Computational Modeling and Simulation of Human Social Behavior and Culture

Jorge Paulo Ferreira Simão


Orientador: Luís Moniz Pereira

Lisboa 2003
to the better half of mankind...
Acknowledgements

I would like to start by acknowledging the continuous help of my formal and informal advisors. To Luís Moniz Pereira for his help in preparing the formal ground for the realization this thesis, for providing me financial support at different stages of the thesis, and for the useful exchange of ideas. To Peter M. Todd for his willingness to accept me as part of the ABC research group, for the knowledge and insights gained by the close collaboration with him, and for providing much needed additional financial support while I was living in Berlin. To Paulo Gama Mota for providing valuable background knowledge in biology during the initial part of this thesis while I was attending his classes, and for believing that this task was feasible.

Equally warm acknowledgments go to Dr. Gerd Gigerenzer for his reception in the Center for Adaptive Behavior, at the Max Planck Institute for Human Development. I learned quite a lot during the near two year period I was living in Berlin, so my acknowledgments must go also to all my colleges at the ABC group. In particular, I would like to mention the intellectually challenging environment created by John Hutchinson, Richard McElreath, Masanori Takezawa, Clark Barrett, Lawrence Fiddick, Oliver Vitouch, Szymon Wichary, Denny Borsboom and Sanjay Chandrasekharan. I also learned quite a lot by having the rare opportunity to have private meetings with several researchers including John McNamara, John Tooby, Leda Cosmides, and Joanna Bryson. My greatest acknowledgement goes to all of them.

At Coimbra, I also had the opportunity to meet two insightful experimental psychologists which I became great friends of — Miguel Oliveira and Luís Diniz Mónica. Miguel Oliveira later moved to ABC, in Berlin, where I had the opportunity to exchange with him ideas almost on a daily basis. My greatest acknowledgment goes to him. To Prof. Luís Diniz Mónica my appreciation for the never ending spontaneous brainstorms both in informal meetings and by phone. He was also the person who first showed me the book “Simple heuristics that makes us smart” from ABC, which motivated me to go work with Peter M. Todd.

I also want to acknowledge several friends, colleagues, and researchers who provided me feedback on my work and reviewed several papers. To Nuno Preguiça for his comments on working material and for his friendly help well beyond duty. To João Sousa for his elaborated comments on several papers and for being a great friend. To Rainer Hilscher for his help while in the UK participating in HBES and visiting Sussex University. To Luís Correia and Tiago Vaz Maia for
their motivating and inspiring attitude.

Finally, I would like to acknowledge my family without which this thesis couldn’t take place. To my aunt Manela goes a great share of this acknowledgment, and to my mother goes the other share. I would also like to acknowledge the financial support of FCT/MCES, for providing me a four year PRAXIS XXI Ph.D. scholarship, and the financial support of CENTRIA and the Computer Science department of FCT/UNL.
Sumário

Esta tese aborda o problema do uso de modelos computacionais para estudar comportamento social humano e fenômenos culturais. Nós apresentamos um novo sistema multi-agente, designado Ethos, especialmente desenhado para este domínio de aplicação. O nosso sistema vai um passo mais além daquilo providenciado por sistemas multi-agente actuais disponibilizando abstrações úteis à estruturação de modelos baseados em agentes neste domínio. Isto inclui: mecanismos de seleção de acções, influência social através de participação em actividades partilhadas, gestão de redes de relações sociais e despacho de eventos de forma flexível. Nós apresentamos estas abstrações do sistema multi-agente e descrevemos o seu desenho arquitectural e implementação.

Para testar a utilidade e generalidade do nosso sistema, são apresentados dois novos modelos de comportamento social humano. O primeiro é usado para estudar padrões a nível populacional de comportamentos de acasalamento humano, fazendo previsões substanciadas nas áreas de níveis de emparelhamento, estabilidade de relações, idade no casamento e efeitos de ratios de sexo.

O segundo modelo tenta explicar fenômenos colectivos de moda em termos de condicionamento ao nível do indivíduo. Nós concluímos que disponibilizar abstrações computacionais para o domínio em estudo facilita a tarefa de desenhar modelos baseados em agentes e diminui o tempo necessário à sua implementação.
Abstract

In this thesis, we focus on the issue of using computer models to study human social behavior and culture. We present a new software multi-agent system (MAS), named ErHOS, specially designed for this target domain. This framework goes a step beyond what currently available MAS provide by allowing a number of abstractions related to the domain to be used to specify agent-based models in a simple manner. This includes: flexible behavior selection mechanisms, social influence through participation in shared activities, management of agents’ social relationships, and flexible event scheduling. We present the MAS framework abstractions, and describe its design and implementation.

To test the usefulness and generality of the framework we present two new models of human social behavior. The first one is intended to study population patterns of human mate choice, making substantiated prediction on areas such as degree of assortment, relationship stability, age of mating, and sex ratio effects.

The second model tries to explain fashion like collective phenomena in terms of individuals neuro-psychological conditioning. We conclude that providing domain specific abstractions facilitates the task of designing agent-based models and speed its implementation.
# Contents

1 Introduction .................................................. 1

2 Modeling Human Social Behavior and Culture ................. 5
   2.1 Introduction .............................................. 5
   2.2 Agent-Based Modeling .................................... 6
   2.3 The Roots of Individuality and Sociality .................. 7
   2.4 Theories of Human Behavior ............................... 9
      2.4.1 Cognitivism: The Logical-Empirical World View ... 11
      2.4.2 Interactionism: Adaptationist and Bounded Rationality World View ... 14
      2.4.3 Constructivism World View: Neuronal and Epigenetic Robotics .......... 17
      2.4.4 Comparison of Theories ............................. 19

3 Multi-Agent Simulation Frameworks .......................... 21
   3.1 Types of Multi-Agent Software Systems .................. 21
   3.2 Simulation Frameworks for Agent-Based Modeling ....... 23
      3.2.1 Simulation Framework Design Elements ................ 24
      3.2.2 Framework Design and Implementation Requirements ....... 27
   3.3 Generic Simulation Frameworks ........................... 30
      3.3.1 Swarm ............................................. 30
4 Ethos: A Software MAS Framework

4.1 Introduction ........................................... 35

4.2 Framework Overview .................................... 36

4.2.1 Modelling the Physical and Social Environment .......... 36

4.2.2 Event Management and Population Structures ............. 38

4.2.3 Agents Behavior Control and Task-Environments ........... 40

4.2.4 Other Features ........................................ 41

4.3 Design and Implementation ................................ 42

4.3.1 Package ethos.model .................................. 42

4.3.2 Package ethos.util .................................... 48

4.3.3 Package ethos.sim ..................................... 49

4.3.4 Package ethos.gui ..................................... 51

4.3.5 Package ethos.stats .................................... 52

4.3.6 Package ethos.viewer ................................... 53

4.4 Demo Application Case Studies ................................ 53

4.4.1 A Toy World ........................................... 54

4.4.2 Another Toy World: A Simple Cooperation Game ........ 55

4.4.3 Higgs's Mimetic Transition .............................. 59

4.5 Comparison with other frameworks .......................... 63

5 Human Mate Choice ........................................ 65

5.1 Introduction ............................................. 65
5.2 Previous Work on Models of Mate Choice ................................. 67

5.3 A Framework for Modeling Human Mate Choice ........................... 70
  5.3.1 The Nature of Preferences .............................................. 70
  5.3.2 Courtship Processes .................................................. 70
  5.3.3 Time Pressure to Mate ................................................ 71
  5.3.4 Interaction Possibilities .............................................. 72
  5.3.5 Estimating One's Own Quality ...................................... 72

5.4 A Model of Mate Choice with Courtship .................................... 73

5.5 Possible Strategies for Choosing and Switching Partners .................. 77
  5.5.1 Switching Partners .................................................... 78
  5.5.2 Flirting to Set Initial Aspiration Levels .............................. 80
  5.5.3 Combining Aspiration Levels with Partner Switching ................. 84

5.6 Implementation in Ethos .................................................... 84

5.7 Results — Comparing the Strategies ....................................... 87
  5.7.1 Strategy Comparison Across Parameter Settings ....................... 88
  5.7.2 Testing the Utility of Courtship with Alternative Life-histories .... 92

5.8 Testing Model Predictions .................................................. 94
  5.8.1 Patterns of correlation in quality .................................... 95
  5.8.2 Relationship Stability ................................................ 96
  5.8.3 Distribution of ages at marriage .................................... 98
  5.8.4 Effects of skewed sex ratios ....................................... 100

5.9 Discussion and Future Work .............................................. 101

6 The Cultural Evolution of Human Preferences ............................. 105
6.1 Introduction ......................................................... 105
6.2 A Model of Fashion Emergence by Conditioning ................. 108
6.3 Implementation in Ethos ........................................... 110
6.4 Results .............................................................. 113
6.5 Discussion and Future Work ....................................... 117

7 Conclusions ......................................................... 119

References ............................................................ 123
List of Figures

1.1 Suggested chapter reading sequences ........................................ 4

2.1 Organismic Automata ............................................................ 12

3.1 Multi-Agent Software Systems Taxonomy .................................. 22

4.1 ETHOS's Framework Main Abstractions .................................... 37

4.2 ETHOS's Class Hierarchy ......................................................... 39

4.3 ETHOS's GUI look-and-feel. ..................................................... 42

4.4 Package ethos.sim and ethos.gui. ............................................. 49

5.1 Model Structure ................................................................. 75

5.2 Pseudo-code for matching / proposal making algorithm. ............... 76

5.3 Possible agent life-histories .................................................... 77

5.4 Strategies statistics .............................................................. 89

5.5 Sensitivity analysis for strategies .............................................. 90

5.6 Sensitivity analysis for strategies .............................................. 94

5.7 Correlation of qualities .......................................................... 96

5.8 Relationship stability ............................................................ 97

5.9 Mean mating time ............................................................... 98

5.10 Distribution of age at mating .................................................. 99
5.11 Mean mating time ................................................................. 99
5.12 Mean quality of mated individuals ................................. 102

6.1 Bit map of trait usage across time ($D = 4$) ......................... 114
6.2 Frequency of trait usage across time ($D = 4$) ....................... 115
6.3 Bit map of trait usage across time ($D = 4$) with deterministic selection of model. 116
6.4 Bit map of trait usage across time ($D = 10$) ....................... 116
6.5 Bit map of trait usage across time ($D = 0$) ......................... 117
List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1</td>
<td>Parameter settings</td>
<td>88</td>
</tr>
<tr>
<td>6.1</td>
<td>Parameter settings</td>
<td>114</td>
</tr>
</tbody>
</table>
Computers are useless. They can only give you answers.

Pablo Picasso
Chapter 1

Introduction

The history of science is marked by a strong divide between natural and social science. On the one hand, natural sciences are seen as having stable conceptual frameworks and methodologies rich in mathematical formalisms. On the other hand, social sciences are characterized by a more pluralistic approach often with each author laying out his own conceptual landscape (Browning, Halcli, & Webster, 2000). One can think of several reasons why social sciences are characterized so differently from natural sciences.

One hypothesis is that the different disciplines in the social sciences are dealing with fundamentally different topics, therefore any attempt to merge them into a coherent framework is a vain exercise. This is perhaps our first intuition given the wide range of disciplinary sub-divisions found in the literature. However, there is a growing interest in crossing insights from different fields, and an increasing number of researchers believe that the different social sciences can and should develop out of a shared pool of knowledge (Gintis, 2003).

Another possible reason for the fragmentation of the social sciences, is that it was only until quite recently that the appropriate research tools became available. Computer simulations of complex social phenomena, as a paradigmatic example, once the privilege of only a few supercomputing research centers can now be executed on the desktop computer of every researcher (Gilbert & Troitzsch, 1999).

This thesis develops out of two working hypothesis: first, that it is possible to establish generic conceptual frameworks to study human social behavior; and second, that the use of computational tools largely facilitates the process of modeling and analyzing social phenomena of interest. We believe not to be alone in taking these two propositions seriously.

