behaviour of humans in experimental cobweb markets. Arifovic draws attention to two different interpretations of the Genetic Algorithm and explores the behaviour of both. In the "single population interpretation", each firm constitutes a single genotype and the Genetic Algorithm operates over the

whole market. In the “multiple population interpretation”, each firm has a number of genotypes representing alternate solutions to the quantity setting decision and operates its own “internal” Genetic Algorithm to choose between them. She shows that using a basic Holland-type Genetic Algorithm, neither interpretation leads to convergence on the Rational Expectations equilibrium for the cobweb market. When she adds her “Election” Genetic Operator however, both interpretations do so. The Election Operator involves using Crossover but then evaluating the offspring for profitability on the basis of the price prevailing in the previous period. The effect of this is to add some “direction” to the application of Genetic Operators, in fact a hill climbing component. An offspring is only added to the population if it would have performed better than its parents did in the previous period. This approach does not require any implausible knowledge as it is based on past events. However, it appears that the motivation for introducing the Election Operator is instrumental, namely to ensure perfect convergence to the Rational Expectations equilibrium (a goal of economic theory rather than a property of real markets necessarily). Interestingly, the graphs shown in the paper suggest that the Genetic Algorithm has done very well in converging to a stable (if mutation noise augmented) price fairly close to the Rational Expectations equilibrium. In fact, Arifovic shows how the Election Operator endogenously reduces the effective mutation rate to zero as the system approaches the theoretical equilibrium. She also points out that the Election Operator does not harm the ability of the Genetic Algorithm to learn a new equilibrium if the parameters of the cobweb model change. What she doesn’t explain is why the goal of the model should be to produce the theoretical equilibrium.

In fact, there are problems with both of her models that serve as instructive examples in the application of evolutionary ideas. The single population interpretation seems to involve a standard Holland-type Genetic Algorithm even down to a Breeding Pool that has no behavioural interpretation in real systems. There is also a problem with the use of Genetic Operators that is general in Genetic Algorithms. The way in which the bit strings are interpreted is very precise. If one firms uses Crossover involving the price strategy of another, it is necessary to “gift” a common representation to all firms and assume that firms know precisely where bit strings should “fit” in their own strategies. Given the encoding Arifovic uses, inserting a bit string one position to the left by mistake doubles the price it produces. In descriptive terms, this seems to be the worst of both worlds. It is easy to see how one firm could charge the same price as another, or (with more difficulty) acquire a “narrative” strategy fragment like “keep investment in a fixed ratio to profit” but not how firms could come to share a very precise arbitrary representation and copy instances around exactly. More generally, encoding price in this way is just behaviourally odd. It is hard to imagine what a firm would think it was doing if it took a “bit” of one of its prices and “inserted it” into another. Of course, the effect would be to raise or lower the price, but the way of going about it is very bizarre. I think the reason for this is that an encoding is not a procedure that is endogenously evolved. A Genetic Programme that calculates price by taking the previous price of another firm, adding unit cost and then adding 2 is telling a firm behaviourally how to determine price. These are “real” procedures given by the ontology of what a firm knows and knows how to do: the set of operators and terminals. By contrast there has to be reason why a firm would bother to encode its price as a bit string rather than just operating on them directly. Unless this encoding is “gifted”, it is not clear how (or why) the firm would develop it.

The multiple population interpretation is much more plausible in behavioural terms since the problem representation only needs to be the same within a firm, although the strangeness of

"combining" prices remains. A firm applying Genetic Operators to its own strategies can reasonably be assumed to know how they are encoded however.

However, both interpretations come up against a serious empirical problem noted by Olivetti (1994). Because the Election Operator is effectively a hill-climbing algorithm, it fails to converge under quite small changes to the assumptions of the model. In particular, Olivetti shows that the system doesn't converge when a white noise stochastic disturbance is added to the demand function. This suggests that Arifovic has not understood the main advantage of the Genetic Algorithm and her pursuit of instrumental convergence at the expense of behavioural plausibility is actually counter-productive. In a sense, this is just a reprise of the previous instrumental insight. Genetic Algorithms perform better on difficult problems precisely because they do not "hill climb" (as the Election Operator does) and can thus "jump" from one optimum to another through parallel search.

Example Using Classifier Systems: The Moss price setting model

As discussed briefly above, Classifier Systems consist of sets of "IF [condition] THEN [action]" rules that can collectively solve problems. They are evolutionary because new rules are typically generated using a Genetic Algorithm to select, recombine and mutate the most effective rules in the population. However, there is one significant (and potentially problematic) difference between Classifier Systems and Genetic Algorithms or Genetic Programming. This is the allocation of fitness to the individual rules, frequently using the so-called Bucket Brigade algorithm. This allows individual rules to "bid" fitness in order to take part in the set that is used to solve the problem in a particular instance. Rules taking part in a successful outcome then receive "recompense" also in terms of fitness. Unfortunately, the behavioural interpretation for this algorithm is not clear. In addition, the system is required to make decisions about how to "allocate" fitness between participating rules. This is the "Credit Assignment Problem" recognised in Artificial Intelligence and it is hard to produce effective general solutions. In particular, rules that are only used occasionally may nonetheless be essential under specific circumstances. (It is possible that an instrumental approach and lack of biological awareness have created this problem but that it is not actually intrinsic to this kind of modelling. In biological evolution there is no credit assignment. Phenotypic traits stand and fall together.) That said, the Classifier System has one definite advantage over both Genetic Programming and Genetic Algorithms assuming these difficulties can be overcome. This is that the individual rules may be much simpler (and hence more easily interpreted behaviourally) than Genetic Programmes. This ease of interpretation also makes it more plausible that individual rules (rather than sub trees from Genetic Programmes or very precisely encoded bit strings from Genetic Algorithms) might be transferred meaningfully between firms either by interpretation of observable actions or "gossip". Interestingly, despite their advantages, Classifier Systems are easily the least applied Evolutionary Algorithms for understanding social behaviour and this lacuna offers real opportunities for new research.

