Neuro-Psychological Social Theorizing and Simulation with the Computational Multi-Agent System ETHOS

Jorge Simão[1*] and Luís Moniz Pereira[2]
Centro de Inteligência Artificial – CENTRIA
Faculdade de Ciências e Tecnologia – Universidade Nova de Lisboa
2829 - 516 Caparica, Portugal
E-mail:[1] **jsimao@di.fct.unl.pt**; [2] **lmp@di.fct.unl.pt**

Keywords: Neuro-psychology, human social behaviour, mate choice, cultural change, multi-agent based computational modelling.

Abstract

Human social behaviour, culture change, and emergent social organization are amongst the most intricate phenomena studied by science. Aided by theoretical and computational tools developed to study emergent phenomena in complex systems, social theorists aim to develop a unified body of knowledge that helps to shed light on long lasting question of human sociality. With this intent in mind, we have been developing a new conceptual framework, in the form of a Multi-Agent System (MAS) based on a simple abstraction of individuals' cognitive hard-wired neuro-psychological behaviour, and implemented as an object-oriented computational paradigm. This framework, named ETHOS, extends the traditional features provided in current MAS for agent-based modelling, with new abstractions specifically designed to model psychologically determined human social behaviour, culture, and organization. These include support for flexible behaviour selection mechanisms, including individual experience based learning, the transmission of information and social facilitation of learning, the management of agents' social networks, and the definition of the task-environments that structure an agent's action and interaction. This is accomplished by providing an object-oriented framework with a relatively small set of main interfaces and top classes abstracting key theoretical constructs, and having the modeler selectively sub-class filling the basic object structure to implement her/his own model. Thus the ETHOS class structure provides a meta-model, which can be instantiated in a flexible manner to implement each concrete model of individual human social behaviour. In addition to presenting the general framework, we report on our own experiences in using ETHOS to (re)implement several models we have developed. This includes models of human mate choice strategies and emergent human mating systems, and of the cultural dynamics of preferences such as identification of fashion-like product careers. We argue that providing the type of functionality afforded by ETHOS, off-the-self, substantially improves theoretical integration and facilitates model comparison. It can also be of great help to students of social and cognitive science who wish to develop theoretical work based on tested, established, and accepted computational building blocks.

*This work was partially supported by a PRAXIS XXI Ph.D. scholarship, and project FLUX, funded by FCT/MCES, Portugal. We would like to acknowledge the scientific and intellectual support and/or comments to earlier draft versions of this paper to Peter Todd, Paulo Gama Mota, John McNamara, John Hutchinson, Rainer Hilscher, Lus Correira, Monique Borgerhoff-Mulder, Nuno Preguiça, and João Sousa.

1 Introduction

Human social behaviour and culture change are amongst the most complex phenomena studied by science. This results from the large number of interacting entities within a society, the different neuropsychological mechanisms underlying human social interaction, and the multiple channels of information inheritance, amongst other factors. All of these contribute to create non-trivial dynamics in inter-personal relationships, social structure, and collective beliefs and values, which require careful analysis. This has challenged many researchers to propose unified theoretical framework, towards a social science based on a naturalistic interpretation of human behaviour and culture (Durham, 1990; Boyd & Richerson, 1985; Laland, Odling-Smee, & Feldman, 2000; Barkow, Cosmides, & Tooby, 1992; Sperber, 1996; Deacon, 1998; Donald, 1993; Mithen, 1996).

In spite of these efforts, there is an understanding that one still lacks a common theoretical language in which to naturally express a large range of models of social phenomena (Gilbert, 2000; Doran, 2000). This is a consequence in part of the recency of the introduction of computers in mainstream social science and, most importantly, of the complexity of the phenomenon studied. The lack of shared language frustrates possible advances in scientific research insofar has it makes harder to compare conflicting models and model predictions, and puts barriers to the communication between overlapping research communities. Computational tools are a particularly valuable avenue in this regard. Because computational tools require all abstractions to be formally specified (at least up to the programming language level), and they need to make explicit what its ontological commitments are. In so doing, they establish the set of basic building blocks upon which models can be built, thus providing a common set of abstractions that modelers can employ and the community can share.

Notwithstanding this possibility, computational tools utilized to support the modelling and simulation of systems with many interacting elements (complex systems) often lack features that permit capturing human specificities in a simple manner. For example, frameworks like Swarm, Repast, and Ascape (Minar, Burkhart, Langton, & Askenazi, 1996; Collier, 2002; Brookings, 2000), while largely convergent on the set of tools provided, do not provide any specific flexible mechanisms to simplify the modelling of human social learning (e.g. observational learning (Bandura, 1985; Boyd & Richerson, 1985)), social relationships' dynamics (Nowak & Vallacher, 1998), dissemination of values (Durham, 1990), emotional contagion (Doran, 2000), or non trivial behavioural control (Bryson, 2000). This has the effect of making many interesting models hard to program, thus losing out on some of the advantages of using computational models as a complement to analytic mathematical ones. Namely, fast prototyping and analysis, and relaxation of assumptions made mostly for mathematical tractability (e.g. focusing on equilibrium states, and the use of infinite populations sizes).

To address these unique characteristics, we propose in this paper a new conceptual framework for a Multi-Agent System (MAS), named Ethos. Our goal is to go a step further than other MAS in providing abstractions that facilitate the ease with which modelers can express naturally a wide range of model designs, namely, cognitive hardwired agent-based models of human social behaviour and culture[1]. In section 2, we start by highlighting the requisites that we per force considered when designing and implementing the framework. Of particular pertinence, is that we have tried to provide abstractions of high expressiveness, but which at the same time were implemented having performance has a key mandatory requisite. In section 3, we present the MAS overall structure and function, and describe selected aspects of its design and implementation. To test the usefulness of

[1]The material presented in this paper is further elaborated elsewhere (Simão, 2003).

ETHOS abstractions, we present two original models of human social behaviour. In section 5, we set forth a model intended to study human mating in monogamous societies. We report results which are backed up by empirical evidence for levels of assortment in mating, relationship stability, and distribution of age at mating. In section 6, we present a model designed as a metaphor for human cultural change. We take a look at the dynamics of the model, with particular focus on fashion-like collective behaviour — that is, continuous change in the trends of trait usage and avoidance. Both of these models were implemented (or re-implemented) with ETHOS. The simplicity and elegance of their implementation in ETHOS is taken as an ominous sign, and as a first step towards validating our MAS framework. In section 7, we discuss our experience in using the services provided by ETHOS, point out directions for future work, and supply concluding remarks.

2 Framework Design and Implementation Requirements

In order to guide our research in developing a MAS framework for the simulation of cognitive driven human social behaviour and culture, we have identified several generic design and implementation requirements that a simulation framework should meet. Below, we enumerate some of these:

- Expressiveness and Flexibility

  Generic simulation frameworks should provide an infrastructure that allows a wide range of models to be easily implemented. Framework abstractions should map naturally into the model application domain's key abstractions, and should anticipate the relationships between them. However, this is not to be achieved rigidly to the extent of excessively constraining the model design space. While some multi-agent systems might not be more than a highly configurable specific model (e.g. Epstein and Axtell's Sugarscape model (Epstein & Axtell, 1996)), a generic simulation framework should support the design of a broad range of model designs. The challenge is to strike a balance between providing only a set of loosely interconnected features, so that it might be hard for the model designer to understand the envisioned ways of their use, and a framework that is so specific that it can capture naturally only a small scope of application domain. It should be noticed that this issue permeates the general theme of framework design and the (arguably fuzzy) distinction between a framework and a software library (Gamma, Helm, Johnson, & Vlissides, 1995).

- Extensibility and Modifiability

  Models are rarely studied in isolation. It is frequent that a family of related models or model variants are explored over time by a research group as part of a larger project to understand a particular type of system. Thus, it is desirable that models be easily extensible and modifiable to explore alternative assumptions and designs, and check if and in what scientifically relevant ways that changes model behaviour. This should be possible with minimal and localized changes to model code. That is, it should be possible to make significant changes in one aspect of model without affecting others. It also requires that data gathering and visualization tools be largely tailored to promote easy exploration and experimentation.

- Transparency

  Models of complex systems are often very sensitive to small changes in model assumptions. This means that whatever framework services are used they should be as clearly specified and easy to understandable as possible. Moreover, this increases the researcher's confidence that the observed behaviour and results are a genuine effect of its model assumptions, rather

than an artifact of some layer of software that she/he did not write. The latter implies that simulation frameworks should be well documented, and it should be possible to bypass particular services if need be.

- Performance, Scalability, and Robustness

  Typically, simulations run for many time steps, many times over, with each run potentially involving a large number of agents. This implies that performance and scalability are central issues. A design that may work well with a few dozen agents, might become inadequate when their number increases to a magnitude of thousands. It also requires framework implementations to be particularly optimized for time critical operations, such as the ones involved in the more inner simulation loops (e.g. the invocation of agent's behaviour by an event scheduling mechanism).

- Portability and Ease of Use

  As the prevalence of computer modelling techniques increases in complex system research, the trend is to have an ever larger number of researchers involved in the study of specific problem domains. This is an indicator that the independent implementation of model specification, replication of results, and sharing of code between research groups may become a widespread practice. Thus, it is convenience for frameworks and model specifications to be inter-operable and portable across a wide range of hardware technologies and software platforms. Portable run-time systems like JAVA are specially attractive in this respect, because they have been conceived from the root to work in virtually all currently available commercial platforms.

3 Ethos Framework Overview

The ETHOS MAS framework offers as basic building blocks the kind of entities the informed modeler is likely to consider when thinking intuitively about human social behaviour and culture. This includes objects describing the structure and topology of physical spaces, physical entities placed in this space (such as resources and agents with varying attributes and genetic makeups), distinct kinds of social relationships amongst agents, behaviour selection mechanisms, mechanisms for the social influencing of agents' cognitive states, defined contexts of individual action and social interaction, and others besides. In the sequel, we shall describe all such abstractions in greater detail. In figure 1, we depict the key abstractions of ETHOS and how they relate to on another. The text labels in figure 1, and the highlighted words in the presentation below correspond to object classes in the framework. The relationships between the main classes of ETHOS's meta-model in terms of inheritance, aggregation, and acquaintance are shown in fig. 2. Due to the wide variety and expressiveness of Ethos code abstractions, specific models may of course use only a subset of all those made available.