Regarding the issue of developing unifying frameworks for the social sciences, one should say that this is far from being a new theoretical exercise. Many researchers from the fields of biology, psychology, anthropology, sociology, and economics have attempted to bring together different schools of thought and disciplines by making synthesis that are both synergistic and consistent.
For example, the approach of using the Darwin's theory of evolution as a starting point to unify the social and the biological sciences has a long tradition (Laland & Brown, 2003; Barkow, Cosmides, & Tooby, 1992; Alexander, 1979). Similarly, researchers have tried to bridge the views of standard economic theory that posits self-interest and fully rational agents with the real constrains of living in socially structured societies (Coleman, 1990). On the issue of the utility of computational tools, one only needs to point to the large spread of computer modeling and simulation techniques that have been taking place over the last 10 to 20 years. Today conferences on the topic of computer simulation and social phenomena take place on Europe, the United States, and Australia, on at least a yearly basis.

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, from the complexity of the studied phenomena. The lack of a shared language frustrates possible advances in scientific research insofar as it makes harder to compare conflicting models and model predictions, and puts barriers to the communication between overlapping research communities. Computational tools represent a promising avenue to address this problem. Because computational tools require all abstractions to be formally specified (at least up to the programming language level), they need to make explicit what are its ontological commitments. In so doing, they establish the set of building blocks upon which models can be built. Thus providing a common set of abstraction that modelers can use and the community can share.

Notwithstanding this possibility, computational tools employed to support the modeling and simulation of systems with many interacting elements (complex systems) often lack features that allow capturing human specifics 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 modeling 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 behavioral control (Bryson, 2000a). 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 for the sake of mathematical tractability (e.g. focusing on equilibrium states, and the use of infinite populations sizes).

To address these unique characteristics, we propose in this thesis a new conceptual framework for a Multi-Agent System (MAS), named Erthos. It constitutes the central part of the thesis. Our goal is to go a step further than other MAS in providing abstractions that can be used to express easily a wide range of model designs, namely, agent-based models of human social behavior and culture. In chapter 4, we present the MAS overall structure and functionality,
and describe selected aspects of its design and implementation. We further present some simple examples that show how easy it is to express valid models of social and cultural processes with Ethos. The MAS framework was developed complying with the design requisites specified in chapter 3. In particular, we tried to provide abstractions of high expressiveness that could be implemented efficiently. In chapter 3, we also survey existing MAS platforms, describe their main features, and compare them with Ethos. In chapter 2, we present the scientific context of Ethos, which is useful to understand why some abstractions, and not others, were included in Ethos.

To test the usefulness of Ethos abstractions, we present two original models of human social behavior. In chapter 5, we present a model intended to study human mating in monogamous societies. We present results which are backed up by empirical evidence for levels of assortment in mating, relationship stability, distribution of age at mating, and effects of sex ratio. In chapter 6, we present a model used as a metaphor for human cultural change. We take a look at the dynamics of the model, with a particular focus in fashion like collective behavior — that is, continuous change in the trends of trait usage and avoidance. Both of these models were implemented or re-implemented in Ethos. The simplicity and elegance of the implementation of these two models in Ethos is taken to be a first step towards validating our MAS framework. In chapter 7, we discuss our experience in using the services provided by Ethos, pointing out directions to future work, and presenting concluding remarks.

The contributions of this thesis to the understanding and practice of computer modeling and simulation of human social behavior are three-fold: At the conceptual level, it provides an analysis of the ontological possibilities to be considered when creating computer models of human social behavior. That is, which basic categories should be considered and how they relate to each other. This is done in chapter 2 and chapter 4. At the engineering level, it provides a detailed description of the design and implementation of a MAS framework and compares it with other existing MAS. This is done in chapter 4 and chapter 3. Finally, at the scientific level it presents the detailed formulation, implementation, and analysis of two agent-based models used to study two different aspects of the human social behavior. Overall, this thesis represents a multi-disciplinary work that contributes to bridge the gap between several research fields. On one hand, it shows how the process of designing and engineering a software MAS can be aided by close inspection of the ontology of the target domain — in this case, social and cultural processes. On the other hand, it shows how problems apparently hard to address in social science can be elaborated upon a computational context.

Several chapters in this thesis are based on published articles in conference proceedings and/or journals. Chapter 4 is based on a article presented at the 4th Workshop on Agent-Based Simulation (Simão & Pereira, 2003a), and a invited presentation at the Congresso em Neurociências Cognitivas, in Évora, Portugal (Simão & Pereira, 2003b). Chapter 5 is based on an article presented at the Sixth European Conference on Artificial Life, an article published in Journal of Adaptive Behavior, and an article published in a special issue of the Journal of Artificial Life.
that came out as a result of the *Workshop on Self-organization, Evolution and Social Behavior*, Monte Verità (Simão & Todd, 2001, 2002a, 2002b, 2003). Other papers inspiring these and other chapters of the thesis have also been submitted to publication at the moment of publication of this thesis (Simão & Pereira, 2003c, 2003d).

Most chapters in this thesis are self-contained, although certain reading sequences are more natural or useful than others. Figure 1.1 shows a dependency graph depicting the suggested reading sequences. Chapter 2 and chapter 3 are preliminary and can be skipped if the reader is familiar with the issues on modeling human behavior and social phenomena and software multi-agent based systems. Chapter 4 is the central chapter of the thesis presenting the MAS Ethos. Chapter 5 and chapter 6 can be read interchangeably. The reading of this thesis does not require any particular background, although acquaintance with computer science, cognitive science, and/or social science can be quite useful.

![Dependency graph](image)

Figure 1.1: Suggested chapter reading sequences

Chapter 2

Modeling Human Social Behavior
and Culture: Conceptual and
Methodological Issues

I’m astounded by people who want to “know” the universe when it’s hard enough to find your way around Chinatown.

Woody Allen

2.1 Introduction

Broadly speaking, studying human social behavior amounts to characterizing human individual behavior, the types and outcomes of social interactions, and how these two elements relate to generate patterns of culture, social structure and social organization. In social science, this is commonly known to as the micro-macro problem — the micro being the individual level and the macro the one of the collectivity (Coleman, 1990; Frank, 2003). From a general system theoretical point of view, a human population can also be interpreted, at least at the metaphorical level, as a complex adaptive system (Cowan, Pine, & Meltzer, 1994; Holland, 1995; Csányí, 1989). Here we will take an hybrid approach when theorizing about human social behavior considering both ecological and neuro-psychological perspectives. The neuro-psychological perspective allows us to take an informed look at the micro level. The ecological level allows us to put the neuro-psychological mechanisms generating human behavior in the social context, check the result of their operation at the collective level, and see how the social realm both constrains and enables individual behavior. We divide this task into three parts. First, in section 2.2 we lay down the general tenets of the agent-based modeling approach to study human behavior and culture. In section 2.3, we discuss some of the conceptual foundations required to understand
the general properties of human social systems. This will enable us to take perspective on what elements need to be incorporated in an ontology useful to characterize human social behavior and culture. Next, in section 2.4, we survey the main schools of thought on the issue of human individual behavior.

### 2.2 Agent-Based Modeling

In this thesis, we will be mostly concerned with an approach to the scientific modeling of human social behavior and cultural processes often referred as *agent-based modeling*. Agent-based models are a subset of the larger domain of *individual based models* where each entity in the system is represented explicitly (Seth, 2000). Agent-based modeling adds to this the representation of system's entities as agents performing actions and interacting with each other in some physical or social arena to produce global patterns. The advantage of this approach, when compared to other computational modeling approaches, is that the mapping between the domain ontology and the model ontology is often more natural. Some researchers have claimed that agent-based modeling is a very general modeling approach that includes as special cases other modeling techniques such *micro-simulation* and *cellular automata* (Brassel, Mohring, Schumacher, & Troitzsch, 1997; Gilbert & Troitzsch, 1999).

Agent-based modeling is often used as a complement or alternative to analytical techniques, often known as *mean-field analysis*, in which aggregate variables are used to represent selected aspects of population structure or behavior. Such approach has the advantage of allowing the model relations to be expressed in a straightforward manner. However, formal analysis can be hard if not impossible to derive even for very simple models (Bak, 1996). On the other hand, agent-based models can handle situations sometimes hard to model analytically such as heterogeneous populations, finite populations, local knowledge, discontinuities and discreteness. The disadvantage is that the analysis of the model is an hard and tedious process. This includes postulating and testing relations between the different model variables (Paolo, Noble, & Bullock, 2000) and performing careful sensitivity analysis of the results (Suleiman, Troitzsch, & Gilbert, 1999).

A further distinction that should be made here is between agent models aiming at quantitative predictive accuracy, and models where predictions are only made at the qualitative level. While quantitative predictive accuracy is desirable, at this stage of research development in social science qualitative predictions is often the most one can hope. This is the type of approach we will pursue in this thesis. Additionally, because models with many interacting agents are often very sensible to the assumptions made about agent's behavior, part of the research strategy is to experiment with many related models. This allows a through investigation of the system at hand, and testing the robustness of the models. Often the goal of a model is to derive a broad range of results from minimal assumptions. As put by R. Boyd and P. Richerson (Boyd
As a matter of practical scientific strategy in biology and the social sciences, the use of general theory built up from simple models is not a substitute for a more sophisticated method, but the most sophisticated possible given the complexity and diversity in the subject matter.

A related approach in agent-based modeling is to eschew from predictive accuracy altogether and instead focus on the intrinsic properties of the model being studied. This can allow important properties of the model to be investigated and then by analogical reasoning to infer what the behavior of the real system is. This approach, often termed as artificial societies, is often advocated as a principal avenue to study human complex systems (Kohler, 2000). The difficulty in deploying this approach is that it can leave too much space for improper interpretation of model results. In this thesis, whenever data is available, we try to check model validity against empirical data.

2.3 The Roots of Individuality and Sociality

A recurrent topic that arises when studying human behavior is that of knowing to what extent behavior is innate or “pre-wired” in individuals’ brains when they are born, or it emerges on the individuals’ behavioral repertoire as a result of the history of interaction with the physical and social environment. In spite of continuous debate over this issue, modern neurological evidence suggests a perspective more in line with the non-nativist (constructivist) end of the spectrum of theoretical positions. For example, a growing body of evidence is being accumulated showing that much, if not all, of the regional cortical specialization in vertebrates nervous systems emerges not due to direct genetic/molecular instructions, but because different regions have different (afferent) input channels (Elman et al., 1996).

In fact, several authors have suggested that vertebrate brain may be organized around relatively simple design principles. For example, E. T. Rolls suggests that the vertebrate brain may be organized around a reward/punishment system, with organisms trying to get the most rewards and the least punishment possible (Rolls, 2002, 1999, 2000). Thus a functional component of brains would be involved in establishing what the rewards and punishments are (e.g. hunger, thirst, affiliation, etc.) and when to activate them — with the remaining components subservient to this “behavior policy”. A similar proposal made by J. Panksepp is that vertebrate brain is structured around a small number of emotional or affective systems (Panksepp, 1998). This includes components such as seeking objects or individuals of value, distress when separated from individuals of affection, sleep and control of arousal, fear and rage, etc. Panksepp uses the metaphor of emotional systems in lower brain structures working as a ROM computer memory whose content is pre-wired by genes and the development process, while upper brain regions are
more like an EPROM that is written by elaboration of the basic behavioral instructions from below.

Likewise, W. H. Durham proposes that cultural values can be divided into two groups: primary and secondary. Primary values are those whose function is directed to the spread of genes — that is, they maximize reproductive or inclusive-fitness and therefore would be selected by natural selection; secondary values are those which individuals learn by trial an error and/or social learning (Durham, 1990). Durham argues that most often secondary values are in line with sociobiological predictions of inclusive fitness maximization (Barkow et al., 1992), but concedes that this is not necessarily so. For example, individuals can be indoctrinated or forced to follow orders of leaders, which satisfies only the leaders interest. The limitation with Durham’s account is that he does not clarify in detail how such secondary values arise. At the neuro-psychological level this would be equivalent to find a functional locus of the implementation of values.

When individuals interact with each other, they tend to receive informational patterns more similar amongst themselves than random. This includes reinforcement and punishment regimes, object perceptions, and experiencing similar action possibilities. Thus, if the same set of individuals interact frequently, this has the effect of generating convergence on the set of mental representation governing individual behavior. That is, social interaction, can with time, make individuals resemble more to each other (Dautenhahn & Nehaniv, 2002; Heyes & Bennett G. Galef, 1996). This process is often referred as imitation, but it includes a wider set of mechanisms. M. Tomasello proposes the following typology: learning by exposure, stimulus enhancement, emulation, mimicking, and (true) imitation (Tomasello, 1999/2001). In learning by exposure individuals, usually youngsters, are attracted to areas where skilled conspecifics are, and by virtue of being exposed to the same spatial area, they learn individually some task. In stimulus enhancement, this learning occurs because individuals are brought attention to certain objects manipulated by others, and this makes youngster more likely to use them, and therefore learn from them. In emulation, behavior acts are facilitated by skilled individuals exposing some desirable/undesirable end state. Mimicking is route copying behavior patterns even when no connection with the goal is made. For example, as when a parrot memorizes words spoken by humans. Finally, (true) imitation involves the ability of recreating possibly complex behavior acts by observation of a performer.