In what appears to be one of the earliest applications to firm decision making, Moss (1992) compares a Classifier System and a (non-evolutionary) algorithm of his own design on the task of price setting in a monopoly. His algorithm hypothesises specific relationships between variables in the market and then tests these. For example, if an inverse relationship between price and profit is postulated, the firm experiments by raising price and seeing whether profit actually falls. If not, the hypothesis is rejected and another generated. If it works, but only over a range, then the hypothesis is progressively refined. The conclusion that Moss draws from this approach illustrates an important

advantage of Genetic Programming over Genetic Algorithms and (some) Classifier Systems, that its solutions are explicitly based on process and therefore explanatory. Moss points out that the simple Classifier System simply evolves a price while his algorithm shows how the firm evolves a representation of the world that allows it to set a price. Although not doing it quite as explicitly as his algorithm, a Genetic Programme may incorporate a stylised representation of market relationships into the encoding of the decision process. (Of course, in certain circumstances the firm may lack the operators and terminals to deduce these relationships adequately or they may not form a reliable basis for action. In this case, simpler strategies like "price following" – simply setting the same price as another firm – are likely to result.)

To return to the point made by Moss, all the Classifier System models so far developed to study firm behaviour seem to be "flat" and "hard coded". By "flat" I mean that only a single rule is needed to bridge the gap between information received and action taken. In practice, the Classifier System paradigm is capable of representing sets of rules that may trigger each other in complex patterns to generate the final output. This set of rules may also encapsulate evolved knowledge of the environment although "hard coding" prevents this. For example, we might model the production process as a Classifier System in which the rules describe the microstructure of the factory floor: where each worker went to get raw materials, what sequence of actions they performed to transform them and where they put the results. In such a model events (the arrival of a partially assembled computer at your position on the production line) trigger actions (the insertion of a particular component). However, running out of "your" component would trigger a whole other set of actions like stopping the production line and calling the warehouse. The construction of such "thick" Classifier Systems is a task for future research. "Hard coding" implies that each rule bridges the gap between input and output in the same way, suggesting the common representation of Genetic Algorithms with its attendant behavioural implausibility. In the models described above decision-makers do not have the option to add to the set of conditions or to change the mappings between conditions and actions: changing price on the basis of customer loyalty rather than costs for example. There is nothing in the Classifier System architecture to prevent this, but all the current models seem to implement the architecture in a simplified and behaviourally implausible way that makes it more like a Genetic Algorithm than Genetic Programming in terms of "hard coding" of representations and decision processes.

Example using Genetic Programming: An artificial stock market

In this example there is a simulated market for a limited number of stocks, with a fixed number of simulated traders and a single "market maker". Each trader starts off with an amount of cash and can, in each trading period, seek to buy or sell each of the kinds of stock. Thus at any time a trader might have a mixture of cash and amounts of each stock. A single market maker sets the prices of each stock at the beginning of each trading period depending on the last price and the previous amount of buying and selling of it. The 'fundamental' is the dividend paid on each stock, which for each stock is modelled as a slowly moving random walk. There is a transaction cost for each buy or sell action by the traders. Thus there is some incentive to buy and hold stocks and not trade too much, but in general more money can be made (or lost) in short-term speculation. The market is endogenous except for the slowly changing dividend rate so that the prices depend on the buy and sell actions and a trader's success depends on "out-smarting" the other traders.

In the original Artificial Stock Market model (Arthur et al 1997) each artificial trader had a fixed set of price prediction strategies. At each time interval they would see which of these strategies was most successful at predicting the price in the recent past (fixed number of time cycles) and rely on the best of these to predict the immediate future price movements. Depending on its prediction using this best strategy it would either buy or sell.

In the model presented here each trader has a small population of action strategies for each stock, encoded as a GP tree. In each time period each artificial trader evaluates each of these strategies for each stock. The strategies are each evaluated against the recent past (a fixed number of time cycles) to calculate how much value (current value based on cash plus stock holdings at current market prices) the trader would have had if they had used this strategy (taking into account transactions costs and dividends gained), assuming that the prices were as in the recent past. The trader then picks the best strategy for each stock and (given constraints of cash and holdings) tries to apply this strategy in their next buy and sell (or hold) actions.

At the end of each trading period the set of action strategy trees are slightly evolved using the GP algorithm. That is to say that they are probabilistically "remembered" in the next trading round depending on their evaluated success, with a few of them crossed in a GP manner to produce new variations on the old strategies and a very few utterly new random strategies introduced. Thus as a result of this there is a lot of evolution of small populations occurring, namely a population for *each* trader and *each* stock. Here each GP tree represents a possible strategy that the trader could think of for that stock. The Genetic Programming algorithm represents the trader's learning process for each stock, thinking up new variations of remembered strategies, discarding strategies that are currently unsuccessful and occasionally thinking up completely novel strategies. This is thus a direct implementation of Campbell's model of creative thinking known as "Blind Variation and Selective Attention" (Campbell 1965). Further, it introduces notions of analogy and expertise into the model. A strategy that is good for one stock is *a priori* likely to be good for another similar stock. Thus, if a new stock is introduced, agents may use existing strategies to decide what to do about it. A new trader will have relatively poor strategies generally and will not necessarily have the feedback to choose the most appropriate strategy for a new stock. By contrast, an expert will have both a good set of strategies to choose from and better judgement of which to choose. These aspects of social (evolutionary) learning are clearly important in domains where there is genuine novelty which many traditional approaches do not handle well (or in some cases at all.)

The nodes of the strategy trees can be any mixture of appropriate nodes and types. (Edmonds 2002) uses a relatively rich set of nodes, allowing arithmetic, logic, conditionals, branching, averaging, statistical market indices, random numbers, comparisons, time lags and the past observed actions of other traders. With a certain amount of extra programming, the trees can be strongly typed (Haynes et al 1996), that is certain nodes can take inputs that are only a specific type (say numeric) and output a different type (say Boolean) – for example the comparison "greaterThan". This complicates the programming of the Genetic Operators but can result in richer and more specific trees.

Below are a couple of examples of strategies in this version of the stock market model. The output of the expression is ultimately a numeric value which indicates buy or sell (for positive or negative numbers, but only if that buy or sell is of a greater magnitude than a minimal threshold (which is a parameter, allowing for the "do nothing" – hold – option).

- [minus [priceNow 'stock-1'] [maxHistoricalPrice 'stock-1']] – *Sell if price is greater than the maximum historical price otherwise buy*;
- [lagNumeric [2] [divide [doneByLast 'trader-2' 'stock-3'] [indexNow]]] – *Do action of the action done by trader-2 for stock-3 divided by the price index 3 time periods ago*.