3.1 Modelling the Physical and Social Environment

The top level abstraction of ETHOS is a World object containing a set of one or more physical spaces (Space object), whose events are timed by a common clock. Each such Space object consists of a topological arrangement of Site objects, each of which is a place-holder for a set of physical bodies (Body objects). A Space object defines its own geometry for the arrangement of Sites, and provides generic services to navigate between Sites. For example, finding the list of neighboring

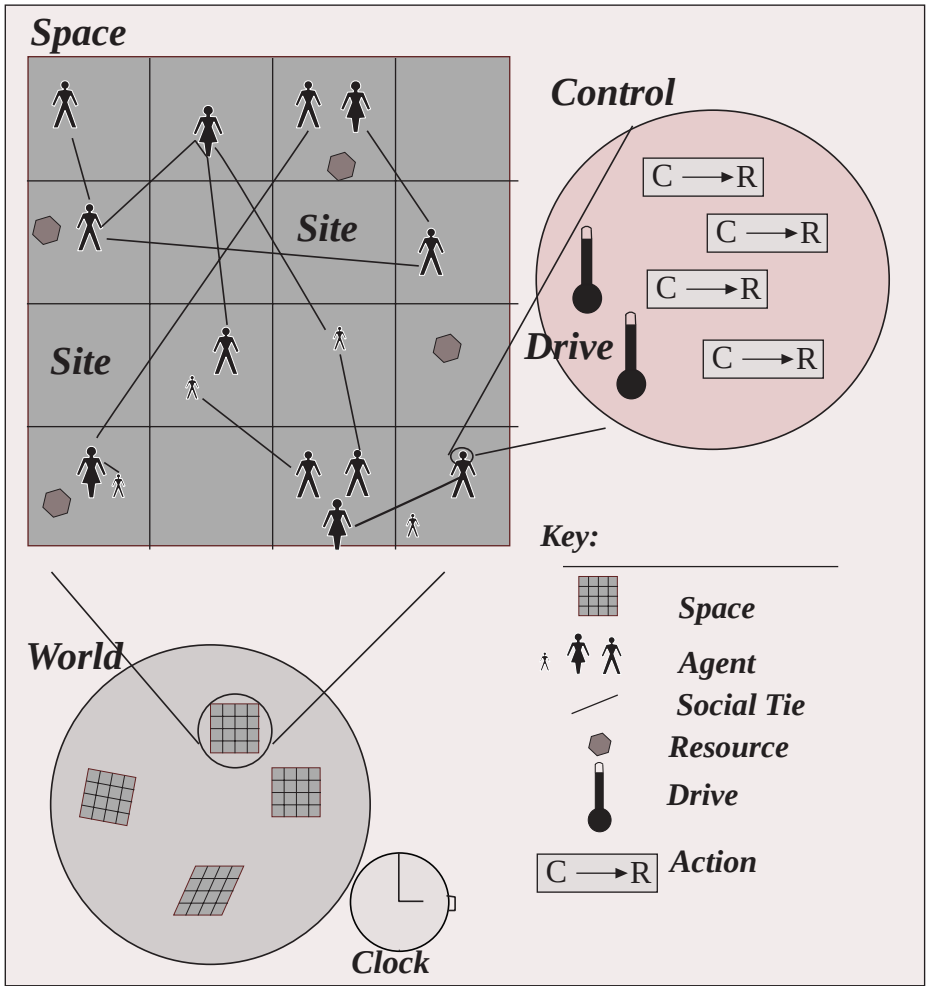

Figure 1: Ethos's Framework Main Abstractions

sites according to some neighbourhood type (e.g. Moore or von Neumann), at a specified metric distance (e.g. Euclidean or Block), and contingent upon the underlying Space topology (e.g. 2D torus or frame). Subclasses of Space currently implemented are: GridSpace that implements a 2D grid world, ListSpace for a one-dimensional Space object (possibly) with a dynamic number of sites, and VoidSpace which has a single site where Body objects are positioned (see below).

The Body objects living in sites may be agents (Agent object), or other physical entities such as growing/consumable resources (Resource objects). Agents are usually made to move from site to site during a simulation, while other physical Bodies usually have a fixed site location. In addition to having a spatial location by virtue of being contained in a Site, Body objects also have (optionally) a position inside the site they currently are stationed in. Body objects also contain generic attributes, such as age, that are used in a large number of modelling scenarios. In general, a Body object provides the basic specification under which subclasses such as Agent objects can be defined. Agent objects themselves proffer additional commonly used attributes, such as sex, and a one-dimensional quality label.

In addition to physical (site) neighbour relationships, agents may establish social relationships with other agents. Specifically, each agent maintains a list of social networks (SocialNet objects), each intended to correspond to a different relationship type (e.g. parent-offspring, acquaintance, sexual,

etc.). Each specific agent-to-agent relationship is coded as a Tie object, that holds information about a relationship, such as the agents involved, its intensity, and its duration. Parent-offspring relationships are created and managed automatically by Ethos during agent creation, while other types of relationship are defined and managed by the simulation model.

Different criteria can be set forth to specify which agents are added/removed from a particular social network of some agent. Agent selection in relationship management, or object selection in general, is facilitated by using Selector objects. Selector objects can be employed, with modeler specified criteria (implementations of interface Criteria), to implement a selection of some type of a subset of objects from a larger set (e.g. roulette-wheel, or tournament). Selection modes can be non-competitive, where objects are selected through a local criterium, or competitive, where agents are selected by rank.

Agents can have finite life-span, and may be dynamically created and eliminated during a simulation run. Agents have a genetic makeup (Genome object) which is inherited from one or two parents. Each agent's genome contains a list of genes, whose number, data type, and initial value distributions (when not inherited) are selected by the modeler. The details of the genetic system, such as type of crossover and mutation intensities can be selected from the ones available or defined by the model designer (e.g. the top level Genome class implements a multi-point crossover, while the subclass Genome1PCX implements a one-point crossover (Goldberg, 1989)). The interpretation of genes' values is left to the modeler, but it is conceivable that in the future we will include genes interpreted by ETHOS's runtime system (e.g. properties that modulate agents' behaviour). Furthermore, this is complementary to agents' offspring being able to inherit behaviour control information from their parents.

Agents usually live only in one space throughout their life-span, although they can migrate on demand between different spaces in the world. If an agent migrates to a different space in the world, all its current relationships are deleted. This simplifies the possible distribution of a simulation by putting different Space objects running in different address spaces and different machines. As a result, the scheduling order of events between different Spaces of a World is undefined. Because most models use only one Space object, we defer discussion of distribution issues.

3.2 Event Management and Population Structures

ETHOS uses a simple yet flexible discrete time step scheme to trigger events. Population objects are utilized to aggregate agents and other bodies into collective units, whose event dispatching is coordinated. Each Population object relays control to each member Body object — by calling some well-defined method — according to a set scheduling policy. The policy specifies several things: whether the iterative sequence of members at each time step should be made random or fixed; the number of phases a simulation step has; and whether dispatching of events is asynchronous or synchronous, when more than one phase is involved.

Population objects are also used to code population level operations on Bodies. Population subclasses refine on the base scheduling policy and services provided. AgentPopulation is a subclass specific for Agent member objects. In addition to the inherited scheduling policies, AgentPopulation can be set to allow agent invocation to continue in the same simulation time step until all member Agents have run out of free time. This works in conjunction with the notification of time usage as agents perform actions and use up their time.

Population objects are made to subclass Body. This allows for population objects to be arbitrarily composed in tree- or graph-like structures[2]. Event dispatching occurs in "depth-first" order, as control passed to a Population always dispatches to member elements, and is not aware of super-ordinate Populations other than the parent's. By default, a sub-population inherits the scheduling policy of its parent Population.

Each Space object has an associated top-level population, automatically created at space initialization. This top population is used to add other Populations (or Bodies), thus structuring the event scheduling order. Usually, the scheduling is static and implicit in the Population structure created. However, dynamic schemes are also possible by modification of Population membership during a simulation. Scheduling of events for an arbitrary number of time steps can also be incorporated into ETHOS in the future, if it proves useful (similar to that found in other MAS, such as SWARM and Repast).

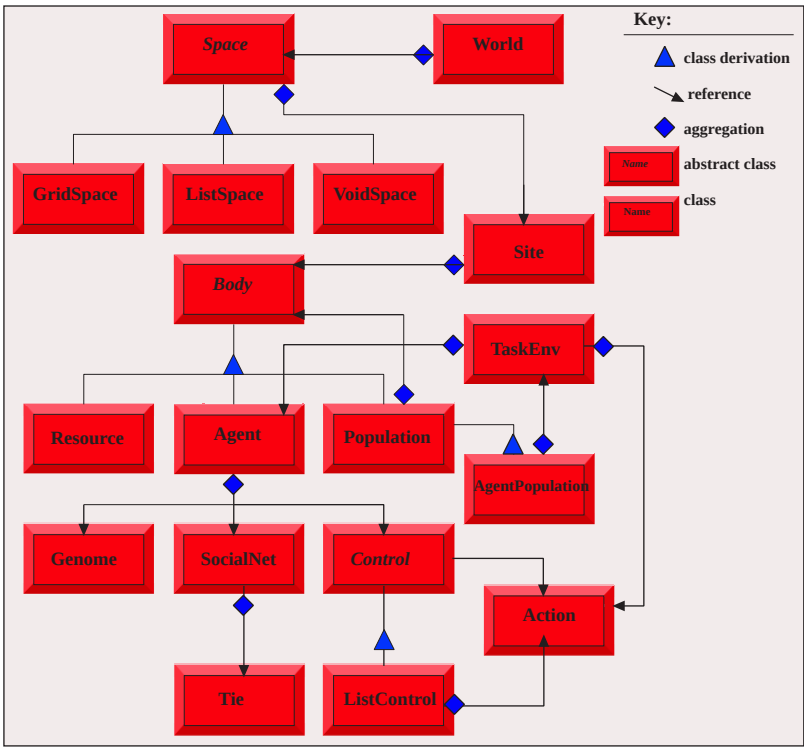

Figure 2: ETHOS's Class Hierarchy

3.3 Agents' Behaviour Control and Task-Environments

Agents can select which action to perform within a simulation time step using a Control object. A Control object maintains all the information that pertains to the Agent's mental state. It is used to decide what action to perform when faced with a context for the action, and it's updated on the basis of the outcome of actions. Action contexts are abstracted using TaskEnv objects. They correspond to opportunities in physical or social contexts for individual and collective action (Reed, 1996). Each TaskEnv has associated an arbitrary (user defined) context identifier that empowers the agent to discriminate between different TaskEnv. During each simulation time step

[2]This corresponds to the implementation of a *Composite* object design pattern (Gamma et al., 1995).

a TaskEnv holds a set of Agent objects, possibly with a role identifier, whose result of interaction is computed. Thus, collective action is represented by a TaskEnv that contains more than one Agent. Sub-classes of TaskEnv can expand the basic services to match specific scenarios. Including: cost–benefit analysis, resource transfer, observation of others attributes and modification of self attributes, etc.