Tomasello and other authors have suggested that a milestone in human evolution is the ability to develop true imitative skills, something which is argued to be lacking in wild chimps, our closest relatives. For example, A. Meltzoff has gathered compelling evidence that children develop competent imitative skill from very early age (Meltzoff, 1996). An additional mechanism proper of humans only is language which allows symbolic representation of complex behavior sequences to be transmitted.

Behavioral convergence takes a particularly consequential form when it refers to youngster individuals in a generation learning the skills of the previous generation. This allows cultural
traditions to persist and transcend a single individual lifetime and is the proper subject matter of anthropology. (Durham, 1990; Boyd & Richerson, 1985; Sperber, 1996; Dunbar, Knight, & Power, 1999). In general, however, whenever a context for shared activity exists there is the potential for a cultural tradition to be formed. Some examples include a nuclear or extended family with familiar traits and traditions (Alexander, 1979), a ethnic-group with a common set of ethnic markers (McElreath, Boyd, & Richerson, 2003; Alexander, 1979), a set of interconnected political beliefs and ideas in the form of a political ideology (Heywood, 1998), or a scientific community trying to solve a related set of problems (Kuhn, 1962).

To notice is the mental and physical economy of social learning. By copying others, individuals can learn new skills and get novel information without the risks and time waste of individual learning (Boyd & Richerson, 1985). An implication of this is that while individuals will in general be motivated to pursue their interests by process of individual learning (based on their system of rewards and punishments), they will do so necessarily using the mental representations they have learned socially. Thus, individual action is constrained by the options that his/her culture made available to him/her. For example, cultures tend to develop stable arrangement of coordination and resolution of conflicts in the form of shared social norms (e.g. traffic rules, and ethic mores). While individuals will act to further their interest, they will be constrained be the social norms learned in their culture's framing (Scott, 2000).

In the case of humans, social interaction can be more diverse and complex than simple convergence of behavior. For example, individuals can engage in complex forms of resource exchange, resource sharing, and mutual help (Fiske, 1993). Moreover, since individuals have multiple needs to satisfy, and there are multiple ways to satisfy each need, with time, individuals tend to build social networks that structure their social life. These social networks define which individuals agents will interact with and which coordination tasks and social transactions they will engage in. For example, one network might define an extended family with which the individual exchanges strong mutual help, and another network might define the wider list of friends with which she/he performs general service exchanges.

The above discussion allow us to set up some of the basic ontological elements to be considered when modeling processes of human social behavior and culture. In chapter 4 we will introduce a MAS framework that includes many of these elements, including: individual agents, social networks for social affiliation and navigation, definition of contexts for collective action, and mechanisms for selection of individual action and social learning.

2.4 Theories of Human Behavior

Central to understanding human social systems (at least at the conceptual level) is the need to devise suitable models of human social behavior. Ideally, such models should be formally operationalizable so that mathematical and/or computer models can be built to understand and
predict the behavior of collectives.

The conceptual structure of a behaving human is very similar to the general case of a performing agent. In figure 2.1 we represent this as an automata or organism in continuous interaction with its environment. The central part of the figure represent the central nervous system of the agent abstracted as performing some computation. The inputs to the system represent the sensorial flow of information through the agent senses. The outputs of the system represent the motor responses of the agent. The sensory flow and response flow is mediated by the agent’s body. Both the input signal and the output signal are assumed vectorial and of high-dimensionality. This captures the fact that high-order animals possess a large number of inputs or sensor elements and a large number of motor actuators — in the order of the hundreds of thousand (Maturana & Varela, 1998). Thus $s_i$ and $m_i$ are contained in finite but large cardinality sets. The time units used to “sample” the interaction of the organism with the environment is made discrete without loss of generality, and can be as small as desired. A discrete model is used here because it is more intuitive and more straightforward to model arbitrary dynamics.

At each point in time the organism is simultaneously receiving input from the environment and generating a motor response. The body structure and dynamics can be arbitrarily complex (e.g. with joins with many degrees of freedom (Hasan, 1991)). Because of this performing motor control can be a difficult task when programmed beforehand. Different theories of human behavior (or other high order animals) differ in the approach taken to explain the performed purposeful behavior. This will be discussed in the following sections.

The goal of the agent is to maintain itself viable, and does this by pursuing a set of goals or values coded in its nervous system. A primary set of goals needs to be assumed as basic to bootstrap the agent’s interaction with the environment (George N. Reeke & Edelman, 1988). During the lifetime of the agent she may find spatio-temporal correlates between the satisfaction of its primary goals and other stimulus or behavior routines. This will typically be coded as secondary values. Thus an agent can be simultaneously satisfying its goals and establishing new ones. For example, in a robotics experiment presented in (Sporns, Almassy, & Edelman, 2000) the authors show how a robot using a brain inspired control architecture can learn that toy blocks have a positive or negative value by virtue of co-occurring with desirable or undesirable stimulus (stripes and blobs). In fact, the robot learns to approach the desirable blocks and to avoid the undesirable ones.

A point to be noticed relates to whether the goal of a theory of behavior is to symbolically describe behavior or to inform the construction of mechanisms to generate behavior (Clancey, 1997). Behavior descriptions require symbols referring to the observed behavior regularities to be postulated by the researcher and to be interpreted by him. This includes the objects or cues the agent observes in the environment as well as a categorization of the behavior responses made by the agent. On the other hand, behavior generation requires mechanisms involving continuous adaptation of the organism to its environment (see below). In particular, this implies that
the way in which an agent represents information about its environment and about its motor actions might be quite different from the ones envisioned by a researcher looking only at behavior descriptions. Because in this thesis we are interested mostly in making simple models simulating collections of interacting agents, most of the complexities of actual behavior generation can be avoided. Most often in a simulation it is enough to describe behaviors as a series of *if S then R* rules.

Another relevant point to understand our automaton is the primacy of motor action over sensorial perception. The agent is in a continuous delivery of motor actions in order to pursue its goals. Sensorial information is used to regulate the ongoing flow of activity, not to instruct it (Maturana & Varela, 1998). This is in contrast with the more intuitive but misleading idea that an agent is in a continuous series of perception–planning–action cycles. Assuming that perception has primacy is to suggest that the represented object needs to be further perceived by a next stage of processing. This leads to an infinite regression that disguises itself in the form of the well known *homunculus* problem — an “intelligent” component that has the ability to interpret the perceptual representation, but which is left unspecified. Indirect evidence to support the perspective of primacy of motor action is provided by work on experimental neuroscience. For example, in (Arbib, Érdi, & Szentágothai, 1998) the authors describe experiments using transplanted members to vertebrate backbones of amphibian to show that semi-random networks of inhibitory and excitatory neurons situated in the spinal chord can spontaneously self-organize to code for simple motor control patterns. This simple motor patterns can include movement such as semi-structured jerking or locomotive motion. The authors suggest that the nervous systems is engaged in forming purposeful routines for action at the lowest levels of the nervous system even before they are modulated by sensorial inputs (other than proprioception).

This aspect of the primacy of action over perception, provides another contrasting point to separate models of behavioral description from models of behavior generation. In the former case, one can assume that the agent acts in perception–planning–action cycles. As noticed above, in simple models this is often captured with simple *if S then R* rules. In the later case, however, the agent is performing a continuous action in the time frame that would otherwise be categorized in a behavior model as a single atomic response.

### 2.4.1 Cognitivism: The Logical-Empirical World View

In the cognitivist or logical-empirical (LE) tradition organisms are modeled after an intelligent mathematician or logician. Usually they assume that the environment is fully predictable and knowable (Russell & Norvig, 1994). Because of this assumption the world can be fully modeled using some formalism (e.g. first-order logic). This is done by postulating the kinds of entities extant in the environment as a set of symbols (e.g. objects, relations, and predicates in first-order logic). The problem left to the organism is then to solve some given well-defined problem. This is done by searching the space of possibilities for action giving a sequence of instructive motor
actions as response.

- Symbolic Reasoning and Planning

In symbolic reasoning the world is abstracted by defining an ontology using some logical formalism (Rich & Knight, 1991). For example, a chess game configuration could be modeled as a set of piece—location pairs. The set of game configurations $S$ represents the state space of the problem. The effects of the set of action $\{a_i\}$ is then defined (intentionally, as opposed to extensionally) for each game configuration. To solve a problem is to find for a given scenario $s_i \in S$, the best possible move $a^*$. Thus, the design effort of a symbolic reasoning approach to cognition takes two components: first, the definition of appropriate formalism to represent knowledge about world states; and second, the definition of algorithms to efficiently search many possible states (Rich & Knight, 1991).

Planning algorithms work with a similar symbolic flavor except that: the output of a planner is a usually a sequence of actions $a_0 \cdots a_n$ to be performed; the input is a goal $g$, and a set of operations represented as a trio $< a_i, o_i, c_i >$, where $a_i$ is an action, $o_i$ is the effect on the world state after performing $a_i$, and $c_i$ are the preconditions on the world state to allow the execution of $a_i$ (Russell & Norvig, 1994). The task of the planner is to assemble a sequence of action that lead to the satisfaction of goal $g$. Every time the world state changes the world data base is updated, requiring the planner to compute new plans again.

A difficulty often meet in practice with this type of approach is that it does not work well when concerned with interaction in the real world (as in robotics). There are three main reasons for that (Pfeifer & Scheier, 1999). First, the representation of the world state needs to be constantly updated whenever something changes in the environment. Second, the symbols used to represent actions and precepts need to be mapped to real motor actions and sensed objects of perception. Third, in some applications all this needs to be done in real time. In a simulation model these issues are not necessarily prohibitive, because
the time constraints can be arbitrarily defined and the model of the agent’s environment interprets the symbols representing the agent’s actions and precepts.

- Rational Choice Theory

*Rational Choice Theory* (RCT) is a family of theories for *decision making* under situation of *risk* and *uncertainty* (Doyle, 1999; Dupuy, 1999; Russell & Norvig, 1994). Decision making is defined, within this context, as the process of choosing a preferred option or course of action from a set of alternatives \{a_1, \ldots, a_n\}. RCT presupposes that individuals assign preferences to to world states. Moreover, it assumes that an action does not have necessarily a fixed outcome (thus, the uncertainty). Instead, individuals, need to estimate the probability that particular actions lead to specific outcomes. RCT postulates that preferences can be defined by a partial order relation. That is, the preferences relation is reflexive \((x \leq x)\), antisymmetric \((x \leq y) \Rightarrow \neg(y \leq x)\), and transitive \((x \leq y \land y \leq z \Rightarrow x \leq z)\). In practice, preferences are usually defined by a *utility function* \(U(x)\) that assigns numerical values to the desirability of world states (thus, imposing a total order to preferences (Russell & Norvig, 1994)). Due to the uncertainty factor, the (average) value of selecting an action, defined as the expected utility (EU) or utility of selecting an action \(a_i\), is the average utility of the possible outcomes weighted by the probability that those outcomes will occur. Formally, \(EU(a_i) = \sum_{x \in \Omega} U(x) Pr(x|a_i)\), where \(\Omega\) is the set of world states. Given this assumptions, it can be proven that an individual will carry out optimal decisions (in terms of satisfying its preferences), if she always chooses the action \(a_i\) that maximizes its expected utility (Doyle, 1999; Dupuy, 1999; Russell & Norvig, 1994). When applied to strategic interaction in the form of game-theoretical scenarios the rationality criteria is extended so that each agent knows that all other agents are rational (Gintis, 2000). This is often enough to guarantee that a unique optimal decision criteria can be defined and computed by agents — called *Nash equilibria,* such that no change in action can deliver a better payoff to agents.

RCT has been widely used as a modeling tool within the fields of economics and political science (Frank, 2003), and to a lesser extent in sociology (Coleman, 1990). Its success arises from its simplicity, universality, and mathematical tractability. In spite of its wide applicability, RCT has some drawbacks. A large body of psychological studies carried out in the last decades show that, in many situations, actual human behavior deviates considerably from what is predicted by the principle of maximization of the expected utility (Kahneman, Slovic, & Tversky, 1982; Plous, 1993).