There are now a number of techniques in the field of Evolutionary Computation that can make such algorithms more efficient or give them different characteristics. Clearly efficiency is not the primary consideration here but rather how to make such algorithms correspond to the behaviour of observed social actors. In particular a large population of strategies would correspond to a very powerful ability in a human to find near-optimal strategies, which is clearly unrealistic. Thus a relatively small population of strategies is "better" since it does mean that particular traders get 'locked-in' to a narrow range of strategies for a period of time (maybe they all do so badly that a random, novel strategy does better eventually). This reflects the existence of "group think" and trading "styles" that can reasonably be anticipated in real markets. Other relevant issues might be that traders are unlikely to ever completely discard a strategy that has worked well in the past. (Many evolutionary models fail to take account of the fact that humans are much better at recall from structured memory than they are at reasoning. Such a model might thus "file" all past strategies but only have a very small subset of the currently most effective ones in live memory. However, if things started going very badly, it would be easy to choose not from randomly generated strategies but from "past successes". It is an interesting question whether this would be a more effective strategy.) Clearly however the only ultimate tests are whether the resulting learning behaviour sufficiently matches that of observed markets and whether the set of operators and terminals can be grounded in (or at least abstracted from) the strategies used by real traders. (Either test taken alone is insufficient. Simply matching behaviour may be a coincidence while "realistic" trading strategies that don't match behaviour have either been abstracted inappropriately or don't really capture what traders do. It is quite possible that what they are able to report doing is only part of what they actually do.)

Given such a market and trader structure what transpires is a sort of learning "arms-race" where each trader is trying to "out-learn" the others, detecting the patterns in their actions and exploiting them. The fact that all agents are following some strategy at all times ensures that (potentially) there are patterns in existence to be out-learned. Under a wide range of conditions and parameter settings one readily observes many of the qualitative patterns observed in real stock markets – speculative bubbles, crashes, clustered volatility, long-term inflation of prices and so on. Based on the simulation methodology proposed by Gilbert and Troitzsch (2005) and the idea of generative social science put forward by Epstein (2007), this outcome shows how a set of assumptions about individual actions (how traders implement and evolve their strategies) can potentially be falsified against aggregate properties of the system such as price trends across the range of stocks. Such models are an active area of research a recent PhD which surveys these is (xxxx).

Example: The functional survival of "strict" churches

There are clear advantages to using existing evolutionary algorithms to understand complex social processes as we hope we have shown through the examples above. Apart from an opportunity to discuss the "technicalities" of evolutionary algorithms through looking at simple cases, it is valuable to have programs that can be used "off the shelf" (rather than needing to be developed from

scratch) and for which there is an active research agenda of technical developments and formal analysis which can be drawn on. However, the major downside of the approach has also been hinted at (and will be discussed in more detail in the conclusion). Great care must be exercised in choosing a domain of application for evolutionary algorithms in understanding complex social systems. The more an evolutionary algorithm is used "as is", the smaller its potential domain of social application is likely to be. Furthermore, while it is possible, by careful choice of the exact algorithm, to relax some of the more socially unhelpful assumptions of evolutionary algorithms (the example of an external Fitness Function and a separate Breeding Pool have already been discussed), the danger is that some domains will simply require too much modification of the basic evolutionary algorithm to the point where the result becomes awkward or the algorithm incoherent. (A major problem with existing models has been the inability of their interpretations to stand up to scrutiny. In some cases, such as the Election Operator proposed by Arifovic, it appears that even the designers of these models are not fully aware of the implications of biological evolution.)

As suggested at the beginning of the chapter, the other approach, formerly rare but now increasingly popular is to start not with an evolutionary algorithm but with a social system and build a simulation that is nonetheless evolutionary based on the structure of that. The challenge of choosing domains with a clear analogy to biological evolution remains but is not further complicated by the need to unpick and redesign the assumptions of an evolutionary algorithm. Such an example of a "bespoke" evolutionary simulation is provided in this section.

(Iannaccone 1994) puts forward an interesting argument to explain the potentially counter-intuitive finding that "strict churches are strong". It might seem that a church that asked a lot of you, in terms of money, time and appropriate behaviour, would be less robust (in this consumerist era) than one that simply allowed you to attend on "high days and holidays" (choosing your own level of participation). However, the evidence suggests that it is the liberal churches that are losing members fastest. Iannaccone proposes that this can be explained by reflecting on the nature of religious experience. The satisfaction that people get out of an act of worship depends not just on their own level of involvement but also that of all other participants. This creates a free rider problem for "rational" worshippers. Each would like to derive the social benefit while minimising their individual contribution. Churches are thus constantly at the mercy of those who want to turn up at Christmas to a full and lively church but don't want to take part in the everyday work (like learning to sing the hymns together) that makes this possible. Iannaccone then argues that an interesting social process can potentially deal with this problem. If we suppose that churches do things like demanding time, money and appropriate behaviour from their worshippers, this affects the satisfaction that worshippers can derive from certain patterns of activity. If the church can somehow make non religious activities less possible and less comfortable, it shifts the time allocations of a "rational" worshipper towards the religious activities and can simultaneously reward him or her with the greater social benefit that comes from the church "guiding" its members in this way. To take a mildly contrived example. Muslims don't drink alcohol. They also dress distinctively. A Muslim who wanted to drink couldn't safely ask his friends to join him, could easily be seen entering or leaving a pub by other Muslims and would probably feel out of place and uncomfortable once inside (quite apart from any guilt the church had managed to instill). The net effect is that Muslims do not spend much time in pubs (while many others in the UK do) and have more time for religious activity. Of course, it is easy to pick holes in the specifics of Iannaccone's argument. Why would the Muslim not dress up in other clothes? (That itself might need explanation though.) Why not engage in another non

religious activity that was not forbidden? Why assume that only religious activities are club goods? (Isn't a good night at the pub just as much a result of collective effort?)