Agents' individual actions are garnered by making a call to the Control of the Agent. This delivers an Action object coding an appropriate response. Once all agents have decided on their actions, they are evaluated by the TaskEnv, and the payoffs are notified to participating Agent. This prompts a call to the Control objects for state update. The state update is usually contingent on each individual receiving a payoff (rewards - punishments), but social learning is also supported by looking at information of related other agents participating in the TaskEnv (see below). By convention, the role of AgentPopulation subclasses is to define and create TaskEnv and assign them to member agents. The workings of a Control are made transparent to other objects, by having an Agent provide the commonly used operations and deferring their execution to the Control [3].

Control class is abstract, and is used indirectly as the modeler instantiates one of its subclasses. At present we support a ListControl subclass that maintains a list of actions of bounded size. When a ListControl is requested to select an action for a TaskEnv, it tries to match one of the Action objects stored with the TaskEnv. By default, Action subclasses match to a TaskEnv if the (perceptual) context identifier of each is the same, but modeler specified subclasses can override this. If a match in ListControl is found, there is a non-zero probability the response of the action will be randomly mutated. This implements a simple exploration mechanism, similar to that of evolutionary strategies (Back, January 1996; Beyer, 2001). If no match is found, a new action is created, derived from an existing one. If the maximum size of the action list is reached, one is selected for removal. Actions whose execution lead to smaller payoffs for the agent are more likely to be removed. In addition to ListControl, we are also planning to provide a neural network based controller that learns by association and reinforcement in the form of a Control subclass.

Control objects also provide several methods to mimic different types of social influence. The method updateByPriming(,) is intended to model the simple types of social influence (Heyes & Bennett G. Galef, 1996). The method updateByObservation(,) is intended to model learning by using others' payoff to update an agent's own control (Bandura, 1977, 1985). In ListControl this corresponds to finding a stored action that matches that performed by another agent and changing its valuation. Finally, updateByImitation(,) is used to model social learning which abstracts how behaviour responses are passed from agent to agent (Boyd & Richerson, 1985). The specifics of the semantics of each of these methods is to be defined by subclasses of Control. They are defined for the purpose of structuring the task of modelling social learning.

3.4 Other Features

Similar to most other MAS frameworks for agent-based modelling, ETHOS provides miscellaneous features within a "ready to use" Graphic User Interface (GUI). These include: the control of a simulation execution; the visualization of the simulation state; the gathering and exporting of statistics resorting to several types of data objects, such as binners and time-series; dynamic parameter setting; amongst others. In figure 3 we display a screen shot of the GUI.

[3] This corresponds to the implementation of a *Facade* object design pattern (Gamma et al., 1995).

In the lower left-hand side, a table of parameters is shown. Parameters can be grouped into logical or conceptually related sets and are displayed according to this grouping. In the lower right-hand side, two viewers are shown, one representing a data plot and another a graphical view of a GridSpace object. In this default GUI object, viewers are all exhibited as internal frames of a desktop area.

At the top of the figure 3, are the controls for setting up, starting, stopping, and stepping a simulation. These are very similar in functionality to those available on the GUI of other MAS frameworks, like REPAST and ASCAPE. The rightmost button allows data objects to be exported to data files, so they can be plotted and analyzed with more specialized tools. Below the control buttons, on the right, a set of progress bars indicate the percentage of the specified number of steps and of runs a simulation has advanced. And to the left of these, a slide bar allows the refresh time of viewers to be set dynamically. This is the time interval after which viewers are updated to reflect their underlying observed object[4]. Also in this area, through a spinner, a time of delay can be imposed upon the simulation progress to perform slow motion execution of the simulation.

In the bottom part of figure 3 the parameter setting panels are shown on the right, and several viewer objects are shown on the left. Since these features do not differ significantly from ones available in other MAS for agent-based modelling, we do not delve into them in detail.

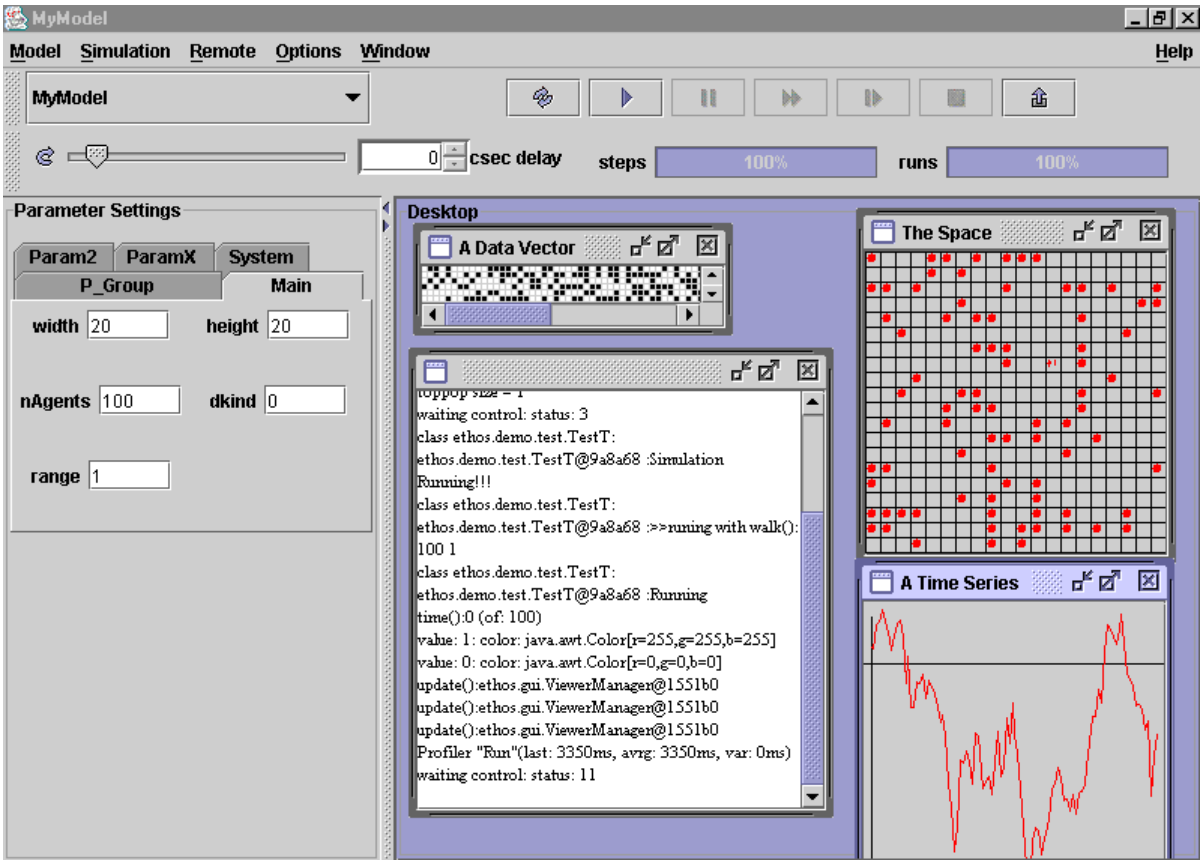

Figure 3: ETHOS's GUI look-and-feel.

[4]A variation of the *viewer–observer* object design pattern is used here (Gamma et al., 1995).

4 Two Application Case Studies

To test the usefulness and generality of our framework, we have re-implemented several agent-based models from the literature, sporting some form of non-trivial social interaction. Of particular interest are models of gene-culture dual inheritance, and inter-personal social dynamics, since they constitute the main application target of ETHOS. We have also implemented new models inspired by the theoretical leverage provided by the ETHOS meta-model. In the following sections, we describe how we implemented two such models, summarizing first the structure of the models, followed by a description of how the abstractions of our MAS framework were employed, and finally the presentation of the modelling results. First, in section 5, we submit a model intended to capture the dynamics of human mating in a monogamous mating system with courtship. We then use this model to make several qualitative predictions that are supported by empirical data. The second model, submitted in section 6, is geared to study the cultural dynamics of trait preferences over time, and glean under what conditions fashion-like collective behaviour will emerge.

5 Human Mate Choice: Case Study I

To model what happens when a set of individuals interacts to find mates, we begin by establishing a population of constant size $2 \times P$ and a fixed sex ratio of 50% (so P is the number of males and of females). Individuals of both sexes have a one dimensional quality parameter q_i, randomly generated from a normal distribution with mean μ and standard deviation σ, truncated such that $0 < Q_{min} \leq q_i \leq Q_{max}$. Time is modelled as a sequence of discrete steps. Pairs of males and females meet at a certain stochastic rate: at each time step each individual has a probability $Y \cdot p_i$ of meeting a new individual of the opposite sex, where Y is a model parameter constant specifying the maximum meeting rate, and p_i is an individual specific discount factor which is dependent on the individual's *interaction capability* (see below). The specific individual j to be met is chosen randomly with a probability proportional to p_j.