First, it is known that individuals are not particularly good at estimating probabilities. Instead they tend to use heuristic methods to estimate the likelihood of uncertain events which makes them prone to error and deviation (Kahneman et al., 1982; Plous, 1993). A controversial interpretation of such results lies in the lack of rationality of human beings (Kahneman et al., 1982). However, this view bypasses (or takes for granted) one important required features of animal and human brains — categorization (Edelman, 1987; George N. Reeke & Edelman, 1988). Real world scenarios are never exactly the same,
thus symbolic representation of events (at least implicitly) always presuppose the process of assigning sets of similar scenarios a single label. As Alfred North Whitehead has put it: "We think in generalities, but we live in detail". Thus, in the real world the concept of likelihood of a (repeated) event is secondary to that of defining what constitutes the event in the first place.

Another key problem with RCT is that it puts too strict requirements on the way preferences are defined. First, the complete ordering of preferences implies it is always possible to make tradeoff between options. Formally, taken almost literally from H. Gintis, given any finite set of outcomes $A_1, \ldots, A_n$, ordered from least to most preferred, there is always a probability $p_i$ such that the agent is indifferent between $A_i$ and a lottery that pays $A_i$ with probability $p_i$ and pays $A_n$ with a probability $1 - p_i$ (Gintis, 2003). This is clearly not the case for humans (neither other animals): What do humans prefer: oxygen or water? richness or love? Moreover, preferences change with several factors such as the emotional state, the exposure to new options, aging, and context of choice. In general, continuous selection between changing desires and behaviors has been proposed to be an important element of organisms psychological makeup (Bryson, 2000b).

A further difficulty with RCT is that it implies that an individual cannot get worse if more options and information is available to him/her. Thus, the pursuit of optimality suggests individuals acquire as much information as possible. For example, RCT suggests that individuals (should) look at all possible individual attributes of an option or outcome, weigh their value, and aggregate to compute expected utilities. However, due to real time constraints, the possible cost associated with acquiring new or updated information, and the intrinsic cognitive limitations to use large amounts of information, humans tend to limit their search.

For all these reasons, RCT is more often used as normative benchmark against which behavioral studies of decision making are compared than as descriptive tool of human behavior. (Although, it is also widely used for engineering purposes within the paradigm of symbolic AI (Russell & Norvig, 1994).) This state of affairs has prompted many researchers to propose many amendments, and alternatives to the basic theory. The commonality of which is often put under the umbrella label of bounded rationality (see below).

### 2.4.2 Interactionism: Adaptationist and Bounded Rationality World View

Interactionist and adaptationist theories of human behavior try to apply concepts and methods used in the study of other animals to humans. Specifically, it is assumed that organisms have traits that were selected for their benefit in terms of reproductive output or fitness consequences. Such traits, referred as adaptations, constitute the set of elements from which the agent behavior is selected. Each adaptation is defined as an agent internal solution to a task-environment. A task-environment is a particular set of activities an agent performs autonomously to exploit environmental resources. Because being adapted to more task-environment is often useful, this
approach tends to view brains and minds as having a large number of modules. As discussed below, this can be problematic because it requires extensive pre-defined internal design (Fodor, 2001).

- Adaptive Control

Following the growing discontent with symbolic tradition in AI, in the late 1980's a group of researchers started a new approach to robotics and AI system design. This new approach, termed *new AI* or *situated AI*, focuses on the design and study of complete creatures in interaction with their environment. The working hypothesis is that intelligence is better characterized as the relation between an agent and its task-environments rather than by disembodied problem solving (Pfeifer & Scheier, 1999).

In embodied experiments, robots with simple architectures are built which, when observed by humans, seem complex. The design philosophy is to incrementally add new layers to the blueprint of an agent corresponding to new tasks or refinements of existing tasks. For example, R. Brooks has developed the *subsumption architecture*, which is defined as a simple set of state machines connected by lines of control, that have been used to create simple robots (Brooks, 1986). For example, a robot was built to search for soda cans and to release them in the trash bin. A simple task, but that eluded much of traditional AI approaches due to the complexity of the robot's environment. A common design rule is that behavior components operate independently, each implementing a fixed sensorimotor relation.

While the basic line of approach of the *new AI* is still much in its infancy, a few basic problems exist. First, at the engineering level it is hard to scale complex architectures where all behavior responses are fixed beforehand (notwithstanding advances in this area (Bryson, 2000b)). Second, it assumes that the set of task-environment can be fixed and defined by the designer. This closes the possibilities for open-ended learning — clearly at odds with the intuition of how humans behave. A. Clark calls this hypotheses to agent design *biological incrementalism* (Clark, 2001). That is, obtaining more and more intelligence with more and more pre-specified design.

- Bounded Rationality

Bounded rationality is a "buzz" word for a series of approaches to understand and model human decision making processes alternative to the full rationality model of RCT. This approach, pioneered by Herbert Simon, assumes that human cognition can be described as a collection of heuristics or *rules of thumb*. For example, a satisficing heuristic can be used to search for items by establishing a minimum level of acceptance rather than computing search costs as prescribed by RCT (Gigerenzer & Selten, 2001). This is in opposition with RCT because it does not posit that humans have unlimited knowledge or unlimited computational or cognitive capabilities. Furthermore, when applied to strategic (game-theoretical) interactions agents are assumed to use simple mechanisms to change
their current strategy — such as imitating the most successful individuals, thus making more plausible assumptions about agent’s psychology than RCT (Gintis, 2000).

Recently this approach was cast with an adaptationist stance by some researchers that try to explain or justify the origin of some heuristics as possible adaptations to human ancestral environment where most evolutionary pressures took place (Gigerenzer, Todd, & the ABC Research Group, 1999). They argue not only that simple heuristics can be good enough for making decisions, but also they can, in some cases, be more advantageous than optimization techniques. Their research methodology is to gather data sets of particular environmental structures (e.g. frequency distributions of cue values for a set of objects), and see how heuristics perform in those data sets. They claim that heuristics can be more efficient and more robust to generalization than RCT methods because heuristics are domain specific and use little information.

A conceptual problem with the bounded rationality perspective on human cognition and action is that it fails to separate behavior description from generation. This makes heuristics look like reifications of internal structures in human agents. As suggested above, simple decision rules should best be seen as tools for research inquiry, rather than necessarily corresponding to direct behavioral mechanisms of organisms. Thus, in a simulated model an agent can be characterized by one or few decision rules or heuristics, without this implying a simple correspondence with the underlying mechanisms generating behavior.

- Evolutionary Psychology

The field of **darwinian or evolutionary psychology** was born as a movement to apply the **adaptationism** program, so successful in the study of animal behavior, to the human and social sciences (Barkow et al., 1992). In this theoretical position, the human mind is composed of a set of domain-specific psychological mechanisms shaped by the process of natural selection (Tooby & Cosmides, 1989; Simpson & Kenrick, 1997; Crawford & Krebs, 1997; Pinker, 1997). These mechanisms constitute adaptations to humans ancestral environment that allowed inclusive-fitness relevant problems to be solved. Methodologically, darwinian psychology aims to create computational models of human psychological mechanism by performing a task decomposition analysis of adaptive problems and proposing plausible psychological mechanisms that implement the respective functions.

Although it is not a formal theory, as the others discussed above, this perspective is very much in the same line of thought as adaptive control and bounded rationality. As a consequence, at the programmatic level, there are also some problems with this approach. Because no constraints are put on what counts as a domain-specific module, a simple one-to-one correspondence between mechanisms and functions is often (although often implicitly) implied. This leads to the massive modularity problem mentioned above.

One way to solve this problem, proposed by some researchers, is to assume that adaptive tasks can be grouped into a fixed set of **core configurations**, each one consisting of functionality intermeshed adaptive problems and associated mechanisms (e.g. specific to different types of dyadic social interactions, collaborative team work, social norms regulation at the
group level, and linguistic mechanisms to support more large scale forms of social organization) (Caporael & Baron, 1997). Moreover, phylogenetic newer mechanisms are viewed as making use of the functions provided by older ones, while new synergies emerge from the interactions between the complete set of mechanisms and the environment.

This proposal mitigates the problem of massive modularity, because the set of adaptive context is smaller. However, it does not explain how novel skills, not possibly present in human ancestral environments, can arise (e.g. musical, writing, or mathematical competence). A “semantic” solution to this problem is to postulate that new functions are “exapted” (Gould, 2002). That is, they piggy-back in some “mysterious way” on existing adaptations. This seems unreasonable, though, because the details of the mechanisms are never specified.

In general, a common difficulty with adaptationist accounts of behavior is that they equate behavioral function with pre-existing design, either by natural selection in the case of humans and other animals (Dawkins, 1996), or pre-engineered behavioral routines in the case of robots. Thus, they exclude the possibility for new complex functions to arise during ontogeny. A problem that constructivist theories can address.

2.4.3 Constructivism World View: Neuronal and Epigenetic Robotics

Constructivist perspectives of human and animal behavior and cognition development are based on the premise that the innate constraints of nervous systems design have limited effects on mature behavior. Instead it is assumed that complex behavior repertoires emerge from the continuous adaptability of the organism to the environment during its ontogenic development. More specifically, behavior arises due to the continuous interplay between environmental structures and the structuring process of the brain. For example, in the field of developmental cognitive psychology, A. Karmiloff-Smith proposes that children's domain-specific knowledge in linguistic, physics and psychology develop out of simple genetic pre-dispositions (Karmiloff-Smith, 1992; Karmiloff-Smith92, 1994). These pre-dispositions give a “small but significant kickstart by focusing the infant’s attention on proprietary inputs” (Karmiloff-Smith92, 1994).

In the neuroscience community an increasing number of researchers have provided evidence and theoretical accounts to support constructivist perspectives in developing brains and cognition. In (Quartz & Sejnowski, 1997) the authors propose that brain structural development should be seen as weaved with its functional development:

[...] The central problem confronting a cognitive system is to find an appropriate class of representations for specific problem domains. Many views suppose that these representations have to be pre-existing, but constructive learning builds these under

---

1John Tobby's personal communication.
the influence of the environment, acting alongside the general constraints that are imposed by the neural architecture.

That is, information structures in the environment shapes the way the nervous system self-organize during development — a process that possibly continues until adult age. This contrasts with selectionist theories of brain development that propose that the development occurs in two phases: first, the brain develops to a state where there is an over-production of synaptic connections; second, the information structures found during learning scenario decide which synapses should be eliminated or used (Edelman, 1987). In contrast with this, Quartz and Sejnowski propose that neural structure and cognitive function are contingent on learning episodes.

The constructivist perspective puts the burden of proper cognitive development of an infant in the physical and social environment as much as in the nervous system. In other words, proper physical and social structures are required to provide the necessary scaffolding to the child’s development (Reed, 1996; Rutkowska, 1994a, 1994b). For example, a child might be placed in a eating chair to constrain her body movements, and to allow her attention to focus on manipulative tasks with objects provided by a caretaker (Reed, 1996; Rutkowska, 1994a, 1994b). A related piece of evidence suggests that the type of social interaction a person has during his/her childhood carries effects to its romantic and affective relationships later in life (Zeifman & Hazan, 1997; Miller & Fishkin, 1997).

Looking back to our automata, the constructivist approach requires agents to thrive even when starting life with little knowledge of the world. That is, they start life with the ability to perform very simple behavior responses and with limited ability to discriminate between stimulus. This should be enough to build complex behavioral routines (e.g. by recombination or generalization of simpler ones), and to increase stimulus discrimination (e.g. by using the simpler stimulus as initial references of attention). Helping the system to bootstrap is the structure of the world, which enforces that future task-environments are similar to past ones and allows the agent to reuse the behavior routines it has learned in the past. Additionally, whenever expected events occur the physical location and body posture of the agent can be used to constrain or “show” what other possibilities for action (affordances) exist (Turvey & Shaw, 1995; Reed, 1996).

Although realizing the constructivist approach in its full computational detail remains elusive, many roboticists are working in this line of development (Weng et al., 2002; Weng, 2002). For the roboticist perspective, the interesting aspect of a constructivist perspective is that it presents the prospects for robots to learn new tasks and adapt to task-environments not setup or envisioned upfront by the designer (Weng et al., 2002; Weng, 2002). That is, the robot could continuously adapt to new physical or social environment to maintain its viability. For example, a legged robot could adapt the locomotion style to a new clumped part of its niche.