However, regardless of the details, the basic evolutionary point is this. Religious groups set up relatively fixed "creeds" that tell members when and how to worship, what to wear and eat, how much money must be given to the church and so on. Given these creeds worshippers join and leave churches. To survive, churches need worshippers and a certain amount of "labour" and income to maintain buildings, pay religious leaders and so on. Is it in fact the case, as Iannaccone argues that the dynamics of this system will result in the differential survival of strict churches at the expense of liberal ones? This is in, fact, a very general framework for looking at social change. Organisations like firms depend on the ability to sell their product and recruit workers in a way that generates profit. Organisations like hospitals are simultaneously required to meet external goals set by their funders and honour their commitments to their "customers": On one hand, the budget for surgery may be exhausted. On the other, you can't turn away someone who is nearly dead from a car crash knowing they will never survive to the next nearest accident and emergency department. This evolutionary interplay between organisations facing external constraints and their members is ubiquitous in social systems.

Before reporting the results and discussing their implications, two issues must be dealt with. Because this is a "two sided" process (involving worshippers *and* churches) we must attend to the assumptions made about the behaviour of these groups. In the model discussed here, it was assumed that churches were simply defined by a fixed set of practices and did not adapt themselves. This is clearly a simplification but not a foolish one. Although creeds do adapt, they often do so over very long periods and this is a risky process. If worshippers feel that a creed is just being changed for expedience (rather than in a way consistent with doctrine) they may lose faith just as fast as in a church whose creed is clearly irrelevant to changed circumstances. Speculatively, the great religious are those that have homed in on the unchanging challenges and solutions that people face in all times and all places while the ephemeral ones are those that are particular to a place or set of circumstances. Conversely, the model assumes that worshippers are strictly rational in choosing the allocations of time to different activities that maximise their satisfaction. Again, this assumption isn't as artificial as it may seem. Although we do not choose religions like we choose baked beans, there is a still a sense in which a religion must strike a chord in us (or come to do so). It is hard to imagine that a religion that someone hated and disbelieved in could be followed for long merely out of a sense of duty. Thus, here, satisfaction is being used in a strictly subjective sense without inquiring into any potential objective correlates. This life, for me, is better than that life. In terms of predicting individual behaviour, this renders satisfaction a truism but in the context of the model (and explaining the survival of different kinds of churches) what matters is not what people happen to like but the fact that they pursue it. To sum up, we could have represented the churches as more adaptive and the worshippers as less adaptive but since we are interested in the *interplay* of their behaviours (and, incidentally, this novel approach reveals a shortage of social science data about how creeds change and worshippers participate in detail), there is no definite advantage to doing so.

In a nutshell, the model works as follows (more details can be found in Chattoe 2006a). Each agent allocates their time to activities generating satisfaction (and different agents like different things to different extents). They can generate new time allocations in two main ways. One is by introspection, simply reflecting that a bit more of this and a bit less of that might be nicer. The other

is by meeting other agents and seeing if their time allocations would work better. This means, for example, that an agnostic who meets a worshipper from church A may suddenly realise that leading their life in faith A would actually be much more satisfying than anything they have come up with themselves. Conversely, someone "brought up in" church B (and thus mainly getting ideas from other B worshippers about "the good life") may suddenly realise that a life involving no churchgoing at all is much better for him or her (after meeting an agnostic). Of course, who you meet will depend on which church you are in and how big the churches (and agnostic populations) are. It may be hard to meet agnostics if you are in a big strict church and similarly, there are those whom a more unusual religion might suit very well who will simply not encounter its creed. Churches are created at a low rate and each one comes with a creed that specifies how much time and money members must contribute and how many non religious activities are "forbidden". Members can only have time allocations that are compatible with the creed of the church. These allocations determine the social benefits of membership discussed above. If a church cannot meet minimum membership and money constraints, it disappears. Thus, over time, churches come and go, differing in their "strictness" and their survival is decided by their ability to attract worshippers and contributions. Worshippers make decisions that are reasonable (but not strictly rational in that they are not able instantaneously to choose the best time allocation and church for them – which may include no church – for any state of the environment). This system reproduces some stylised facts about religion. New churches start small and are often (but not always) slow to grow. Churches can appear to fade and then experience resurgences. There are a lot of small churches and very few large ones.

What happens? In fact, there is almost no difference between the lifetimes of liberal churches and mildly strict ones. What is clear however is that very strict churches (and especially cults – which proscribe all non religious activities) do not last very long at all. It is important to be clear about this as people often confuse membership with longevity. It is true that strict churches can grow very fast and (for a while) very large but the issue at stake here is whether they will survive in the longterm. To the extent that the assumptions of the simulation are realistic, the answer would appear to be no. Thus we have seen how it is possible to implement a reasonably coherent biological analogy in a social context without using a pre-existing evolutionary algorithm.

Conclusion: Using biological analogies to understand social systems

Having presented a number of case studies of evolutionary algorithms in different application areas, we are now in a position to draw some general conclusions about the design and use of evolutionary simulations. Despite the fact that some have claimed that a generalised version of evolution (Blind Variation and Selective Retention) *is* the basic template for human creativity (Campbell xx) and that it is plausible that some processes similar to biological evolution do occur in human societies, it is unlikely that these processes will be direct translations of biological evolution in all its details. For this reason, we would propose that research into evolutionary models proceeds as follows (although it is inevitable that there will be some backward and forward interplay between the stages for reasons discussed below):

1) Start with your substantive research domain of interest (linguistics, stock markets, the rise and fall of religious groups) and consider the general arguments for representing these (or parts of them) in evolutionary terms. While it is seldom spelt out explicitly, there are actually rather few candidate "general social theories" to explain the dynamic interaction of choice

and change. Unless one believes that individuals have the power and knowledge required for rational action to benefit them (and note that this condition isn't met in situations as simple as the two person one shot Prisoner's Dilemma), evolution is really the only coherent and completely specified theory available.[2] Thus (and obviously the authors are biased in this regard) if you believe that agents act on imperfect knowledge in an independently operating[3] environment (such that there often a gap between what you expect to happen and what happens, however effectively you collect and process data about your environment), it is worth considering an evolutionary approach. We would argue that these conditions are met in most social settings but economists would diasgree.