Each individual keeps a list of the potential mates already met — the *alternatives list*. The alternatives list has a maximum size of N. This corresponds to maximum number of opposite-sex individuals an agent can maintain in its social network for making courting proposals. If the social network becomes saturated, that is, if the alternatives list is filled, new meetings happen at the expense of forgetting one randomly selected individual already in the list (other than the current partner).

Within the alternatives list, one member can have the "special status" of being the individual's current *date*. This happens when both individuals previously agreed to court and have not changed partners in the mean time (see below). It is also possible for an individual not to be courting anybody (e.g., in the beginning of its "life", or when it gets "dumped"). The length of time that two individuals court one another is referred to as the courtship time c_t. Each individual has a specific *minimum courtship time* (K_i), which specifies how long it takes for the individual to fully commit herself/himself to a relationship, and become willing to mate (metaphorically: to fall in love with its partner). If both dating individuals "fall in love" in this way, then they mate, and do not consider further dating opportunities. Every individual has a maximum *reproductive* lifetime of L time steps, so some individuals might never be able to find a mate.

Since individual reproductive potential is lower at very young ages and reproductive lifetime is

finite, the costs of delaying mating is higher later in life. Therefore, we assume that K_i has a maximum value at the beginning of the individual's reproductive life and decreases monotonically with time. Specifically, we define $K_i = K \cdot (1 - \frac{t_i}{L})$, where t_i is the age of individual i, and K is a constant model parameter defining K_i at age 0.

The interaction capability p_i of an individual is negatively correlated with the degree of involvement in the current courtship process (and therefore with c_t). This is intended to model increasing levels of intimacy and exclusivity as courtship progresses (see (Simão & Todd, 2002) and (Nowak & Vallacher, 1998) for a detailed discussion on the issue). Specifically, $p_i \in [0,1] = max\{0, 1 - \left(\frac{c_t}{K_i}\right)^I\}$, where $I > 0$ is a model constant that defines the shape of the "intimacy curve".

At each time step, every individual i has a certain probability of interacting with every member j of its alternatives list. This probability is set as $p_i \cdot p_j$. We call the set of all alternatives for which an interaction is selected to occur, according to these probabilities, the *interaction list* of i. After the interaction lists are computed for all individuals, each one decides what action to perform based on his or her state: If an individual is single, he/she has to decide whether to try to start a relationship (with some member of the interaction list), or postpone that decision to see if a better alternative becomes available. If an individual is courting, she/he has to decide whether to continue to court the same partner or try to court some other individual. Below, we specify the exact decision functions applied by the agents in our model. Mark that, although an individual can make requests to court several others at each time step, she/he can only court one individual at a particular point in time. (See (Simão & Todd, 2002) for a more formal description of the matching algorithm used.)

We make individual behaviour consistent with evolutionarily plausible constraints by defining a fitness function F, in order that individuals behave in such a way as to maximize it. Specifically, we define $F(q_m, t) = q_m \times \frac{L-t}{L}$, were q_m is the quality of the individual's chosen mate, and t is the age at which the individual mates. Thus, we reward a preference for high quality, and introduce time pressure to motivate individuals to mate early (in addition to the motivation stemming from their limited life-time).

If an individual is already courting somebody else, she/he will switch partners whenever that provides a fitness gain. Specifically, partner switching attempts are made by an individual i if the following inequality holds:

$$F(q_a, t + K_i) > F(q_d, t + max\{0, K_i - c_t\}) \tag{1}$$

where q_a is quality of the alternative partner being considered, q_d is the quality of the current partner (the date), c_t the current courtship time, and t is the age of agent i. The equation specifies that if the expected fitness of mating with the alternative (calculated using an optimistic estimation of the required courtship time K_i) is greater than the expected fitness of mating with the current partner (calculated using an optimistic estimation of the remaining courtship time $K_i - c_t$), then switching should be attempted. For example, if the agent i has just started a date or courtship period ($c_t = 1$), then virtually all individuals with higher quality than the current partner would be sought as alternative partners. On the other hand, if agent i has been courting for a long time ($c_t \approx K_i$), then prospective alternative partners would need a quality somewhat higher than the current partner to be considered as a date, because any switch needs to compensate for the lost investment already spent in the current relationship.

Because there are costs to entering a relationship, a reasonable strategy is for individuals to set acceptance or aspiration levels to decide whether or not to begin courting some partner. Potential partners falling below the aspiration level in quality are not sought (or proposed to) as dates. This aspiration level should reflect to some extent the individual's own quality, with high quality individuals avoiding lower quality ones, and lower quality individuals having realistic aspiration levels tuned to their unfortunate lower rank. Because rationally bounded agents cannot be assumed to have information about their own relative quality automatically — their rank is relative to all other individuals in the population, which they cannot know ahead of time — agents must estimate their quality dynamically and use it to perform mating decisions as they go along.

Specifically, an individual i starts out being totally non-discriminating by setting her/his self-quality estimate q_i^* (here equivalent to the individual's aspiration level) to 0. If an agent was previously dating and the partner j took the initiative of breaking the relationship, q_i^* is updated according to the following rule:

$$q^*_{i_{new}} = q^*_{i_{old}} \cdot (1 - \alpha) + \omega \cdot q_j \cdot \alpha \tag{2}$$

where q_j is the quality of the agent's departing partner, $\omega \in [0, 1]$ is a correction factor to decrease the agent's expectations to slightly below the quality of that partner (we will use the value .8), and α corresponds to the learning rate which is used to avoid individuals moving aspiration levels too fast (we use the value .2 in our simulations). The overall procedure is likely (although not certain) to assign q^* an appropriate value, because the agent's partner will break the relationship only to start courting a higher quality individual — and this gives a rough indication that the agent is aiming too high and is unable to retain partners of quality q_j, so that a new aspiration level less than q_j is called for.

Finally, because the expectations of an individual should reflect not only its own quality but also the availability of partners, aspiration levels should be reduced whenever waiting for a higher quality partner does not pay off in terms of lost reproductive lifetime. Moreover, since the initial aspiration level might not have been properly calibrated, individuals should not be too confident about it. This means it might be advisable to attribute the failure to mate to an inflated value of one's aspiration level — which, in turn, should prompt a drop in the value of q^*. One simple way to model this is to keep track of the time t_w an individual has been waiting for a partner, and to lower his/her aspiration when a waiting time threshold t_{max} is reached. Specifically, we will define this threshold as follows:

$$t_{max} = \tau \cdot \frac{L - t}{L} \cdot (1 - \frac{q_b}{q^*}) \tag{3}$$

where τ is a proportionality constant (we use the value 60), t is the age of the current individual, and q_b is the quality of the best alternative in the alternatives list whose quality is lower than q^*. Intuitively, this threshold specifies that the time an individual is willing to wait to court someone with the current minimal sought quality is inversely proportional to the individual's age and the quality of the best (attainable) alternative likely to be already available. If t_w reaches t_{max} the aspiration level q^* is set to q_b (and t_w is reset to 0).

Overall, this behavioural strategy can be interpreted, metaphorically, as individuals trying to climb up (and sometimes falling down) a ladder of qualities. When accepted by higher quality

individuals, individuals tend to move their aspiration level higher (although no actual update of q^* is performed or required until a relationship breaks). On the other hand, when rejected, or in any case with the passage of time, individuals will tend to move their aspiration level down.

5.1 Implementation in Ethos

To model this with ETHOS, we defined two AgentPopulation sub-classes and an Agent class. The constructor of the Agent class defines two SocialNet objects. One, identified by the integer SNET_AQ, is used to maintain the list of agents of the opposite sex that the agent knows about. This list has a maximum size of NS and the method trim() in the base class is overridden to ensure that an agent's date is not removed from its list of acquaintances. The second social net maintains at most one member — the agent's date. The sex of agents is defined by an integer, set to be 0 or 1. Agents also set their one-dimensional quality parameter, attributed to them at creation. This is used as primary information whenever other agents decide whether to date the agent.

Each agent maintains a reference to a object that is its strategy. In the act() method, called at each simulation time step, the strategy object is used to decide which of the agents of the opposite sex referenced in the TaskEnv object the agent should propose to. Still in the act() method, a Selector is used to add a random agent of the opposite sex to the social network using a criterium that gives all agents an equal probability of being chosen (if they are not known acquaintances already).

```
class MyAgent extends Agent implements Comparable {
  CourtStrategy s = new CourtStrategy(this);
  ...

  MyAgent(int sex) {
    super();
    setSex(sex);
    setQuality(getRandomQuality());
    setMaxAge(L);
    addSocialNet(new SocialNet(this, SNET_AQ, NS) {
       protected void trim() {
         Agent ag = getMember(Global.getRandom().nextInt(getMaxSize()));
         if (ag != getDate()) {
           tbreak(ag);
         }
       }
    });
    addSocialNet(new SocialNet(this, SNET_DATE));
  }

  MyAgent getDate() {
    return (MyAgent) getSocialNet(SNET_DATE).getMember(0);
  }
  ...

  public void act(TaskEnv te) {
```

```
    if (Global.getRandom().nextDouble() <= Y) {
      Agent ag = (Agent) Selector.selectOne((getSex() == 0 ? sex1:
        sex0).getAllMembers(), Selector.EquiCriteria.crit, null,
        getSocialNet(SNET_AQ).getAllMembers());
      getSocialNet(SNET_AQ).addMember(ag);
    }

    for (Iterator i = te.getAllAgents().iterator(); i.hasNext(); ) {
      MyAgent ag = (MyAgent) i.next();
      int action = s.getAction(getDate(), ag);
      if (action == 1) {
        props.add(ag);
      }
    }
    Collections.sort(props);
  }
}
```