A consequence of a constructivist perspective of human behavior is that it makes behavior look
like an historical process\textsuperscript{2}. This occurs at the individual level — which continuously acquire new skills or refine old ones, but by implication also at the societal level. This makes intuitive sense because human cultures vary considerably, and within each culture individuals vary amongst themselves and across time. This means that behavioral models are always a truncated view of human behavior, liable to loose predictive accuracy when removed from its initial observation context and time. A corollary of this is that some models are intrinsically more robust than others. Those models, whose symbols tap on more stable behavior regularities will have their predictions applicable in a larger set of contexts. In contrast, models that take too much particulars in consideration are more prone to loose their validity. Our research strategy, as presented in this thesis, is to consider behavior regularities and population pattern stable across a wide range of contexts (as elaborated in chapter 5), or to postulate behavioral mechanisms that are thought to be universal in humans (see chapter 6).

2.4.4 Comparison of Theories

We close this chapter by putting in contrast the theories of behavior presented above. First, it should be noticed that the symbolic reasoning approach is commonly portrayed in sharp opposition to the adaptationist perspectives such as adaptive control. Symbolic reasoning can be seen as a high level abstraction for mental reasoning while adaptive control is more in tandem with ongoing flows of activity of agents’ interaction with the environment. As suggested above in the presentation of our automata, the focus of adaptive control in real world organism–environment interaction is well taken as a starting point to understand situated behavior. However, adaptive control accomplishes this by taking the radical step of wasting away the importance of mental representations. Thus, it is not clear how new functions can emerge in this paradigm. On the other hand, most approaches in symbolic reasoning have difficulty in dealing with real world interaction. Moreover, while it is most likely that mental representation of precepts and motor actions need to be included in a theory of human behavior, it is not clear that these representations can be abstracted in the way implied by symbolic approaches. (See (Bryson, 2000a) for a review of hybrid approaches.)

Second, while both symbolic and adaptative theories fall short of providing a complete answer to the puzzling question of understanding human behavior, they are at this stage better developed at the computational level than constructivist approaches. This is because constructivist approaches have an holistic character which is harder to capture within the reductionism and modular methodologies of science and engineering. Lacking a complete answer to the important question of how purposive behavior is generated one must be satisfied with making informed guesses of what may be driving behavior in each particular case.

Finally, the descriptive theories, such as bounded rationality (and RCT), should be seen as

\textsuperscript{2}The notion of historical is used here in the sense that the constraints that define the system’s dynamics are not fixed (Paolo, 2001).
such: models that abstract with rules what may otherwise be complex causal mechanism in the human brain. Because of this, models built with their support should better be evaluated by their predictive accuracy regarding aggregates of agents rather than by the explanatory power regarding the causal mechanisms generating behavior. In many cases, this is not overly problematic because we are interested in simple abstractions of complex behavior rather than rich and accurate individual specifications of mechanisms that make aggregate dynamics hard to comprehend. On the other hand, this tends to give models an ad-hoc character rather than fitting into a single unified theory. As noticed earlier, in this thesis we follow the approach of using simple descriptions of agent’s behavior but at the same time (whenever suggestive knowledge is available), making the attempt to tap on the underlying mechanistic explanations of behavior.
Chapter 3

Multi-Agent Simulation Frameworks: A Survey

In the beginner’s mind, there are many possibilities; in the expert’s mind, there are few.

Shunryu Suzuki, in *Zen Mind, Beginner’s Mind: Informal Talks on Zen Meditation and Practice*

In this chapter, we focus on the topic of multi-agent systems for simulation of agent-based model. In section 3.1, we start by situating multi-agent systems for agent-based modeling in the wider spectrum of multi-agent systems developed and studied in the context of different research traditions and disciplines. In section 3.2, we develop further the discussion of simulation frameworks for agent-based modeling. In section 3.2.1, we describe in some detail key services and abstractions provided by these frameworks, and in section 3.2.2 we discuss additional computational requirements for these software systems. Finally, in section 3.3, we survey several generic simulation frameworks well-known and widely used by the agent-based modeling community.

3.1 Types of Multi-Agent Software Systems

Broadly defined, one can define a *Multi-Agent Software System* (MAS) as any computational system where multiple software components (the agents), interact in order to perform some computation (Weiss, 1999). Clearly, this often used characterization is very encompassing, mainly due to that the notion of *agent* is understood in different ways by different research communities. In the Artificial Intelligence literature the concept is mostly associated with some notion of “smart” behaving entity that interacts with other “smart” entities: humans or other agents (e.g. as in a desktop assistant). In the agent-based modeling community an agent usually
refers to a software abstraction of some real world entity that has capabilities for acting in its environment (e.g. an animal, a firm, or a country). And in the distributed computing community, there is not a clear distinction between what is called an agent in a software system, and the traditional notions of program, object, component, or process (Szyperski, 1998).

For the purpose of situating the particular kind of multi-agent software system that is most relevant to our work, we characterize these systems according to three main dimensions: number of agents involved, the complexity of the agents computation, and the degree of interaction between agents. Figure 3.1 depicts a graphical representation of the taxonomy present below.

- Agent-based modeling MAS

Multi-Agent Software System for agent-based modeling refer to simulation frameworks intended to support the modeling of the structure and dynamics of some real world system (Gilbert & Troitzsch, 1999). The number of agents involved is usually large, because they intend to model complex systems with many interacting entities. Because the purpose of this models is to study in a rigorous way the detailed behavior of the system, agents are usually light computational entities. Additionally, the interaction between the agents usually follows a simple protocol, and the complexity of the model arises from the intense interactions of the large number of agents. This feature makes, in general, the physical distribution of the agents as a performance boosting technique infeasible. On the other hand, distribution can be used to parallelize the execution of different runs of a simulation possibly running with different model parameter settings. This is the kind of MAS we will mostly consider in this thesis.
• Distributed Computing and Distributed AI MAS

Traditional distributed computing systems when involving “smart” agents and complex interaction protocols, as practiced in distributed AI applications (DAI), present very different engineering aspects than simulation frameworks (Wooldridge, 1999; Wellman, 1999). In these systems the number of agents can also be very large, specially if no centralized computational bottlenecks are introduced in the system design. However, because the degree of interaction between agent is not so intense as in agent-based models, physical distribution is feasible even if interaction protocols are somewhat complex (e.g. multiphase resource locking protocols in distributed transactional databases (Ullman, 1988)). The complexity of the agent computation may vary from application to application, but because physical distribution is feasible they can be potentially very complex. MAS for DAI applications often rely on formal modeling techniques to model aspects of the system, namely, the agent’s behavior and interaction protocols (Weiss, 1999). This contrasts with the engineering of many distributed computing system, which take a more informal approach unless critical application are involved (e.g. defining formal object replication semantics to allow a transparent or graceful handling of system fault). Although the ability to formally model and debug the emergent behavior of a complex system from its individual components is quite welcome, for most models this can ne mathematically untractable (Bak, 1996). Therefore, most techniques used in DAI research might turn out to be quite inept in the context of simulation framework for agent-based modeling.

• Cognitive Modeling MAS

A third class of MAS is used in modeling complex psychological processes. In these systems, agents’ internal architecture capture, with some level of realism, aspects of the cognitive architecture of natural living agents (Rosenbloom, Laird, Newell, & Rosembloom, 1993). When several of these agents are made to interact they become a MAS (e.g. two communicating agents used to model a social interaction or a language acquisition process). Because of the possible complexity of these agents computations and interaction, and the need to study and understand the system dynamics in detail, only a small number of agent are usually involved. Distribution might be an option or even required, as in a case of a group of interacting and/or cooperating autonomous robots, or it can be dispensable when fast-enough computing resources are available.

3.2 Simulation Frameworks for Agent-Based Modeling

Although one can use agent-based models to study many application domains and specific problems, there are certain features that are common to a large class of such models. Thus, while a (social) researcher can implement a supporting software infrastructure each and every time it designs a new model, it is clearly advantageous to factor out this shared functionality into a reusable framework. The goal of such modeling and simulation frameworks is to reduce the work
required by researchers to build models and simulations, and to provide standardized tools for configuring, running, and analyzing results from a simulation. The use of simulation frameworks reduce the duplication of effort by the modeling research community, and speeds up the development process. Factoring functionality and making good use of specialized expertise also means that performance issues of key software components can receive due attention, and that frequent design and implementation pitfalls are more likely to be avoided. Computer programming can also be a complicated technical task, with much room for error, specially for non-experts. As put eloquently by one group of researchers:

[...] computer modeling frequently turns good scientists into bad programmers. Most scientists are not trained as software engineers. As a consequence, many home-grown computational experimental tools are (from a software engineering perspective) poorly designed.

(Minar et al., 1996)

Additionally, social researchers prefer and tend to think in terms of high-level abstractions, more suitable to their target domain, rather than in computational aspects and implementation details. For example, an economist is more comfortable in thinking about a system made of producers and consumers, rather than objects and interaction protocols. Thus, they will usually be willing to leverage on the experience of other modelers and the competence of skilled software engineers. While for very simple models social researchers may prefer to code their model without much specific modeling support, for more elaborate models and longer projects this becomes less attractive. On the other hand, a poorly designed simulation framework might bring some disadvantages such as the loss of transparency of model semantics, in addition to an extra learning curve.

3.2.1 Simulation Framework Design Elements

Simulation frameworks for agent-based modeling provide design elements to map real world domain elements into model abstractions. At the most general level, we can identify four broad categories of abstractions: time and concurrency management, spatial structure, agent structure, and agent interaction\(^1\). Later in the chapter 4, we will use this basic categorization together with the concepts introduced in chapter 2 as a spring board to carve the abstractions provided by our MAS framework.

- Time and concurrency management

\(^1\)Not surprising, this corresponds loosely to the four basic ontological elements common to all physical systems: time, space, matter, and energy (transfer).
Time and concurrency management facilities are used to specify what happens in a model and when. Two main approaches are used: defining a structure of events to be triggered at each discrete time step of a simulation, or scheduling of events to arbitrary points in time within a reference time-line. In the former case the event structure is maintained after the execution of the events, while in the later case the schedule events are “consumed” when they are executed. In both cases the scheduling can be made static, in which case the order of events is maintained during the execution of simulation, or dynamic where the order of events is defined and/or modified during the execution of a simulation. The simplest possible form of scheduling is to define a static event ordering for every time step using a sequence of instructions in the underlying programming language. This simple scheme is, however, limited in several respects: it cannot be easily changed during a simulation if a model requires that, and it does not allow composibility of events naturally. To overcome this, some sort of reflective capabilities can be implemented in a framework that allow a particular set or sequence of instructions to be modified, extended, or merged (Zimmermann, 1996; Kiczales, Rivières, & Bobrow, 1991). When scheduling of events is made to arbitrary points in time, these issues are handled naturally. The price to pay, though, is that the dynamics of the simulation might not be easily observed by simple inspection of the code.

- Spatial structure

Spatial structures allow to represent the physical arena where the model dynamics takes place. There are many useful spatial structures that can be used, with the most common being a discrete two-dimensional grid space (e.g. as in a two dimensional cellular automata (Wolfram, 2002)). Some models do not require an explicit spatial structure (e.g. a social network model), or it can be so simple that can be abstracted in a trivial way (e.g. a population genetics model with two different environments (Boyd & Richerson, 1985)). In general, however, the structure of the space can have important effects on the results of a model. For example, in (Pepper & Smuts, 2000) the authors present a model where they show that the resource distribution in a landscape can influence the ease and type of cooperative behavior that can evolve in a population — in this case, feeding restrain. For this reason, frameworks usually provide several ready to use spatial structures (e.g. continuous space, torus like two-dimensional structure, space with a arbitrary graph-like topology, etc.) Additionally, spatial structures can be defined as composable — with one component of the space being itself a space with its own internal structure. In practice, simple models tend to use a single level spatial structure.