2) Consider the explicit specification of an evolutionary process for your particular domain of research (perhaps using the four process specification above as a guide). The key choice made in this context is a "coherent" object of selection (OOS) whose presence or absence is empirically accessible. This makes organisations and firms with any kind of formal status particularly suitable. For informal groups like families, for example, it is much less clear what constitutes a "unit". (Is it, in a traditional society setting, that they physically survive, or, in a modern setting, that they still cohabit or are still on speaking terms? The problems here are evident.) Interestingly, individuals (while obviously "physically" coherent) are still problematic as objects of selection. Unless the model involves "bare" survival, it is less obvious what happens when an agent is "selected". However, examples still exist, such as who is trading in particular markets for example. Most of the rest of the evolutionary process specification follows naturally from the choice of an OOS. It then becomes fairly clear what the resource driving selection is (food for tribal groups, profit for firms, membership for voluntary organisations, attention for memes), what causes the birth and death of OOS (sexual reproduction, merger, religious inspiration, bankruptcy, lack of interest or memorability and so on) and what variation occurs between OOS. This last is an interesting area and one where it is very important to have a clearly specified domain of application. For example, consider industrial organisation. Textbook economic theory creates in the mind an image of the archetypal kettle factory (of variable size), selling kettles "at the factory gates" direct to customers and ploughing profits straight back into growth and better technology. In such a world, a firm that is successful early on can make lots of poor judgements later because it has efficient technology, market dominance, retained profit and so on. As such, evolutionary pressure rapidly ceases to operate. Further, this kind of firm does not "reproduce" (it merely gets larger) and even imitation of its strategy by other firms (that are smaller and poorer) may not cause the effective "spread" of social practices required by an evolutionary approach. (What works for the dominant firm may actually be harmful to smaller "followers".) By contrast, we can see the more modern forms of competition by franchises and chains (Chattoe 1999) or the more realistic detail of "supply chain production" as much more naturally modelled in evolutionary terms. In the first case, firms do directly "reproduce" a set of practices (and style of product, décor, amount of choice and so on) from branch to branch. More successful chains have more

[2] In fact, it might be argued that it is the *only* one. Rational choice cannot contend with novelty or the origin of social order. By focusing on *relative* performance, no matter how absolutely poor, evolution can produce order from randomness.

[3] This independence comes *both* from other social actors and physical processes like climate and erosion.

branches. Furthermore, the "scale" of competition is determined by the number of branches and it is thus reasonable to say that successful business practices proliferate. Wimpy may drive out "Joe Smith's Diner" from a particular town but Joe Smith is never a real competitor with the Wimpy organisation even if he deters them from setting up a branch in that town. This means that selection pressure continues to operate with chains at any scale competing with other chains at similar scales. Short of outright monoply, there is never a dominant market position that is stable.[4] In the second case, we can see how open ended evolution may create new opportunities for business and that supply chains as a whole constitute "ecologies" (Chattoe-Brown 2009). Initially, each firm may transport its own goods to market but once markets are sufficiently distant and numerous, there may be economies of scale in offering specialist transport and logistics services (for example, all goods going from Bristol to Cardiff in one week may be carried by a single carter or a firm may create a distribution infrastructure so not all goods are transported directly from origin to destination pairwise but via cost saving looped routes.) Again, it is clear how the organisations here must operate successful practices that satisfy both suppliers (those who want to deliver goods) and customers (those who want to receive them) and, further, how the nature of the business environment may change continuously as a consquence of innovation (whether technical or social). The creation of the refrigerated ship or the internal combustion engine may foreclose some business opportunities (like city raising of animals or harness making) and give rise to others which may or may not be taken up (spot markets, garages). These examples show several things. Firstly, it is necessary to be very clear what you are trying to understand as only then can the fitness of the evolutionary analogy be assessed. Secondly, it is useful to have a systematic way (Chattoe 1998, 2006b) of specifying evolutionary models since these stand or fall on their most implausible assumptions (particularly in social sciences which aren't very keen on this approach).[5] Thirdly, there are a lot more opportunities for evolutionary modelling than are visible to the "naked eye" particularly to those who take the trouble to develop both domain knowledge and a broad evolutionary perspective. Considering the ubiquity of branch competition and intermediate production in the real world, the economic literature is amazingly distorted towards the "autonomous kettle factory view" and simulation models of realistic market structures are scarcer still (though this is just starting to change). The price of adopting a novel method is scepticism by ones peers (and associated difficulties in "routine" academic advancement) but the rewards are large domains of unexplored research opportunities and the consequent possibility for real innovation. Finally, don't forget that it is always possible to use an evolutionary algorithm as a "black box learning system" within the "mind" of an agent or organisation, although there is a design issue about interpreting this kind of model discussed in the next section. Further, even as a "black box", the learning algorithm can make a crucial difference in simulations (Edmonds xx) and one cannot simply assume that any learning algorithm will do.

[4] This is probably because the market is spatially distributed and the only way of making additional profits is by opening more branches (with associated costs). There are no major economies of scale to be exploited as when the kettle factory simply gets bigger and bigger with all customers continuing to bear the transport costs.
[5] More informally, "The assumptions you don't realise you are making are the ones that will do you in".

3) Explore whether data for your chosen domain is available (or can readily be got using standard social science methods).[6] If it is available, does it exist at both the individual level and in aggregate? For example, is there observational data about firm price setting practices (in board meetings for example) and long term historical data about the birth, death and merger of firms in a particular industry and their prices over time? Because simulation is a relatively new method, it is still possible to build and publish exploratory (or less flatteringly "toy") models of evolutionary processes but it is likely to get harder and may become impossible unless the evolutionary model or the application domain is novel. It is almost certainly good scientific practice to make the accessibility of data part of the research design but it does not follow from this that only models based on available data are scientific. The requirement of falsifiability is served by the data being collectable "in principle" not already collected. The best argument to support claims of scientific status for a simulation is to consider (as a design principle) how each parameter could be calibrated using existing data or existing research methods. (The case is obviously weaker if someone has to come up with a new data collection method first although it helps if its approach or requirements can be sketched out *a priori*.) This aspect of research design also feeds into the decision about whether to use an existing evolutionary algorithm and if so, which one. The emerging methodology of social simulation (Gilbert and Troitzsch 2005, pp. 15-18, Epstein 2007) is to make a set of empirically grounded hypotheses at the micro level (firms set prices thus) and then falsify this ensemble at the macro level. (The real distribution of survival times for firms is thus: It does or does not match the simulated distribution of survival times produced by the model.) A problem will arise if it is hard to interpret the simulated price setting practices. Suppose, for example, we use GP to model the evolution of trading strategies in stock markets. We may use interviews or observation of real traders to decide what terminals and operators are appropriate but, having let the simulation run and observed plausible aggregate properties, we may still not know (and find it extremely hard to work out because of the interpretation issue) whether the evolved strategies used are actually anything like those which traders would (or could) use. Equating the empirical validation of the GP grammar with the validation of the strategies evolved from it is bit like assuming that, because we have derived a Swahili grammar from listening to native speakers that we are then qualified to decide when Swahili speakers are telling the truth (rather than simply talking intelligibly). It muddles syntax and semantics. The design principle here is then to consider how the chosen evolutionary algorithm will be interpreted to establish the validity of evolved practices. (Creative approaches may be possible here like getting real traders to design – or choose – GP trees to trade for them or getting them to "critique" what are effectively verbal "translations" of strategies derived from apparently successful GP trees as if they from real traders.) In this regard, it is their potential ease of interpretation that makes the relative neglect of CS models seem more surprising in evolutionary modelling.