The first sub-class of AgentPopulation is used to represent each sex's sub-population. It takes as arguments in the constructor the identifier of the sex and the number of agents created. At each simulation time step, it selects the set of agents an agent will interact with by adding them to a TaskEnv. It also performs population wide operations such as removing from the population mated pairs of agents, updating agents' strategies, and killing agents that exceed the maximum set age.

```
class MyAgentPopulation1 extends AgentPopulation {
  int sex;
  TaskEnv te = new TaskEnv();

  MyAgentPopulation1(int sex, int n) {
    for (int i = 0; i < n; i++){
      addMember(new MyAgent(sex));
    }
    this.sex = sex;
  }

  public void actOne(Agent ag) {
    te.clearAgents();
    MyAgent ag0 = (MyAgent) ag;
    List asex = sex == 0 ? sex0.getAllMembers() : sex1.getAllMembers();
    List ags = Selector.select(asex, InteractionCriteria.crit, null,
      (ag0.getDate() != null ? ag0.getDate().getThisList() : null), 1);
    te.addAllAgents(ags);
    ag.act(te);
  }

  public void act() {
    super.act();
    if (sex == 0) {
      removeMated();
    }
```

```
      updateStrats();
      kill();
    }
  }
```

The second sub-class of AgentPopulation is used to store the two sub-populations, each representing one sex. Its only operation, other than passing control to its sub-populations, is to pair up agents as a result of mutual proposal and acceptance. (For brevity reasons, we have omitted here the details of the proposal making and matching algorithm, full specification of which can be found in (Simão & Todd, 2002).)

```
class MyAgentPopulation2 extends AgentPopulation {
  MyAgentPopulation2() {
    sex0 = new MyAgentPopulation1(0, (int) R*P);
    sex1 = new MyAgentPopulation1(1, P);
    addMember(sex0);
    addMember(sex1);
  }

  public void act() {
    super.act();
    pairUp();  //make new pairs
  }
}
```

Comparing the implementation of our mate choice model(s) using ETHOS with one done from scratch we found some interesting tradeoffs. On the one side, using ETHOS required us to be aware of ETHOS meta-model abstractions, and how to properly use them. On the other, the existence of a conceptual landscape from which objects can be picked and mixed simplifies the cognitive effort in developing non-trivial models. Moreover, the code tends to be much shorter, and the possibility of design and/or implementation errors much smaller.

5.2 Simulation Results

5.2.1 Patterns of correlation in quality

In this section, we investigate the kind of global patterns that emerge from the individual hardwired decision rules and social interaction model described above and see how well they account for the self-organization of real human mating populations. We set the male population size parameter $P = 100$, and the female to male sex ratio $R = 1$, giving 100 males and 100 females in the population. Additionally, we set the individual reproductive lifetime to $L = 200$ (corresponding to 20 years, with each time step a tenth of an year). The parameters for the (quasi) normal quality distribution were set by equating agent quality with the total number of offspring produced during a complete (female) lifetime, using a data set from a particular human population, the *Ache* (Hill & Hurtado, 1996) — although similar values apply to other societies that have not adopted significant contraceptive use. The intimacy constant I was set to 2.0 to model a quadratic reduction of interaction capabilities, which corresponds to a super-linear increase in couples' intimacy as

courtship develops. Each simulation run consists of the pairing and mating of individuals until L time steps are reached. The results shown correspond to averages across 100 such runs.

Figure 4 depicts the linear correlation between the qualities of individuals in mated pairs as a function of rate-of-meeting Y and the initial courtship time K. The results show that the more alternatives individuals meet (as $Y \times K$ gets bigger), the more likely they will mate with an individual close to them in quality. Only a small value of K is required to make the correlation value almost independent of the meeting rate. This suggests that individuals are making good use of their mating potential even if encounters are rare and despite the fact that they have no initial, direct knowledge of their own mate value. Most importantly, the results are in accordance with the reasonably high correlation coefficients (mostly between .6 and .7) empirically observed in sampled human populations (Kalick & Hamilton, 1986).

Furthermore we found that, for the same parameter values, only a small number of dates is required for individuals to mate (mean between 1.4 and 3.2), and the average age at mating time is always lower than $K + 20$. This means that (on average) individuals do not delay mating by searching for partners for long periods of time, beyond the required courtship interval. Moreover, in virtually all simulation runs all individuals in the population were able to mate before the end of their lifetime. This occurs because the sex ratio here is $1 : 1$ and all individuals become less and less choosy over time. If the model is modified so that agents are replaced in the population by new ones as soon as they mate (as we did in our previous models (Simão & Todd, 2001, 2002)), the percentage of mated individuals drops slightly, but its always above 90%. Again, these findings are consistent with demographic data, which indicates that in most human populations between 85% and 95% of individuals are able to mate at least once in their lives (typically under the official seal of the marriage institution) (Kalick & Hamilton, 1988).

Overall, the current model replicated the empirically realistic results we found in (Simão & Todd, 2002), without requiring the previous artificial split between a flirting period to set aspiration levels and a separate period to search for mates. This move to a more plausible psychological design makes it more likely that the model assumptions better approximate the actual causal mechanisms producing the macro-level patterns found in the real world — although further empirical work is needed to establish this. It should be noted that the combination of these three empirically validated statistics — correlated mate values, high rate of matings, and little search — was never obtained in previous models of (human) mating using realistic psychological constraints. Kalick and Hamilton's attractiveness-preference model requires a high number of courtships (or at least individuals met) to achieve a realistic intra-couple quality correlation and a realistic proportion of mated individuals (Kalick & Hamilton, 1986). Todd and Miller's model produces unrealistically low proportions of individuals mated (Todd & Miller, 1999). Finally, a game-theoretical model presented by Johnstone produces statistics similar to ours, but only by giving individuals initial knowledge of the distribution of qualities in the population and their own exact quality, and by assuming that the cost of waiting or searching for a potential partner is constant (Johnstone, 1997). Our model avoids the full-information requirements typical of normative, optimizing approaches used in behavioural research, by assuming bounded rational agents that are able to gather and exploit the rich information structures presented in their task-environments to make robust decisions (Reed, 1996; Gigerenzer, Todd, & the ABC Research Group, 1999).

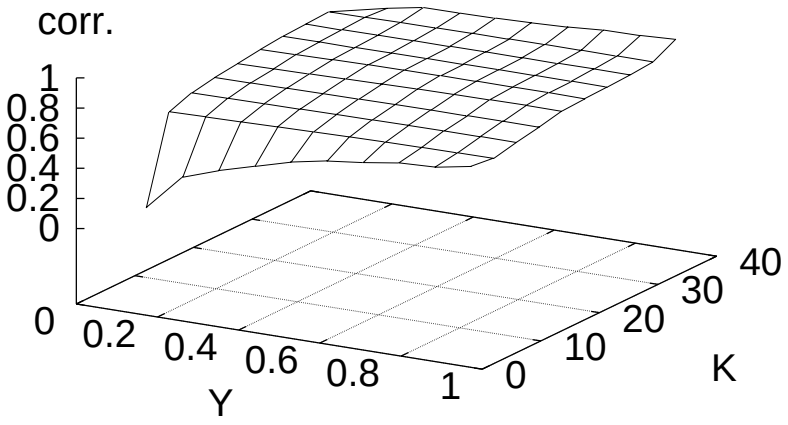

Figure 4: Correlation of qualities of the two individuals in each mated pair, across a range of settings for parameters Y (meeting rate) and K (courtship time).

5.2.2 Distribution of ages at marriage

One of the most robust empirical findings concerning population-level patterns of mating is that the distribution of age at first marriage follows a right-skewed bell curve in many cultures, rising more or less sharply from an early age to a broad peak between 20 and 30 years of age and then trailing off slowly into older ages (Coale, 1971; Todd & Billari, 2003)[5]. How can this consistent pattern be produced through the self-organization of individuals following simple mate choice rules? A parsimonious hypothesis is that this could arise if age at marriage is negatively correlated with an individual's quality, that is, higher quality individuals marry earlier than lower quality individuals. Then, given that quality is normally distributed, this should be enough to create the common right-skewed bell curve. However, we found that our model did not produce such a linear relationship between age at mating and quality. Instead, low-quality individuals ($q < \mu$) show great variation in mating age while high-quality individuals ($q \geq \mu$) show little variation (see Figure 5). This occurs because high-quality individuals mate assortatively with similarly high-quality individuals, who are fewer in number and therefore harder to find than average-quality individuals. Thus, although high-quality individuals are sought after, they are unable to mate much earlier than the average individual (nor do they mate much later than average). As a consequence, the expected overall age-at-marriage pattern does not emerge[6]. Instead, a spike pattern appears with the majority of individuals mating as soon as it is possible, but low-quality individuals spreading the age of mating across the lifespan (see Figure 6). Thus, if the nonlinear relationship between quality and age at marriage holds in the real world as well, then normal variation in individual quality is not

[5]In the analyses that follow, we consider demographically-observable marriage as a stand-in for long-term mating behaviour.

[6]When we made the quality distribution uniform ($q \in [Q_{min}, Q_{max}]$), the relationship between age at mating time and quality become linear, but because the quality distribution is no longer bell-shaped the typical age-at-marriage pattern does not emerge then either.

sufficient to generate the empirically-observed age-at-marriage curve.