- Agent structure and aggregations

All agent-based models include collections of interacting objects which represent particular target domain entities. These objects are usually broadly designated as agents, and although the name suggests some form of “aliveness” and “pro-activiness” they can be used to refer to entities as diverse as companies, countries, human beings, animals, or cells. The level of specificity a framework imposes in a model design for these objects may
vary across a broad spectrum. In the simplest case objects can be just regular objects in a object-oriented language, leaving all the specific detail to be defined by the designer. For example, data fields can be used to represent attributes and methods can be used to represent operations performed by/on the agent (Gamma, Helm, Johnson, & Vlissides, 1995). Alternatively, the specific layout of the agent’s internal structure might be more directly supported. For example, by providing agents with abstractions for cognitive mechanisms such as behavior selection (Bryson, 2000b), neural networks based learning mechanisms (McLeod, Plunkett, & Rolls, 1998; O’Reilly, Munakata, & McClelland, 2000), or a learning by feedback and reinforcement such as in a classifier system (Goldberg, 1989). Additionally, it might also be interesting to support hierarchical structures such that agents can be made of inner agent (e.g. a firm or a households can be modeled as a collection of individuals that have both their personal agendas but that are stockholders in collective agent (Dean et al., 2000)). Although some frameworks do provide features to support this kind of hierarchical composition, typical models tend to use only one and occasionally two levels of agent structural complexity.

Since agents behavior is studied both at the individual level and in the collective, abstractions to aggregate agents in collective units or populations are also convenient. These populations can be used either as units of agent management (e.g. creation and destruction), structural units (e.g. to create an assemble of agents at a certain level of hierarchical abstraction), or as units for event scheduling and synchronization. Some frameworks collapse all these complementary functions within a single abstraction, while others separate the functions. While the former option as the advantage of being more parsimonious and regular, the latter might have some flexibility advantages.

- Agent interaction

Several mechanisms may be provided to support different kinds of interaction in a multi-agent simulation. In the simplest case, agents interact only indirectly through their shared environment. Changes made by one agent to the environment influence other agent behavior by virtue of the later responding to the altered state of the environment (e.g. due to depletion of a shared resource (Jager, 2000), or through pheromone dispersal as in social insects (Bonabeau, Dorigo, & Theraulaz, 1999)). Alternatively, agents can interact directly by having their behavior observed by other agents, or by targeting other agents with messages or signals. Framework support for direct interaction is in the simple case made by invoking methods in other agent’s objects, but more elaborated communication mechanisms may be provided.

A slightly orthogonal issue to the mode of direct agent interaction is with who do agents interact? When a spatial structure is laid out in the model, a common arrangement is to have agents communicate with each other in their physical neighborhood. Alternatively, agents can communicate through established social networks. These networks can be statically defined according to some model designer setup, or they can be built by using some selection criteria over the collection of available agents. A third option, is to have role
assignment scheme with some form of shared directory or blackboard where agents can obtain references to agents that suit a particular structural role. Potentially, these two last options can be used in combination to create complex social structure (e.g. as used in computational models of organizations (Lomi & Larsen, 2001)).

• Other features

In addition to the four main types of abstraction described above, frameworks also provide a set of tools useful to run the simulation in scientifically controlled and transparent environment. Foremost in these tools, is the ability to collect statistics about the behavior of the model, and possibly to visualize them. Visualization of parts of the model state may also be important in some simulations, whether it is a view of complete layout of agents in a landscape, or more narrowly the attribute values of a specific agent in that landscape — usually referred, as probing. The question of how useful is to visualize the behavior or results of a simulation while it is still running depends on the type of model, the phase of development, and the working experience of the model designer. Often, it is good enough to have a crude mechanism of collecting statistics about a simulation, and later process the data in specialized and sophisticated statistics and/or visualization packages or applications.

Other tools include the ability to easily manage the parameter setting of the model, and simplify the often tiresome process of sensitivity analysis. A mundane but rather import service is the provision for good quality random number generator for common statistical distributions.

Finally, the ubiquitous deployment of computer networks — either local, or the Internet — also makes very appealing to have distribution and parallelization services to capitalize on idle CPU cycles and fasten the process of gathering results of a simulation.

### 3.2.2 Framework Design and Implementation Requirements

In addition to the specific tools and abstractions described in the previous section, there are other more generic computational requirements that simulation framework aim to meet in their design and implementation. Some of these requirements are common to most system level support platforms — both for low-level OS services (e.g. performance, and scalability) (Tanenbaum, 1987), and other typical of domain specific middleware infrastructures (e.g. balanced tradeoff between expressiveness and flexibility). Below, we enumerate some of these requirements:

• 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 key abstractions of the application domain, and should anticipate the relationships between these abstractions. However, this should not be so rigidly done as to constrain the model
design space too much. While some multi-agent systems are designed as a specific model with many parameters (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 balance between providing only a set of loosely interconnected features that might be hard for the model designer to understand the envisioned ways of its use, and a framework that is so specific that can capture naturally only a small application domain. (It should be noticed that this point underlines the general theme of framework design and the (arguably fuzzy) distinction between a framework and a software library (Gamma et al., 1995)).

Object-oriented languages are particularly helpful in satisfying this goal. Due to its mechanisms for modularity, encapsulation and inheritance, it allows framework design to provide a core generic infrastructure, while at the same time allows model specific elements to be “plugged-in” the base system. For example, while some framework abstractions are likely to be quite reusable (e.g. charting, and parameter settings), others tend to be more specific (e.g. agent’s internal cognitive architecture). Not surprisingly, most generic simulation frameworks are implemented using this programming paradigm.

- 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 systems. Thus, it is desirable that a model can be easily extensible and modifiable to explore alternative model assumptions and designs, and see if and in what scientifically relevant ways that changes model behavior. 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 the model without affecting the others. It also requires data gathering and visualization tools to 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 understand as possible. This increases the researcher confidence that the observed behavior and results are a genuine effect of its model assumption, rather than an artifact of some software layer that he/she did not write. This implies that simulation frameworks should be well documented, and should be possible to bypass particular services if needed to. The term transparency is used here to mean that a modeler can use (or not use) the framework’s code with the same ease as if it was programmed by she/he.

- Performance and Scalability

Typically simulations run for many time steps, many times over, with each run having potentially a large number of agents. This means that performance and scalability are a central issue. A design that may work well with a few dozen agents, might become
inadequate when the number of agents increase to the magnitude of the thousands. This requires framework implementations to be optimized for time critical operations, such as the ones involved in the most inner simulation loops (e.g. the invocation of agent’s behavior by a event scheduling mechanism). Parallelism might be part of the solution for tackling this issue, but since agent interactions occur at a very-fine grain in many models standard distribution or concurrency techniques might not be adequate (e.g. having agents in different host machines interacting trough a network, or even a concurrency scheme based on multiple threads for different agents might be plainly infeasible). On the other hand, performance is often traded off with code robustness, easy of debug, and development speed. Thus many frameworks are implemented in interpreted or semi-interpreted languages — e.g. JAVA, whose speed is now comparable to C++ in many cases.

- Portability and Ease of Use

As the prevalence of computer modeling increases, the trend is to have a similarly increasing number of researchers working in the study of the same problem domain. This is an indicator that the independent implementation of model specification, replication of results, and sharing of code between research groups might become a widespread practice. Thus, it is convenient for frameworks and model specification to be interoperable and portable across a wide range of hardware technologies and software platforms. Portable run-time systems like JAVA are specially attractive, because they have been conceived to work in virtually all currently available commercial platforms.

Another effect of the widespread use of computer simulations is that model designers will inevitably have very diverse backgrounds and experience in software engineering and computer programming. This makes convenient for frameworks to allow model designers to immerse in the details of its underlying software infrastructure at different levels of depth. At the easiest level, it should allow researchers with little or no formal training in computer programming to experiment with simple models and gain valuable intuition from them. This can be done by having an integrated graphical user interface that allows models to be designed, parameterized, run, tested, and analyzed without requiring explicit coding. More sophisticated users might then be exposed directly to the application programming interface of the framework, and use the underlying programming language tools to create more complex models that are not easily expressed in the high level interface. An alternative approach is to create a simple programming language specialized for simulations and model design. As described below, the frameworks surveyed in this chapter pursue the approach of providing direct access to programming interfaces, although some of them also provide simpler programming mechanisms.
3.3 Generic Simulation Frameworks

In this section, we describe the key abstractions and tools provided by several generic simulation frameworks for agent-based modeling. For each of the frameworks, we provide a historical context and general motivation for the framework development, describe how its main features support the broad service categories referred to in section 3.2.1, make some notes on its underlying technological infrastructure, and discuss its relative merits and limitations. We have chosen the particular frameworks described below from the larger set of available tools because they are the most similar with our own simulation framework to be presented in chapter 4, thus making comparison easier. Others include SMLD (Moss, Gaylard, Wallis, & Edmonds, 1998), CORMAS (Bousquet, Bakam, Proton, & Page, 1998), and PS-I (Lustick, 2002).

3.3.1 Swarm

SWARM was one of the first generic simulation frameworks for modeling complex systems to have a considerable impact in the research community (Minar et al., 1996). It was envisioned by the artificial life group at the Santa Fe Institute to provide a general architecture for model designers in a wide variety of disciplines ranging from physics to biology to economics.

The core of SWARM is an object-oriented framework for defining the behavior of agents and other objects that interact during a simulation. The core abstraction of the system is a Swarm — a collection of agents executing a schedule of actions. SWARM supports hierarchical model structures, whereby agents can be defined as swarms themselves composed of other agents. Thus, a Swarm is both the basic unit used for aggregation of agents, for time and concurrency management, and to structure complex agents. Additionally, a Swarm provides additional features to support memory management and probing of agent’s state.

Swarm makes very few assumptions about the type of model being implemented. In particular, it does not impose any specific representation for agents or interaction patterns. An agent can be any object in the underlying object-oriented language. Thus, the simplest case is to have agents attributes, internal state, and/or memory to be represented as data fields, and to use class methods to code for behaviors that modify their environment (and the internal state). Although libraries to support genetic algorithms and neural networks are provided, they are not tightly integrated with SWARM main abstractions. The exception to this “structure free” approach is the definition of an agent as a Swarm itself which can be used to create composite structures.

Schedule objects associated with a Swarm are used to register events that are triggered according to a time-line. Events are coded as Action objects used to specify both the object and the method to be invoked when the event triggers. Events scheduled for execution at the same time are executed according to a set schedule policy. Additionally, ActionGroup objects can also be
used to group several events together with the purpose of having fine-grain control over the relative order of their execution. Both events scheduled for the same time and event in the same ActionGroup can be set to execute sequentially and in a random order each time, sequentially according to a fixed order, or concurrently. Events specified as concurrent inform the runtime that observable model results do not depend on the order in which the sequences are executed, and therefore allows them to be executed in parallel if appropriated hardware is available. If the platform does not support concurrency at the programming level than concurrent event will be emulated by some sequential execution. Action and ActionGroup objects can be scheduled for execution once at a particular point in time, at every time period, or for a certain time duration. A Schedule is an active data structure, allowing events to be added or removed as required, thus supporting both static and dynamic scheduling.

Spatial structure is supported by a library of space objects with several topologies and representing different kinds of landscapes where agents can live. These include: 2D grids for objects or values, grids with diffusion of some real quantity between neighboring cell, double buffering of cell updates for synchronous execution of cellular like automata, etc. These space objects can be associated with a Swarm object, although this is not required to do so.

Fine-grain inspection or manipulation of a model is supported with ProbeMap objects, which are collections of VarProbe objects — used to let one look at an instance variable inside an object, and MessageProbe that allows one to call methods on an object when the modeler wants to have some extra behavior control in a simulation run.

Swarm kernel is implemented in OBJECTIVE C, both for historical and performance reasons. Namely, to have SWARM main services to run in native code. The kernel can then be linked with APIs written in several programming languages such as JAVA, OBJECTIVE C, and other.

3.3.2 Repast

REPASt is a simulation framework conceived initially to operate on top of the SWARM runtime system, with the intent of overcoming some of the usage complexities and long learning curve of Swarm and OBJECTIVE C. When a JAVA adaptor for SWARM was finally made available in the Swarm package, REPASt was turn into an independent system (Collier, 2002). REPASt borrows many of the key abstractions from SWARM, and for this reason it has been called by its developers at the Social Science Research Computing at the University of Chicago as a “Swarm-like” simulation framework.

REPASt new contributions are more in terms of the infrastructure supporting model experimentation, rather than key model design abstraction. Like Swarm it provides mechanism for creating, running, and collecting and displaying data from agent-based model. Additionally, REPASt also provides simple mechanisms for taking snapshots of running simulations and to
create *QuickTime* movies.