4) Having first got a clear sense of what it is that needs to be modelled, it is then possible to choose a modelling technique in a principled way. As the analysis of case studies suggest, the danger with a "methods led" approach is that the social domain will be stylised (or simply falsified) to fit the method. A subsidiary difficulty with the methods led approach is

[6] In a way, it is black mark against simulation that this needs to be said. Nobody would dream of designing a piece of statistical or ethnographic work without reference to the availability or accessability of data!

that even if the researcher is wise enough to use a modified evolutionary algorithm to mirror a social process accurately (rather than distorting or abstracting the domain to fit the method), inadequate technical understanding may render the modified algorithm incoherent or ineffective. It is thus very important to understand fully any methods you plan to apply particularly with regard to any instrumental assumptions they contain. (In making convergence her goal for the GA cobweb model, Arifovic introduced an election operator which actually rendered the GA *less* effective in solving hard problems. This issue would probably have been foreseen in advance by a competent instrumental user of the GA technique. The muddle arose from the *interface* between social description and the GA as a highly effective instrumental optimisation device.) Having chosen a modelling technique, all its supporting assumptions must also be examined in the light of the application domain. For example, it is very important not to confuse single and multiple population interpretations of a GA: Do firms each have multiple candidate pricing strategies and chose them by an evolutionary process or is the evolutionary process of interest one in which single pricing strategies succeed and fail with the associated firms "carrying" them? Each model (or some combination) might be justified on empirical grounds but only if the difference in interpretation is kept clearly in mind. Although we are sceptical that systems of realistic social complexity would allow this, the principled choice of methods means that it is even possible that some domains would not require simulation at all but could be handled by mathematical models of evolution like replicator dynamics (Weibull 1995) or stochastic models (Moran 1962). By contrast, however, if the properties of the chosen social domain are too far from a standard evolutionary algorithm (such that it can neither be used wholesale or deconstructed without collapsing into incoherence), the best solution is to build a bespoke evolutionary model as was done for the "strict churches" case study. (At the end of the day, evolutionary algorithms were themselves "evolved" in a completely different engineering environment and we would not therefore expect them to apply widely in social systems. Thus, great care needs to be taken to use them only where they clearly do apply and thus have real value.) With free and widely used agent based modelling packages like NetLogo <http://ccl.northwestern.edu/netlogo/> and associated teaching materials (Gilbert and Troitsch 2005, Gilbert 2009), this is now much easier than it was. Ten years ago, one reason to use an existing algorithm was simply the significant cost of building your own from scratch. To sum up this strategy of research, the decision to use, modify or build an evolutionary algorithm from scratch should be a conscious and principled one based on a clear understanding of the domain and existing social science data about it.

The final piece of advice is not technical or methodological but presentational. In applying a novel method, be prepared to suffer equally at the hands of those who don't understand it and those who do! One of the hardest things to do in academia is to strike a balance between rejecting ill founded criticisms or those that translate to "I just don't like this" without also rejecting real objections that may devalue months (or even years) of your effort (and still, frustratingly for you, be part of "good science"). To judge criticisms in a novel area, you must be especially well informed and thus confident of your ground. For example, there is no clear cut evidence for Lamarckism (modification of the genotype by the phenotype during the life of the organism in a way that can the be transmitted by reproduction) in biology but in social systems such processes are ubiquitous. (Someone discovers a good way to discipline children. Those who were thus disciplined do the same

thing to their children. This is an acid test because, with hindsight, the "victims" have to see it as beneficial, and thus not have been warped by it, even if it was hateful at the time. Punishments so nasty that the victims won't inflict them or so ineffective that the parents stop bothering will die out.) Failure to understand this issue may either set you on the path of non-Lamarckian (and thus quite possibly implausible) evolutionary models of social systems or of apologising mistakenly for building Lamarckian models which don't "truly" reflect biological evolution (when that was never the design criterion for using biological analogies in social science anyway). The best way to address these issues is hopefully to follow the systematic procedure outlined above. This minimises the chances that you will miss things which critics can use to reject your models (and if they are hostile enough, your whole approach) and ensures that by justifying the models to yourself, you can actually justify them to others. In popular scientific folklore Darwin (still the greatest evolutionist) spent a considerable period trying to anticipate all possible objections to his theory and see how valid they were (and what counters he could provide) before he presented his work. Given how fraught the acceptance of his theory has been anyway, imagine if he had not troubled to take that step!

We hope we have shown, by the use of diverse case studies and different evolutionary modelling techniques both the considerable advantages and (potentially avoidable) limitations of this approach and encourage interested readers to take these ideas forward both in developing new kinds of models and applying evolutionary models to novel domains. The field is still wide open and we are always pleased to hear from potential students, co-workers, collaborators, supporters or funders!

Further Reading

GA

Koza

[xx. What's this for? Why would one read FR rather than references?]

References

Antoinisse H. 1991. A grammar based Genetic Algorithm. In: Rawlins G (ed). Foundations of Genetic Algorithms: proceedings of the first workshop on the foundations of Genetic Algorithms and Classifier Systems, Indiana University, 15-18 July 1990. Morgan Kaufmann.

Arifovic J. 1994. Genetic Algorithm learning and the cobweb model. Journal of Economic Dynamics and Control 18: 3-28.

Arthur WB Holland JH LeBaron B Palmer R and Tayler P. 1997. Asset pricing under endogenous expectations in an artificial stock market. In: Arthur WB Durlauf SN and Lane DA (eds). The economy as a complex evolving system II. Addison-Wesley.