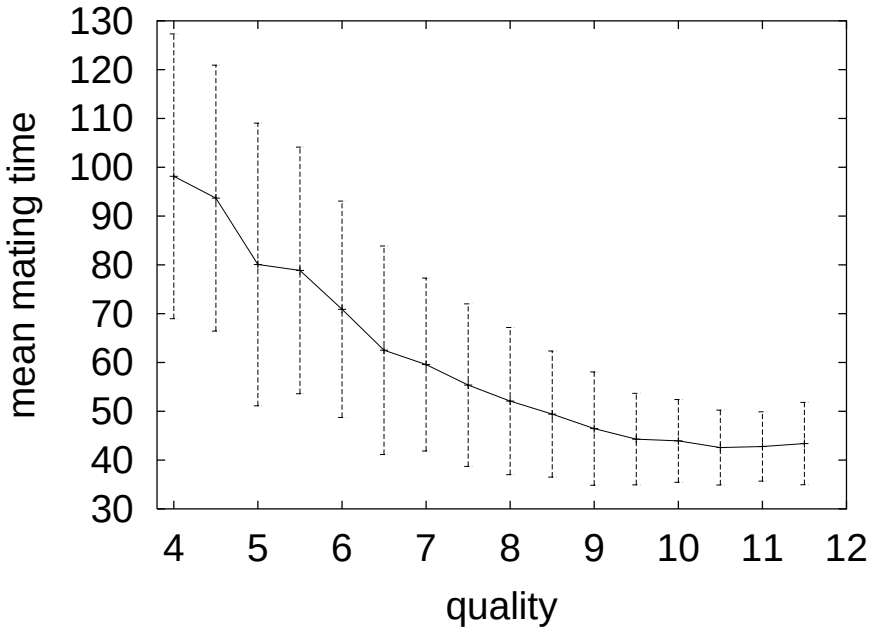

Figure 5: Mean mating time as a function of individuals' quality (with $Y = .1$), for a fixed value of courtship time $K = 40$.

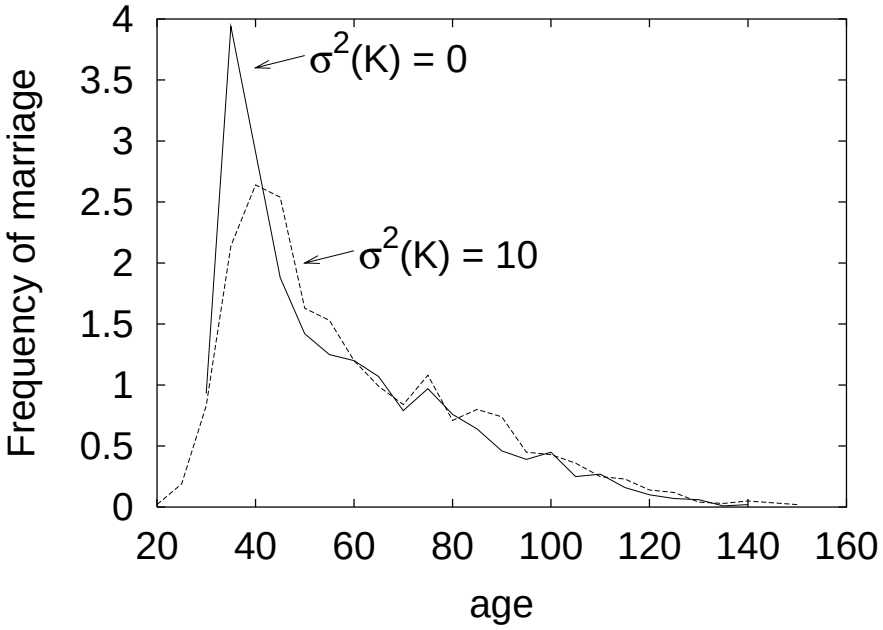

Figure 6: Distribution of age at mating (with $Y = .1$), for a fixed value of courtship time K ($\mu(K) = 40$) and for normally-distributed courtship time with same mean but variance 10.

To match the demographic data through their individual-level mate search model, Todd and Billari (Todd & Billari, 2003) found it necessary to introduce variation in the number of dates (or potential marriage partners) that individuals encounter during adolescence. But because we do not use

a distinct (and artificial) period for setting up aspiration levels in our model, we must look for another explanation to the age-at-marriage pattern. Indeed, the explanation is quite similar: When we include normally-distributed individual variation in the courtship time K, the distribution of marriage times comes much closer to the observed right-skewed bell curve (see Figure 6). This form of individual differences is reasonable to build into the model because variations in social-economic and employment status are known to affect the propensity and ability of individuals to establish long-term relationships (Lloyd & South, 1996; Oppenheimer, 1988). Further comparison against empirical data could help to clarify how much variation in the age of mating (or marriage) can be accounted for by variation in courtship time (as well as individual quality), and in what ways these two factors relate to other individual differences in explaining this striking demographic pattern of human mating.

6 A Model of Fashion Emergence by Conditioning: Case Study II

We assume a population of P agents, with two attributes and two preferences. One attribute is real-valued and remains fixed during the entire simulation runs. It represents, in a nutshell, some quality or cluster of qualities of the agent that is positively valued by all members of the population. For example, physical attractiveness, speech ability, or some aggregated (holistic) perception of several of such traits[7]. We dub this attribute the *quality attribute* or *quality* for short. Its value q_i is assumed to be randomly generated from a normal distribution with mean 0 and standard deviation σ^2. A second attribute t_i is a binary trait, and represents some agent trait or item that the agent either carries with him (when $t_i = 1$), or not (when $t_i = 0$). For example, t_i may represent a type or brand of clothes, or a body ornament. This second attribute can be changed by the agent at will, according to what is perceived by the agent as being the most valued option (from its subjective aesthetical point of view). To distinguish it from the quality attribute, we will call it the *trait attribute* or *trait* for short. Two additional preference attributes represent an agent's subjective perception of what is the value of not having the trait v_i^0, or of having the trait v_i^1. The values of v_i^0 and v_i^1 are updated according to a process of conditioning (or association) between observed traits and qualities, described next. Overall, an agent a_i can be represented formally as a four element vector: $< q_i, t_i, v_i^0, v_i^1 >$.

At each time step, every agent observes a set of N different agent role models. This is the set of agents which influence an agent's current perception of the value of having or not the *trait*. As a very simple abstraction of the process of conditioning, we assume that the value (v^1) of having the trait is changed by an amount proportional to the average of the qualities of the role models that have the trait. Additionally, since we want to explore the effects of memory within agents, we let an agent's new valuation to be influenced by the previous valuation. For this purpose, we define a learning rate parameter α, that specifies how insensitive agents are to new observations. That is, it specifies how slow or how fast agents are in forgetting previous valuations and changing to new ones. Specifically:

$$v_i^1(t) = v_i^1(t-1) \cdot \alpha + \frac{1}{N} \sum_{a_j : a_j \in M_i \wedge t_j = 1} q_j \cdot (1 - \alpha)$$

[7] On the specific topic of mating preference this has been abstracted by Donald Symons as *mate value* or *mate quality* (Symons, 1979).

In the equation above, M_i is the set of models for agent a_i ($\#(M_i) = N$), $\alpha \in [0, 1]$ is the learning parameter, and $v_i^1(t)$ and $v_i^1(t-1)$ are, respectively, the new and previous valuations of trait usage for agent a_i. In a dual manner, we define the value of not having the trait to be the average of the qualities of the models that do not have the trait, weighted by the learning parameter α. Formally:

$$v_i^0(t) = v_i^0(t-1) \cdot \alpha + \frac{1}{N} \sum_{a_j : a_j \in M_i \wedge t_j = 0} q_j \cdot (1 - \alpha)$$

The set of role models for an agent is selected stochastically at each time step, such that the probability of an agent being a model to another agent is proportional to how close their qualities are. This is intended to model a scenario of a socially stratified society, where agents are more likely to interact (or observe) agents that are closer to them in quality than they are to interact with average quality individuals. We implement this by using a roulette-wheel selection mechanism (Goldberg, 1989), and making the probability of agent q_j to be a model of q_i proportional to: $max\{(3\sigma^2 - |q_i - q_j|)^{-E}, 0.001\}$, where the exponent parameter E specifies the degree of social stratification (assortment) in the population. This setting implies that, for positive values of E, individuals with extremely high or low qualities have little chance of interacting amongst them, while individuals with medium quality have maximum probability of interaction. When $E = 0$, the population is not stratified, and individuals have an equal probability of interacting with other individuals with regard to qualities. Thus, higher E values imply higher correlation coefficients between the quality of agents and their role models (numerically illustrated in the next section).

At every time step each agent goes through two phases. In the first phase, the agent is assigned a model set of size N, and it updates its subjective value of having and not having the trait, as described above. In the second phase, each agent a_i decides whether or not to switch its trait attribute value. If $v_i^1 > v_i^0$, the agent will start using the trait by setting $t_i = 1$ (if not set already). Conversely, if $v_i^0 > v_i^1$ the agent will set $t_i = 0$, indicating that it does not have the trait. We assume that, once an agent changes trait usage, it takes at least D time steps until it can be switched again. This is intended to represent a cognitive or material inertia factor (Jager, 2000). Although changes do not occur, the values v_i^0 and v_i^1 continue to be updated.

On top of this, in order to allow the model to escape stationary situations where all or none of the agents have the trait, we define a parameter θ that specifies the probability that an agent will choose a random value for t_i independent of its own valuation (e.g. due to some exogenous factor).

This two-phase procedure can be anecdotally interpreted as follows: individuals go out every day sporting or not the trait (e.g. wearing a certain type of clothes). During the day they interact and/or observe other people and, based on their observations, they update their mental subjective perception of the value of having or not having the trait accordingly (possibly sub-consciously (LeDoux, 1998)). At night, back home, they sleep on the matter. In the morning, with their new sense of aesthetics properly in place, they make a fresh new decision on whether or not to wear the trait during the day. Since, in this version of the model, updates and actions of agents are asynchronous, we should assume, to let the metaphor hold, that each agent performs its updates and trait changes on different days.

6.1 Implementation in Ethos

To implement this model with ETHOS we define one sub-class of Agent and one of AgentPopulation. The sub-class MyAgent, defined below, contains one member field, item, that represents the item or trait value the agent is carrying. Two other member fields, vs1 and vs0, represent the subjective value the agent attributes to having the trait and to not having the trait. The last field memorizes the time step the Agent changed trait value.