The mechanism used for event scheduling is very much similar to the one used in SWARM, although in Repast only one top level schedule is used. Other schedules can then be attached in a recursive way to the top level. Events in the form of actions or actions groups can be added anywhere in the tree of schedules. Subschedule objects cab be used to schedule actions several times for every time step of the parent Schedule. Scheduling can be made transparent for an inexperienced model designer, by having an object that imposes a fixed scheduling plan. The ones provided in the framework enforce the execution of an iterative sequence over a three phase time-step: preparatory phase, execution phase, and post- or cleanup phase. A BasicAction object is used to represent an atomic event, and it can be created by the modeler usually as a derived JAVA inner-class, or it can be automatically created by the Scheduler when provided with an object and method name. Such automatically created BasicAction objects have their associated class dynamically constructed and their byte-code loaded in the Java virtual machine to avoid performance penalties (when compared with the use of the Java reflection mechanisms). ActionGroup objects are also provided as in SWARM. Events can be scheduled to occur every time-step beginning at the specified time, once at some specific time, repeatedly at a specified interval (tempo), once when a simulation is paused, or at the end of the simulation. The scheduling mechanisms are used both to manage the concurrency in the agent model proper, and also in the underlying infrastructure to schedule observational actions: updating the display of viewers, recording data, etc.

**Repast** provides some support for (social) network management. Namely, graph layouts, automatic generation of commonly used topologies (e.g. small world topology, random density, square lattice, etc.), and serialization and permanent storage. Similar to SWARM and other frameworks, it also supports multiple spatial topologies as Space objects, including: 2D grids, torii, and single or multiple occupancy spaces.

Simulations in **Repast** can be controlled from a GUI, that provides buttons for starting a simulation, pausing, stopping, etc. Alternatively, a batch mode can be used to run and control the simulation by specifying the control commands in a file. Repast also includes a GUI based developer environment — **EVOLVER**, which allows simple models to be designed and analyzed rapidly from the GUI without requiring complex programming.

### 3.3.3 Ascape

**ASCAPE** is a JAVA simulation framework created to support the design, analysis and distribution of agent-based models (Brookings, 2000). It has been developed at the *Center for Social and Economic Dynamics, at Brookings Institution*, as a tool to support their research but it is also publicly available. Its principle design goals include the ability to make model design as accessible as possible, and it provides very regular abstractions that make it easy to use and
configure, both through programming or through a GUI supported designed environment.

In ASCAPE, the concepts of agent, agent aggregation, and spatial structure are collapsed in a single abstraction — a Scrape. As the authors put it, "all scapes are themselves agents" (Brookings, 2000). Additionally, scapes serve also as behavior iteration units. Events are coded as Rule objects that can be added to scapes. These rules are then iterated over all the members of a scape. The different rules in a scape can be executed serially, in which case a rule is applied to all members of the scape before moving to the next rule, or in parallel in which case all rules in a scape are executed in each member of the scape before moving to the next member. Rules can provide information about themselves to allow possible optimizations during the iteration of the rules. Namely, whether or not random order execution is required, and whether the rule can cause the deletion of scape members.

Default rules exist for common functions, such as: agent movement in a scape, fission of an agent into two agents, etc. Additionally, new rules can be created as subclasses. Because the rule set is a dynamic structure of a scape, this provides a kind of simple dynamic method dispatch capability to the framework. Comparatively with the event scheduling mechanism of SWARM and REPAST, ASCAPE rule system provides a simpler structuring mechanism to support complex static schedules. On the other hand, because rules apply to aggregates the mechanism becomes less flexible when different agents need to execute different behaviors. Moreover, it is harder to handle complex dynamic schedules. Namely, if the behavior of agents varies frequently during a simulation.

Different scape classes support different spatial and aggregation structures: variable size vectors, arrays for 1D and 2D spatial structures, arbitrary graphs, etc. Scapes hide their structure from member objects, which use a common interface to navigate and move through the scape. This mechanism provides a regular interface that promotes uniformity of use and modificability, that was absent in SWARM and REPAST.

Notwithstanding its technical qualities, the generic frameworks discussed above ASCAPE, SWARM and REPAST, are limited in the degree of mechanisms that they provide to structure models. One would like to have in addition to tools for visualization, data gathering, parameter setting, event management, and control, some that provide hints on how to structure a model and still retain some level of generality. This requires that ontological commitments be made about what type of entities will or can be present in model. Our framework presented in the next chapter follows this approach.
Chapter 4

Ethos: A Software MAS Framework for Modelling Human Social Behavior and Culture

As metaphysical, it told scientists what sort of entities the universe did and did not contain.

Thomas Kuhn, *The Structure of Scientific Revolutions*

4.1 Introduction

In chapter 2, we have layout the some of the key concepts to be taken into consideration when modeling human social behavior and culture, and in the previous chapter we have seen what kind of services several multi-agent systems provide. In this chapter, we present a new Multi-Agent System specially tailored to implement models of social behavior and culture change\(^1\). This framework, Ethos, takes the traditional approach of most other MAS a step further. While traditional frameworks provide computational abstractions that allow for the modeler to ignore implementation details of features such as data visualization and event scheduling, Ethos provides features that help to shape model structure. These include the provision of flexible behavior selection mechanisms, social influence through participation in shared activities, and management of agents' social relationships. Ethos also ensures a simple and yet flexible mechanism for event scheduling based on hierarchies of population objects, thus implementing a scheduling scheme defined by a (mostly static) order of events for each discrete time step (see section 3.2.1). All these features, taken together, permit the social science researchers to start with model design employing higher-level building-blocks than otherwise possible with

---

\(^1\)This chapter is partly based on published or submitted articles (Simão & Pereira, 2003a, 2003c, 2003b).
current MAS. This simplifies their work and speeds up the design development cycle. Moreover, students of social theory can be more easily inducted into their modelling activity by relying on abstractions made available to them from the start — and ideally consensual in the research community.

This chapter presents the Ethos framework in four parts. First, in section 4.2, we describe Ethos's main abstractions in terms of the class structure implemented in the Java language. Next, we go a bit deeper in the level of detail and present the interface of several of Ethos classes and describe the overall architecture of the framework. In section 4.4, we present several demo examples, intended to preview how the framework's features are used. Finally, in section 4.5, we compare Ethos's features with those afforded by other MAS.

4.2 Framework Overview

The Ethos framework offers as basic building blocks the kind of entities the informed modeler is likely to consider when thinking intuitively about human social behavior 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, behavior selection mechanisms, mechanisms for the social influencing of agents' mental states, defined contexts of individual action and social interaction, beside others. In the sequel we describe all such abstractions in greater detail. In figure 4.1, we depict the key abstractions of Ethos's and how they relate to each other. The text labels in figure 4.1, 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. 4.2. 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.

4.2.1 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 sites according to some neighborhood type (e.g. Moore or von Neumann), at a specified metric distance (e.g. Euclidean or Block), and contingent in 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 objects), 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 specify additional attributes, such as sex and a one-dimensional quality.

In addition to physical (site) neighbor 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 age. 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. Agent selection in relationship management, or object selection in general, is facilitated by using Selector objects. Selector objects can be employed, with a modeler specified criteria (implementations of interface Criteria), to implement a selection of some type. Selection modes can be non-competitive, where objects are selected by a local criteria, or competitive, where agents are selected by rank (e.g. using roulette-wheel, or tournament procedure (Goldberg, 1989)).

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 one-point crossover (Goldberg, 1989)). The interpretation of gene's 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' behavior). This is complementary to agents' offspring being able to inherit behavior control information from their parents.

Agents live usually 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 distribution of a simulation such as when different Space objects are run in different address spaces and different machines. Because of this, the scheduling order of events between different Space of a World is undefined.

4.2.2 Event Management and Population Structures

ETHOS uses a simple yet flexible discrete time step scheme to trigger events. Population objects are used 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 subclasses can refine the basic scheduling policy.
Population objects are also used to code population level operations on bodies. In particular, AgentPopulation is a subclass specific for Agent 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 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, with each Population object relaying control to its member elements including other Population objects. By default, a sub-population inherits the scheduling policy of its parent Population.

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

\(^2\)This corresponds to the implementation of a Composite object design pattern (Gamma et al., 1995).
Agents can select what 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 action, and it’s updated based on the outcome of actions. Action contexts are abstracted using TaskEnv objects, which correspond to opportunities for individual and collective action in some physical or social context (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 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 obtained 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 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 a Agent providing the commonly used operations and deferring the execution of these to the Control. 

Control class is abstract, and is used indirectly as the modeler instantiates one of its subclasses. At the present time 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 be randomly mutated. This implements a simple exploration mechanism, similar to evolutionary strategies (Back, 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 &

---

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

4.2.4 Other Features

Similar to most other MAS frameworks for agent-based modelling, ErHOS provides miscellaneous features in a “ready to use” Graphic User Interface (GUI). These include: the control of the simulation execution, the visualization of the simulation state, the gathering and exporting of statistics using several types of data objects such as binners and time-series, dynamic parameter setting, amongst other. In figure 4.3 we present a screen shot of the GUI.

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 showed, one representing a data plot and another a graphical view of a `GridSpace` object. In this default `GUI` object, viewers are all showed as internal frames of a desktop area.

At the top of the figure 4.3, it appears 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 number steps and 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 between which viewers are updated to reflect their underlying observed object. 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 4.3 the parameter setting panels are show on the right, and several viewer objects are shown on the left. Since these features do not differ significantly from the ones available in other MAS for agent-based modelling, we do not discuss them in greater detail.

---

4 A variation of the `viewer-observer` object design pattern is used here (Gamma et al., 1995).
4.3 Design and Implementation

Ethos design structure contains a total of six Java packages, and about 100 classes (and interfaces). In this section we describe a selected sub-set of the interfaces of these classes. First, we take a look at the details of the package implementing Ethos meta-model described in the previous section. We then present the remaining packages and describe how they relate to each other.

4.3.1 Package ethos.model

This package contains near all classes and interfaces of Ethos meta-model. Package ethos.model is conceptually separated from the rest of the framework. This allow the meta-model to be easily ported to other frameworks. We enumerate here the key classes and present the main services provided by them, mostly as public methods. In most cases, the presentation of the method signature is accompanied by a short note that should make the functionality intuitive. When this is not the case, we explain in more detail the service. This allows easier understanding of the code snippets that appear in this and in following chapters. The class relation diagram presented in the previous section should be kept in mind for easy reading. For full specification
of the package classes the reader should consult the API documentation.

- **Body** — An entity living in a space.

It is an abstract class. Implemented sub-classes are Agent and Resource.

Body objects are added to Population objects or a sub-class of this class. The act() method is called by the containing Population in every time step according to a set scheduling policy (see below). Sub-classes should override this method in order to specify model specific behavior. A position of a Body is relative to the Site it lives in. The Site topological relation with other Site objects in the containing Space defines the global position of the Body.

Below is tabulated a list of the key methods for Body:

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>int getAge()</td>
<td>Get age of Body (in number of time steps).</td>
</tr>
<tr>
<td>int getIndx()</td>
<td>Get index of Body in Population.</td>
</tr>
<tr>
<td>int getMaxAge()</td>
<td>Get maximum age of Body.</td>
</tr>
<tr>
<td>void setMaxAge(int maxAge)</td>
<td>Set maximum age of Body.</td>
</tr>
<tr>
<td>Population getPopulation()</td>
<td>Set population of Body.</td>
</tr>
<tr>
<td>void setPopulation(Population pop, int idx)</td>
<td>Set population of Body.</td>
</tr>
<tr>
<td>Point getPosition()</td>
<td>Get position of Body.</td>
</tr>
<tr>
<td>void setPosition(Point pos)</td>
<td>Set position of Body.</td>
</tr>
<tr>
<td>Site getSite()</td>
<td>Get site where Body lives.</td>
</tr>
<tr>
<td>void setSite(Site site)</td>
<td>Set site where Body lives.</td>
</tr>
<tr>
<td>Space getSpace()</td>
<td>Get space where Body lives.</td>
</tr>
<tr>
<td>void setSpace(Space space)</td>
<td>Set space where Body lives.</td>
</tr>
<tr>
<td>void kill()</td>
<td>Kill Body.</td>
</tr>
<tr>
<td></td>
<td>Removes itself from containing Population.</td>
</tr>
</tbody>
</table>

- **Agent** — An individual agent.