Becker G. 1976. Altruism, egoism and genetic fitness: Economics and sociobiology. Journal of Economic Literature 14: 817-826.

Belew R. 1989. When both individuals and populations search: Adding simple learning to the Genetic Algorithm. In: Schaffer J (ed). The proceedings of the third international conference on Genetic Algorithms, George Mason University, 4-7 June 1989. Morgan Kaufmann.

Belew R. 1990. Evolution, learning and culture: Computational metaphors for adaptive search. Complex Systems 4: 11-49.

Blackmore S. 1999. The meme machine. Oxford University Press.

Boorman S and Levitt P. 1980. The genetics of altruism. Academic Press.

Boyd R and Richerson PJ. 1985. Culture and the evolutionary process. University of Chicago Press.

Buss DM. 1998. Evolutionary psychology: The new science of the mind. Allyn and Bacon.

Calvin W. 1996a. How brains think: evolving intelligence, then and now. Basic Books.

Calvin W. 1996b. The cerebral code: thinking a thought in the mosaics of the mind. The M. I. T. Press.

Campbell DT. 1965. Variation and selective retention in socio-cultural evolution. In: Barringer HR Blanksten GI and Mack RW (eds). Social change in developing areas: a reinterpretation of evolutionary theory. Schenkman.

Campbell DT. 1974. Evolutionary epistemology. In: Schlipp PA (ed). The library of living philosophers, Vol. XIV: The philosophy of Karl Popper. Open Court.

Campbell DT. xx. [Variation and selective retention *is* model for innovation. May be one of last two references.]

Cavalli-Sforza L and Feldman M. 1973. Cultural versus biological inheritance: phenotypic transmission from parents to children. Human Genetics 25: 618-637.

Chattoe E. 1998. Just how (un)realistic are evolutionary algorithms as representations of social processes? Journal of Artificial Societies and Social Simulation 1: <http://www.soc.surrey.ac.uk/JASSS/1/3/2.html>.

Chattoe E. 1999. A co-evolutionary simulation of multi-branch enterprises. Paper presented at the European Meeting on Applied Evolutionary Economics, Grenoble, 7-9 June.

Chattoe E. 2002. Developing the selectionist paradigm in sociology. Sociology 36: 817-833.

Chattoe E. 2006a. Using simulation to develop and test functionalist explanations: A case study of dynamic church membership. British Journal of Sociology 57: 379-397.

Chattoe E. 2006b. Using evolutionary analogies in social science: Two case studies. In: Wimmer A and Kössler R (eds). Understanding change: Models, methodologies and metaphors. Palgrave Macmillan.

Chattoe-Brown E. 2009. The implications of different analogies between biology and society for effective functionalist analysis. Draft paper, Department of Sociology, University of Leicester.

Chattoe E and Gilbert N. 1997. A simulation of adaptation mechanisms in budgetary decision making. In: Conte R Hegselmann R and Terna P (eds). Simulating social phenomena. Springer.

Cloak FT. 1975. Is a cultural ethology possible? Human Ecology 3: 161-182.

Costall A. 1991. The meme meme. Cultural Dynamics 4: 321-335.

Csányi V. 1989. Evolutionary systems and society: A general theory of life, mind and culture. Duke University Press.

Darwin CR. 1859. The origin of species by means of natural selection. John Murray.

Dautenhahn K and Nehaniv CL (eds). 2002. Imitation in animals and artifacts. The M. I. T. Press.

Dawkins R. 1976. The selfish gene. Oxford University Press.

Dawkins R. 1982. Organisms, groups and memes: Replicators or vehicles? In: The extended phenotype. Oxford University Press.

Dawkins R. 1993. Viruses of the mind. In: Dahlbohm B (ed). Dennett and his critics. Blackwell Publishers.

Dennett D. 1990. Memes and the exploitation of imagination. Journal of Aesthetics and Art Criticism 48: 127-135.

Dobzhansky T Ayala FJ Stebbins GL and Valentine JW. 1977. Evolution. W. H. Freeman.

Dosi G Marengo L Bassanini A and Valente M. 1999. Norms as emergent properties of adaptive learning: The case of economic routines. Journal of Evolutionary Economics 9: 5-26.

Edelman G. 1992. Bright air, brilliant fire: On the matter of the mind. Basic Books.

Edmonds B. 2002. Exploring the value of prediction in an artificial stock market. In: Butz VM Sigaud O and Gérard P (eds). Anticipatory behaviour in adaptive learning systems. Springer.

Edmonds B. xx [Effect of learning algorithms on outcomes.]

Epstein JM (ed). 2007. Generative social science: Studies in agent-based computational modelling. Princeton University Press.

Fagin R Halpern J Moses Y and Vardi M. 1995. Reasoning about knowledge. The M. I. T. Press.

Forrest S. 1991. Parallelism and programming in Classifier Systems. Pitman.

Friedman DP and Felleisen M. 1987. The little LISPER. The M. I. T. Press.

Gilbert N. 2009. Agent-based models. Sage.

Gilbert N and Troitzsch KG. 2005. Simulation for the social scientist, second edition. Open University Press.

Goldberg DE. 1989. Genetic Algorithms in search, optimization and machine learning. Addison-Wesley.

Goldberg DE Deb K and Korb B. 1990. Messy Genetic Algorithms revisited: Studies in mixed size and scale. Complex Systems 4: 415-444.

Grefenstette J Gopal R Rosmaita B and Van Gucht D. 1985. Genetic Algorithms for The Travelling Salesman Problem. In: Grefenstette J. (ed). Proceedings of an International Conference on Genetic Algorithms and Their Applications, Carnegie Mellon University, Pittsburgh, PA, 24-26 July 1985. Lawrence Erlbaum.

Hannan MT and Freeman J. 1993. Organizational ecology. Harvard University Press.

Harvey I. 1993. Evolutionary robotics and SAGA: The case for hill crawling and tournament selection. In: Langton CG (ed). Artificial Life III: Proceedings of the workshop on Artificial Life, Santa Fe, New Mexico, June 1992. Addison-Wesley.

Haynes T Schoenefeld D and Wainwright R. 1996. Type inheritance in strongly typed Genetic Programming. In: Angeline PJ and Kinnear JE Junior (eds). Advances in Genetic Programming 2. The M. I. T. Press.