On construction, each agent is assigned a random quality and the subjective trait values are initialized. The act(.) is used to perform one of two operations based on the TaskEnv each corresponding to a different scheduling phase. If the TaskEnv is not null them it contains as agents the list of role models that the Agent uses to update the subjective trait values. The other case is when Agents check for trait change.

```
class MyAgent extends Agent {
  int item;
  double vs0;
  double vs1;
  int lastChange = -1;

  MyAgent(double v) {
    setQuality(randomQuality());
    v0 = v1 = v;
  }

  public void act(TaskEnv te) {
    if (te != null) {
      updateValues(te);
    } else {
      changeItems();
    }
  }
```

The code below is executed when updating values. This implements in JAVA code the model equations specified in the previous section. It consists of a simple interaction over the set of role model, taking their average quality and updating the trait valuations.

```
  public void updateValues(TaskEnv te) {
    double v0 = 0, v1 = 0;
    int n0 = 0, n1 = 0;
    for (Iterator j = te.getAllAgents().iterator(); j.hasNext(); ) {
      MyAgent ag = (MyAgent) j.next();
      if (!ag.item.getBit(i)) {
        v0 += ag.getQuality();
        n0++;
      } else {
        v1 += ag.getQuality();
        n1++;
      }
      if (n0 != 0) {
```

```
        vs0 =  A*v0 + (1-A)*(v0/n0);
      }
      if (n1 != 0) {
        vs1 =  A*v1 + (1-A)*(v1/n1);
      }
    }
  }
```

The method changeItems() below is used for trait value switching. Three cases are considered: first, if the agent switched trait less or equal than D time steps, change is not allowed; second, with a fixed probability, a random trait selection is made; third, if none of the two previous cases apply, a switch is made to the trait value that the agent values most.

```
  public void changeItems() {
    if (lastChange >= 0 && getTime() <= lastChange + D) {
      return;
    }
    if ((R != 0) && Global.getRandom().nextDouble() <= R) {
      item = (Item) allItems.get(Global.getRandom().nextInt(allItems.size()));
      mc.add(getQuality());
      lastChange = getTime();
      return;
    }

    int it = getMaxItem();

    if (getValue(it) > getValue(item)) {
      item = it;
      lastChange = getTime();
      nchanges++;
    }
  }
  ...
}
```

The AgentPopulation sub-class is used to dispatch TaskEnv to agents. First, in the constructor, the scheduling policy is set for 2 phases (instead of 1 as default). Also in the constructor, the agents are created and an initial common trait value is assigned to them.

```
class MyAgentPopulation extends AgentPopulation {
  TaskEnv te = new TaskEnv();

  public MyAgentPopulation() {
    setSchedulingPhases(2);
    for (int i = 0; i < P; i++) {
      MyAgent ag = new MyAgent(V);
      addMember(ag);
      ag.setItem((Item) allItems.get(0));
    }
  }
```

The method actOne(,) is responsible for the role of relaying control to each Agent. If, in the first scheduling phase, getModels(.) is used to find the appropriate models by using a Selector operation, the criterium QualityCriteria. Once role models are obtained, they are added to a TaskEnv and control is passed to the Agent. In the second phase, control is passed immediately to the agent for switching of trait value, as described above.

```
    public void actOne(Agent ag, int phase) {
      if (phase == 0) {
        List models = getModels(ag);
        te.clearAgents();
        te.addAllAgents(models);
        ag.act(te);
      } else if (phase == 1) {
        ag.act(null);
      }
    }

    public List getModels(Agent ag) {
      List models = Selector.select(getAllMembers(), QualityCriteria.crit,
                                    ag, ag.getThisList(), N);
      return models;
    }
  }

  static class QualityCriteria implements Criteria {
    static QualityCriteria crit = new QualityCriteria();

    public double score(Object obj, Object ctx) {
      MyAgent ag1 = (MyAgent) obj;
      MyAgent ag2 = (MyAgent) ctx;
      return ag2.pmeet(ag1);
    }
  }
```

6.2 Results

To study the model's dynamics, we set the population size parameter $P = 50$, and the model set size $N = 5$. The standard deviation of agents' quality σ^2 is set to 2. We start by setting the assortment parameter $E = 4$, which generates a correlation coefficient r between the quality of agents and their role models to be approximately 0.75. We set the learning rate parameter $\alpha = 0.2$, and the random change parameter $\theta = 0.04$. We start all simulation runs with no agent having the trait ($t_i = 0$), and we make equal to $-3 * \sigma$ the initial value that agents attribute to having or not having the trait. This corresponds to a situation where trait usage is initially neutral for all agents, and therefore it's indifferent for them whether or not they possess it. We make each simulation run from $T = 100$ to $T = 500$ time steps. In Table 1, we present a summary of the model's parameters and the base values we assign them here.

In Figure 7, we depict a bit map (Wolfram diagram) representing trait usage across time. In the

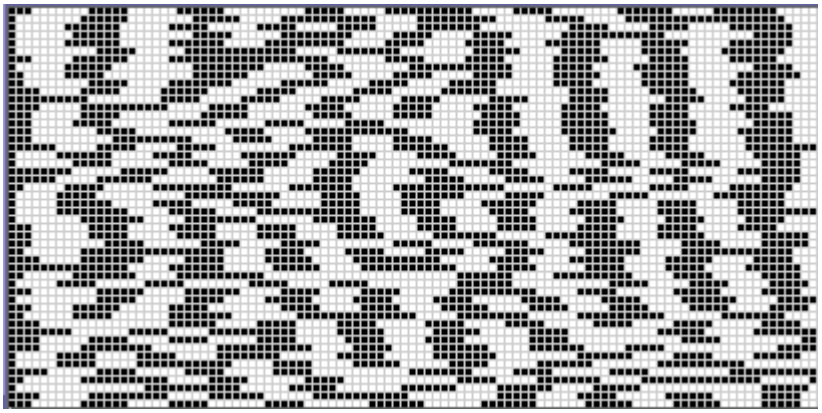

Figure 7: Bit map of trait usage across time ($D = 4$).

y-axis agents are sorted by quality, with higher quality agents towards the top of the figure and lower quality ones towards the bottom. Time runs left to right. As can be seen in the figure, with the parameter values set above, the simulation produces a wave-like pattern. This pattern can be more clearly demonstrated by plotting, in Figure 8, the frequency of trait usage across time. This wave-like pattern emerges because individuals tend to switch to the trait value of higher quality individuals. This feature is responsible for the diffusion of a trait value, but also makes the opposite trait value more preferable over time. However, due to the inertia of agents in switching traits, as specified by parameter D, it takes some time till agents are allowed to switch to back.

Contrary to our initial intuition, the diffusion of the trait tends to occur very rapidly. Just a few time steps are required for a large wave of a particular trait value to emerge. We found out that this is caused largely due to the low (but realistic) degree of assortment. To test under what scenarios diffusion of a trait value would be slower, we made the model selection deterministic such that agents only observe the immediately higher or lower quality individuals. In Figure 9, we present a typical simulation result for this case. As can be gathered from the figure, chance mutations of trait values in high quality individuals produce more ordered spreading of trait usage. However, in this case, the correlation between agents and models raises to a value near 1. This contradicts empirical evidence about modern societies. For example, when considering assortment among individuals of the opposite sex in mating relationships, correlation values between .3 and .8 have been found in attributes such as IQ, physical attractiveness, and others (Kalick & Hamilton,

Parameter	Description	Value(s)	Note
P	population size	50	small sample
N	number of models	5	small
E	assortment	4	$r \approx 0.75$
α	1 - learning rate	0.2	fast learning
σ	standard deviation	2	
D	delay	2	cognitive or material

Table 1: Parameter settings.

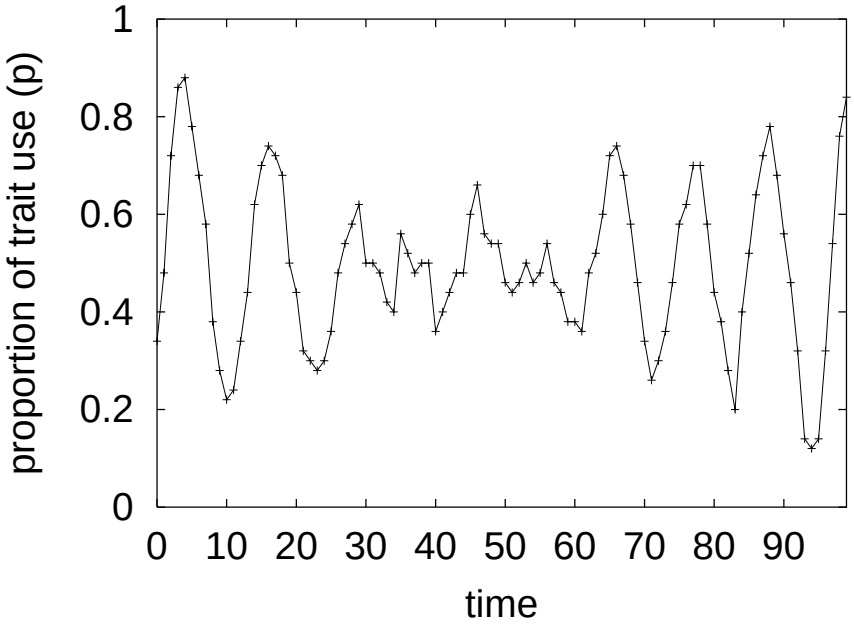

Figure 8: Frequency of trait usage across time ($D = 4$).

1986; Marcus W. Feldman, 2000). Similarly, it has been found in a series of empirical studies that the same pattern of assortment does occur amongst friends of the same sex (Grusky, 2000). This result attracted our attention because some researchers have proposed to use spatial metaphors to represent social status in deterministic small neighborhoods (Pedone & Conte, 2000, 2001). By comparing Figure 9 with Figure 7, it becomes apparent that the way social assortment is modelled can be relevant in obtaining simulation results and deriving theoretical conclusions.