Heyes CM and Plotkin HC. 1989. Replicators and interactors in cultural evolution. In: Ruse M (ed). What the philosophy of biology is: essays dedicated to David Hull. Kluwer Academic Publishers.

Hodgson G. 1993. Economics and evolution: Bringing life back into economics. Polity.

Hoenigswald HM and Wiener LS. 1987. Biological metaphor and cladistics classification. Francis Pinter.

Holland JH. 1975. Adaptation in natural and artificial systems. University of Michigan Press.

Hughes A. 1988. Evolution and human kinship. Oxford University Press.

Hull DL. 1982. The naked meme. In: Plotkin HC (ed). Learning development and culture: essays in evolutionary epistemology. John Wiley and Sons.

Hull DL. 1988. Interactors versus vehicles. In: Plotkin HC (ed). The role of behaviour in evolution. The M. I. T. Press.

Iannaccone L. 1994. Why strict churches are strong. American Journal of Sociology 99: 1180-1211.

Kampis G. 1991. Self-modifying systems in biology: A new framework for dynamics, information and complexity. Pergamon Press.

Kauffman SA. 1993. The origins of order, self-organization and selection in evolution. Oxford University Press.

Koza JR. 1991. Evolving a computer program to generate random numbers using the Genetic Programming paradigm. In: Belew R and Booker L (eds). Proceedings of the fourth international conference on Genetic Algorithms, UCSD, San Diego 13-16 July 1991. Morgan Kaufmann.

Koza JR. 1992a. Genetic Programming: On the Programming of Computers by Means of Natural Selection. A Bradford Book/The M. I. T. Press.

Koza JR. 1992b. Genetic evolution and co-evolution of computer programmes. In: Langton C Taylor C Farmer J and Rassmussen S (eds). Artificial Life II: Proceedings of the workshop on Artificial Life, Santa Fe, New Mexico, February 1990. Addison-Wesley.

Koza JR. 1992c. Evolution and co-evolution of computer programs to control independently acting agents. In: Meyer J-A and Wilson S (eds). From animals to animats: Proceedings of the first international conference on simulation of adaptive behaviour (SAB 91), Paris, 24-28 September 1991. A Bradford Book/The M. I. T. Press.

Koza JR. 1992d. A genetic approach to econometric modelling. In: Bourgine P and Walliser B (eds). Economics and cognitive science: Selected papers from the second international conference on economics and Artificial Intelligence, Paris, 4-6 July 1990. Pergamon.

Koza JR. 1994. Genetic Programming II: Automatic discovery of reusable programs. A Bradford Book/The M. I. T. Press.

Kuhn TS. 1970. The structure of scientific revolutions. University of Chicago Press.

Kummer H Daston L Gigerenzer G and Silk J. 1997. The social intelligence hypothesis. In: Weingart P Richerson P Mitchell SD and Maasen S (eds). Human by nature: Between biology and the social sciences. Lawrence Erlbaum.

Lomborg B. 1996. Nucleus and shield: The evolution of social structure in the Iterated Prisoner's Dilemma. American Sociological Review 61: 278-307.

Lynch A. 1996. Thought contagion, How belief spreads through society: The new science of memes. Basic Books.

Macy M. 1996. Natural selection and social learning in the Prisoner's Dilemma: Co-adaptation with Genetic Algorithms and Artificial Neural Networks. Sociological Methods and Research 25: 103-137.

Metcalfe J (ed). 1994. Metacognition: Knowing about knowing. A Bradford Book/The M. I. T. Press.

Mitchell M. 1996. An introduction to Genetic Algorithms. A Bradford Book/The M. I. T. Press.

Moran PAP. 1962. The statistical processes of evolutionary theory. Clarendon Press.

Moss S. 1992. Artificial Intelligence models of complex economic systems. In: Moss S and Rae J (eds). Artificial Intelligence and economic analysis: prospects and problems. Edward Elgar.

Nelson RR and Winter SG Junior 1982. An evolutionary theory of economic change. Belknap Press of Harvard University Press.

Nelson RR. 1987. Understanding technical change as an evolutionary process. North-Holland.

North DC. 1990. Institutions, institutional change and economic performance. Cambridge University Press.

Oliphant M. 1996. The development of Saussurean communication. BioSystems 37: 31-38.

Olivetti C. 1994. Do Genetic Algorithms Converge to Economic Equilibria? Discussion Paper Number 24, Department of Economics, University of Rome “La Sapienza”.

Popper KR. 1979. Objective knowledge: An evolutionary approach. Clarendon Press.

Reader 1970. xx.

Runciman WG. 1998. The selectionist paradigm and its implications for sociology. Sociology 32: 163-188.

Schraudolph N and Belew R. 1992. Dynamic parameter encoding for Genetic Algorithms. Machine Learning 9: 9-21.

Smith R Forrest S and Perelson A. 1992. Searching for diverse co-operative populations with Genetic Algorithms. TCGA Report Number 92002, The Clearinghouse for Genetic Algorithms, Department of Engineering Mechanics, University of Alabama (Tuscaloosa).

Tarde G. 1884. Darwinisme naturel et Darwinisme social. Revue Philosophique XVII: 607-637.

Tarde G. 1903. The laws of imitation. Henry Holt

Vega-Redondo F. 1996. Evolution, games and economic behaviour. Oxford University Press.

Weibull J. 1995. Evolutionary game theory. The M. I. T. Press.

Westoby A. 1994. The ecology of intentions, How to make memes and influence people: Culturology. <http://ase.tufts.edu/cogstud/papers/ecointen.htm>.

Whitley D. 1989. The GENITOR algorithm and selection pressure: Why rank-based allocation of reproductive trials is best. In: Schaffer J (ed). The proceedings of the third international conference on Genetic Algorithms, George Mason University, 4-7 June 1989. Morgan Kaufmann.

Wilson EO. 1975. Sociobiology: The new synthesis. Harvard University Press.

Windrum P and Birchenhall C. 1998. Developing simulation models with policy relevance: Getting to grips with UK science policy. In: Ahrweiler, P and Gilbert N (eds). Computer simulations in science and technology studies. Springer-Verlag.

xx. PhD reference.

xx. SIMIAN credit.