In is useful to compare the model's behaviour with its fixed parameter setting, with standard results from gene-culture co-evolutionary theory. According to R. Boyd and P. Richerson's model of two (discrete) traits' cultural transmission, the trait frequency of those traits with no bias should converge at equilibrium to the ratio of the mutation rates of the two trait values (Boyd & Richerson, 1985). Since in our model the probability of random introduction of any of the trait variants is equal, the frequency of trait usage should converge to 50%. This does not happen in our model because in it we also introduce time varying bias for the preference of trait usage or avoidance. In another model, R. Boyd and P. Richerson showed that, if the bias is fixed, the population should converge to the trait with the highest bias (Boyd & Richerson, 1985). This also does not occur in our model because the bias changes with time. It gets higher whenever medium or low quality individuals still do not have the trait of high quality individuals. This drives the model behaviour into a continuum oscillation between near full trait usage to near full trait avoidance. This can be verified in Figure 8.

7 Conclusions

In section 3, we described Ethos, a MAS framework devised to support the development of cognitive hardwired agent-based models of human social behaviour and cultural change. We described

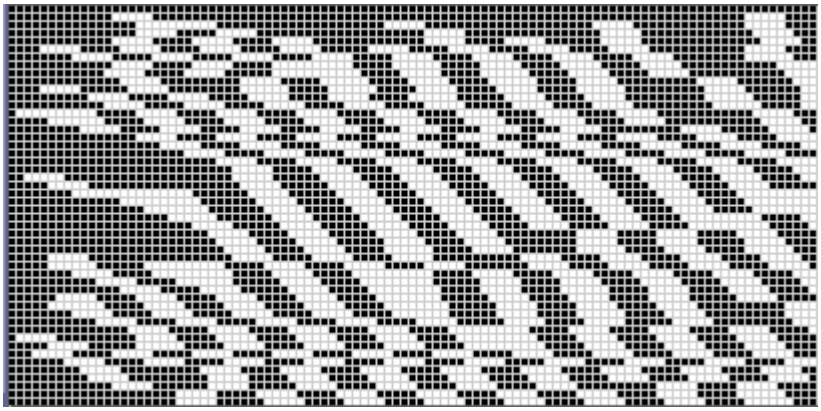

Figure 9: Bit map of trait usage across time ($D = 4$) with deterministic selection of model.

its main abstractions and how they fit together. To test the usefulness of the framework, we presented several examples that use its abstractions. Indeed, in section 5 and section 6, we set forth two novel models that make extensive use of ETHOS, and showed how substantially different models can be expressed naturally through its abstractions. Overall, we concluded that ETHOS's features can simplify the modelling process of a wide range of agent-based models intended to study human social behaviour. The intrinsic tradeoff is that the modeler needs to be aware of the specifics of the framework. We hope our approach can entice other researchers, both to use the proposed abstractions and/or to devise alternative sets of abstractions that prove more useful than the ones provided by ETHOS.

To keep on testing the usefulness of ETHOS's main abstractions (possibly extending and refining them), we plan to implement additional models of human social behaviour, either from the literature or designed specifically for our future studies. The design of an editor to specify models without requiring complete knowledge of the JAVA programming language would also be an important next step. This could include either a mid-level programming language, full support for model creation in a GUI, or some combination of both. Our design philosophy is that a feature should be incorporated in ETHOS only if it used in a wide range of models, and if its availability considerably simplifies model development and testing. This contrasts with the seductive but dangerous design strategy of providing excessive features in a framework, that are rarely or never used by modelers. In any case, although scientifically useful models should be kept simple, it is our working experience that some types of models benefit from additional abstractions, other than just those provided by current MAS. Still, further work is required to learn which of ETHOS's features are most useful and which should be added.

References

Back, T. (January 1996). *Evolutionary algorithms in theory and practice: Evolution strategies, evolutionary programming, genetic algorithms.* Oxford University Press.

Bandura, A. (1977). *Social learning theory.* N.J.: Prentice-Hall, Inc.

Bandura, A. (1985). *Social foundations of thought and action: A social cognitive theory.* N.J.: Prentice-Hall, Inc.

Barkow, J. H., Cosmides, L., & Tooby, J. (Eds.). (1992). *The adapted mind: Evolutionary psychology and the generation of culture.* New York: Oxford University Press.

Beyer, H.-G. (2001). *Theory of evolution strategies.* Springer Verlag.

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

Brookings, R. S. (2000). *Ascape: An agent based modeling framework in java.* (http://www.brook.edu/es/dynamics/models/ascape/)

Bryson, J. (2000). Cross-paradigm analysis of autonomous agent architecture. *Journal of Experimental and Theoretical Artificial Intelligence*, *12*(2), 161–184.

Coale, A. J. (1971). Age patterns of marriage. *Population Studies*, *25*, 87–94.

Collier, N. (2002). *Repast: An extensible framework for agent simulation.* (http://repast.sourceforge.net/projects.html)

Deacon, T. W. (1998). *The symbolic species: The co-evolution of language and the brain.* New York: W.W. Norton & Company.

Donald, M. (1993). *Origins of the modern mind: Three stages in the evolution of culture and cognition.* New York: Harvard Univ Press.

Doran, J. E. (2000). Trajectories to complexity in artificial societies: Rationality, belief, and emotions. In T. A. Kohler & G. J. Gumermman (Eds.), *Dynamics in human and primate societies — agent-based modeling of social and spatial processes* (pp. 89–105). New York: Oxford University Press.

Durham, W. H. (1990). *Coevolution: Genes, culture, and human diversity.* Standford, CA: Standford University Press.

Epstein, J. M., & Axtell, R. (1996). *Growing artificial societies: Social science from the bottom up.* Cambridge, MA: MIT Press.

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1995). *Design patterns: Elements of reusable object oriented software.* Addison-Wesley.

Gigerenzer, G., Todd, P. M., & the ABC Research Group. (1999). *Simple heuristics that make us smart.* New York: Oxford University Press.

Gilbert, N. (2000). Modeling sociality: The view from europe. In T. A. Kohler & G. J. Gumermman (Eds.), *Dynamics in human and primate societies — agent-based modeling of social and spatial processes* (pp. 355–371). New York: Oxford University Press.

Goldberg, D. E. (1989). *Genetic algorithms: in search, optimization & machine learning.* Addison Wesley.

Grusky, D. B. (Ed.). (2000). *Social stratification: Class, race, and gender in sociological perspective.* Westview Press (2nd edition).

Heyes, C. M., & Bennett G. Galef, J. (Eds.). (1996). *Social learning in animals: The roots of culture.* San Diego: Academic Press.

Hill, K., & Hurtado, A. M. (1996). *Ache life history — the ecology and demography of a foraging people.* Chicago: Aldine de Gruyter.

Jager, W. (2000). *Modelling consumer behavior.* Universal Press.

Johnstone, R. (1997). The tactics of mutual mate choice and competitive search. *Behavioral Ecology and Sociobiology*, *40*(1), 51–59.

Kalick, S. M., & Hamilton, T. E. (1986). The matching hypothesis reexamined. *Journal of Personality and Social Psychology*, *51*(4), 673–682.

Kalick, S. M., & Hamilton, T. E. (1988). Closer look at a matching simulation: Reply to Aron. *Journal of Personality and Social Psychology*, *54*(3), 447–451.

Laland, K. N., Odling-Smee, J., & Feldman, M. W. (2000). Niche construction, biological evolution, and cultural change. *Behavioral and Brain Sciences*, *23*, 131–175.

LeDoux, J. (1998). *The emotional brain: The mysterious underpinnings of emotional life.* Touchstone Books.

Lloyd, K. M., & South, S. J. (1996). Contextual influences on young men's transition to first marriage. *Social Forces*, *74*(3), 1097–1119.

Marcus W. Feldman, F. B. C., Sarah R. Otto. (2000). Genes, culture, and inequality. In K. Arrow, S. Bowles, & S. Durlauf (Eds.), *Meritocracy and economic inequality.* Princeton University Press.

Minar, N., Burkhart, R., Langton, C., & Askenazi, M. (1996). *The swarm simulation system: A toolkit for building multi-agent simulations.* (http://www.swarm.org/index.html)

Mithen, S. (1996). *The prehistory of the mind: The cognitive origins of art, religion and science.* London: Thames and Hudson.

Nowak, A., & Vallacher, R. R. (1998). *Dynamical social psychology.* New York: Guilford Press.

Oppenheimer, V. K. (1988). A theory of marriage timing. *American Journal of Sociology*, *94*(3), 563–591.

Pedone, R., & Conte, R. (2000). The simmel effect: Imitation and avoidance in social hierarchies. In S. Moss & P. Davidsson (Eds.), *Proceeding of the 2th workshop on agent-based simulation.* Berlin: Springer.

Pedone, R., & Conte, R. (2001). Dynamics of status symbols and social complexity. *Social Science Computer Review: Special Issue on the Simulation of Social Agents*, *19*(3).

Reed, E. S. (1996). *Encountering the world: Toward an ecological psychology.* New York: Oxford University Press.

Simão, J. P. (2003). *Computational modelling and simulation of human social behavior and culture.* Forthcoming, New University of Lisbon.

Simão, J. P., & Todd, P. M. (2001). A model of human mate choice with courtship that predicts population patterns. In J. Kelemen & P. Sosik (Eds.), *Sixth european conference on artificial life* (pp. 377–380). Heidelberg: Springer-Verlag.

Simão, J. P., & Todd, P. M. (2002). Modeling mate choice in monogamous mating systems with courtship. *Journal of Adaptive Behavior*, *10*(2), 113–136.

Sperber, D. (1996). *Explaining culture: A naturalistic approach.* Oxford: Blackwell.

Symons, D. (1979). *The evolution of human sexuality.* New York: Oxford University Press.

Todd, P. M., & Billari, F. C. (2003). Population-wide marriage patterns produced by individual mate-search heuristics. In F. C. Billari & A. Prskawetz (Eds.), *Agent-based computational demography* (p. 117-137). Berlin: Springer Verlag.

Todd, P. M., & Miller, G. F. (1999). From pride and prejudice to persuasion: Satisficing in mate search. In G. Gigerenzer, P. M. Todd, & the ABC Research Group (Eds.), *Simple heuristics that make us smart* (pp. 287–308). New York: Oxford University Press.