Normally, added to an AgentPopulation. It contains several types of methods and features which a modeler can use selectively.

Method void act(TaskEnv te) is called by an AgentPopulation, whose sub-classes have the responsibility to assign a TaskEnv to the agent. Agent maintains a list of SocialNet which contains links to other agents. SocialNet need to be added to the associated Agent by the modeler. An Agent can have set a Control object although is not required to. A Control object is used for two purposes: first, to decide which action to take given a set TaskEnv; and second, to update its internal structure based on the payoff the Agent as received. It can also update its internal structure based on payoff of other agents to model social learning. It is either a AgentPopulation sub-class or a TaskEnv sub-class that decides what kind of updates should be performed after an agents performs some action.

Below is tabulated a list of the key methods for Agent:
<table>
<thead>
<tr>
<th>Function/Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void act(TaskEnv te)</td>
<td>Act on specified TaskEnv.</td>
</tr>
<tr>
<td></td>
<td>Called by containing AgentPopulation.</td>
</tr>
<tr>
<td>void addSocialNet(SocialNet snet)</td>
<td>Add new SocialNet to Agent's social network list.</td>
</tr>
<tr>
<td>SocialNet getSocialNet(int id)</td>
<td>Get SocialNet with specified index.</td>
</tr>
<tr>
<td>void tbreakAll()</td>
<td>Break all social ties.</td>
</tr>
<tr>
<td>void kill()</td>
<td>Kill Agent. Break all social ties.</td>
</tr>
<tr>
<td>Action.getAction(TaskEnv te)</td>
<td>Get action to be performed in the TaskEnv.</td>
</tr>
<tr>
<td>Control.getControl()</td>
<td>Get and set Control of Agent.</td>
</tr>
<tr>
<td>void setControl(Control ctrl)</td>
<td>Set Control of Agent.</td>
</tr>
<tr>
<td>Action.getLastErrorAction()</td>
<td>Get last Action performed in current time step.</td>
</tr>
<tr>
<td>void setLastErrorAction(ulong a)</td>
<td>Set last Action performed in current time step.</td>
</tr>
<tr>
<td>double getBenefit()</td>
<td>Get last benefit to agent for acting on a TaskEnv.</td>
</tr>
<tr>
<td>double getCost()</td>
<td>Get last cost for acting on a TaskEnv.</td>
</tr>
<tr>
<td>double getPayoff()</td>
<td>Get last payoff (benefit - cost) to agent for acting on a TaskEnv.</td>
</tr>
<tr>
<td>void setBenefit(double benefit)</td>
<td>Set last benefit to agent for acting on a TaskEnv.</td>
</tr>
<tr>
<td>void setCost(double cost)</td>
<td>Set last cost to agent for acting on a TaskEnv.</td>
</tr>
<tr>
<td>void setPayoff(double payoff)</td>
<td>Set payoff to agent for acting on a TaskEnv.</td>
</tr>
<tr>
<td>Genome.getGenome()</td>
<td>Get Genome.</td>
</tr>
<tr>
<td>void setGenome(Genome gen)</td>
<td>Set Genome.</td>
</tr>
<tr>
<td>List.getOffspring()</td>
<td>Get list of Agent’s offspring.</td>
</tr>
<tr>
<td>Agent getParent1().Agent getParent2()</td>
<td>Get the first and second parent of the Agent.</td>
</tr>
<tr>
<td>double getQuality()</td>
<td>Get quality of Agent.</td>
</tr>
<tr>
<td>void setQuality(double quality)</td>
<td>Set quality of Agent.</td>
</tr>
<tr>
<td>int getRole()</td>
<td>Get Agent role in current or last TaskEnv.</td>
</tr>
<tr>
<td>void setRole(int role)</td>
<td>Set Agent role in current or last TaskEnv.</td>
</tr>
<tr>
<td>int getSex()</td>
<td>Get sex of Agent.</td>
</tr>
<tr>
<td>void setSex(int sex)</td>
<td>Set sex of Agent.</td>
</tr>
<tr>
<td>void addAttr(Object attr)</td>
<td>Add the specified attribute to last positions in list.</td>
</tr>
<tr>
<td>List.getAllAttr()</td>
<td>Get all attribute as a list.</td>
</tr>
<tr>
<td>Object.getAttr(int i)</td>
<td>Get specified attribute.</td>
</tr>
<tr>
<td>void setAttr(int i, Object attr)</td>
<td>Set specified attribute.</td>
</tr>
<tr>
<td>int getTimeForAction()</td>
<td>Get time for action in each simulation time step.</td>
</tr>
<tr>
<td>void setTimeForAction(int t)</td>
<td>Set time for action in each simulation time step.</td>
</tr>
<tr>
<td>boolean hasFreeTime()</td>
<td>Check if any free time is available in the current simulation time step.</td>
</tr>
<tr>
<td>void makeFreeTime()</td>
<td>Make maximum time available of action to be equal to set time.</td>
</tr>
<tr>
<td>void useTimeInAction(Action a)</td>
<td>Use some free time in performing specified action.</td>
</tr>
<tr>
<td>void useAllTime()</td>
<td>Use all free time to perform actions.</td>
</tr>
<tr>
<td>void useTime()</td>
<td>Use some free time in performing last action.</td>
</tr>
<tr>
<td>List.getAllNeighbors(double range, int dir)</td>
<td>Get list of neighbor Agents in specified range using kind of direction.</td>
</tr>
</tbody>
</table>

- **Population** — A population of Body objects.

Three parameters are invoked in the scheduling policy: the interleaving, order, and number of phases. The number of phases is the number of times control is relayed to each member Body
The interleaving can be synchronous (Clock.SCHD_SYNC) — all Body are invoked in a given phase before moving to the next phase; or asynchronous (Clock.SCHD_ASYNC) — each Body completes all phases before moving to the next one. If only one phase is set, the interleaving has a null semantics. The order can be random (Clock.SCHD_RANDOM) — each simulation time step a different order of Body is used for relaying control; or fixed (Clock.SCHD_FIXED) — the same order is used in all simulation time steps.

Below is a list of the key methods for Population:

| void act()              | Perform activities for one time step. |
| void actOne(Body bd)   | Call members according to set policy. Make the specified member act. |
| void addAllMembers(List members) | Add list of members. |
| void addMember(Body bd) | Add new member to Population. |
| List getAllMembers()   | Get members of Population. |
| void setSchedulingInter(int schdInter) | Set scheduling interleaving — synchronous or asynchronous. |
| void setSchedulingOrder(int schdOrder) | Set scheduling order — fixed or random. |
| setSchedulingPhases(int nphases) | Set number of scheduling phases. |

- **AgentPopulation** — Population for Agent objects.

AgentPopulation allows the same scheduling options as Population plus one: time usage. If time usage is activated (Clock.SCHD_FREETIME) it keeps relaying control to agents until agents have some time for performing actions — boolean hasFreeTime() return false. When all agents run out of time the AgentPopulation relinquish control for the current time step.

Below is a list of the key methods for Population:

| void setupActions(TaskEnv te) | Make all members (with any free time) select an action for the specified TaskEnv. |
| void setSchedulingTimeUsage(int schdTimeUsage) | Set schedule time usage — free time or single. |

- **TaskEnv** — A Task-Environment defining an opportunity for (individual or collective) action.

TaskEnv maintains a list of Agents that are to perform a task or interaction. It contain a context identifier which identifies the TaskEnv. This allows member Agent objects to respond differently to different TaskEnv. If an Agent uses a Control the context identifier is used as input to the decision making process.

The perform() is by convention overridden by a sub-class in order to specify that sequence of events that occur when a TaskEnv is “engaged in” by a single or collective of Agent objects.

Below is a list of the key methods for TaskEnv:

| void addAgent(Agent ag) | Add agent to TaskEnv, in last index. |
| void addAgent(Agent ag, int role) | Add agent to TaskEnv in last position, and specified role. |
| void addAllAgents(List ags) | Add list of agents to TaskEnv. |
| void clearAgents() | Clear set agents for this TaskEnv. |
| Object getContextID() | Get context ID as an object. |
| void perform() | Perform TaskEnv with set agents. |
| void useTime() | Make all set agents use time in last action (performed in this TaskEnv). |
- Control — An action selection mechanism. This is an abstract class. At present, we implement the sub-class ListControl. Control is used by agents to perform action decisions, based on the identifier of the specified TaskEnv.

Control defines three types of methods for updating: updateByPayoff(.) is used to update the Control state contingent on the associated agent last action’s payoff — usually changes the likelihood or triggering the action in the same situation; updateByDecay() is used to update the Control based of passage of time — e.g. decay of action activation; finally, updateByPriming(), updateByObservation(), and updateByCimitation(), are used to update the state contingent on other agent’s actions and/or payoffs. Different sub-class specify which kind of update modes are supported.

Below is a list of the key methods for Control:

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Action getAction(TaskEnv te)</td>
<td>Get action to be performed in the specified TaskEnv.</td>
</tr>
<tr>
<td>double getValue(TaskEnv te)</td>
<td>Get (perceived) value for specified TaskEnv.</td>
</tr>
<tr>
<td>void updateByPayoff(double payoff)</td>
<td>Update actions based on last payoff.</td>
</tr>
<tr>
<td>void updateByDecay()</td>
<td>Update actions internal structure due to passage of time.</td>
</tr>
<tr>
<td>void updateByPriming(List ags, double payoff)</td>
<td>Update actions based on (priming effects of) other agents actions.</td>
</tr>
<tr>
<td>void updateByObservation(List ags, double payoff)</td>
<td>Update actions by observation of others payoffs and actions.</td>
</tr>
<tr>
<td>void updateByCimitation(List ags, double payoff)</td>
<td>Update actions by observation of others payoffs and actions, possibly with acquisition of new actions.</td>
</tr>
<tr>
<td>void setLearningRate(double lr)</td>
<td>Set learning rate.</td>
</tr>
<tr>
<td>void setSocialLearningRate(double slr)</td>
<td>Set social learning rate.</td>
</tr>
</tbody>
</table>

- ListControl — An action selection mechanism based on a linear set of Action.

It contains a list of actions. Typically, an action is selected with a probability proportional to its match value with the TaskEnv identifier, and the value (payoff) it delivered in the past. More specifically, ListControl uses the match score obtained by invoking its action, so sub-classes of Action can define whatever match criteria they wish. ListControl uses match scores to perform a stochastic selection of an action to be performed in the desired TaskEnv. If no matching action is found for a TaskEnv one is created which has the same context identifier as the TaskEnv. This action is inserted in the list of action, and returned in Action getAction(.). If the list of action exceeds a maximum size one is selected for removal with probability inverse to its value. Finally, with a small probability actions can be mutated — usually this preserves the context identifier, but changes the response to be performed (see below).

- Action — An action in a Control with discrete actions.

An Action contain three pieces of information: a context identifier — that specifies the TaskEnv where it should be applied; the response identifier that specifies which response the action codes for; a value which is a (usually relative) estimation that specifies how much the Action has contributed to the payoff of the agent.

Below is a list of the key methods for Action:
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object getContextID()</td>
<td>Get the context ID of the Action.</td>
</tr>
<tr>
<td>double getMatch(TaskEnv te)</td>
<td>Get match level for specified TaskEnv.</td>
</tr>
<tr>
<td>Object getResponseID()</td>
<td>Get response ID.</td>
</tr>
<tr>
<td>double getValue()</td>
<td>Get overall value of Action.</td>
</tr>
<tr>
<td>void mutate()</td>
<td>Mutate action.</td>
</tr>
<tr>
<td>void updateValue(double payoff, double lr)</td>
<td>Update value of Action based on specified payoff and learning rate.</td>
</tr>
</tbody>
</table>

- **LongAction** — An Action in a Control with discrete actions with a context identifier coded as Long objects.

- **Resource** — Resource object with a real level. Like all Bodies it lives in a Site. It can be consumed by a variable amount, and renews itself by a multiplicative or additive factor every time step. A minimum level and maximum level (the capacity) can be imposed on the renewing regime.

- **SocialNet** — An Agent Social Network.

  An agent can contain a list of SocialNet. The model code is responsible for creating and adding the SocialNet to the Agent with the addSocialNet() method. Information about members in the SocialNet is maintained in Tie objects, that contain in addition to the identities of the Agent objects, the strength of the social connection, and other information (see below). Social connections are reciprocal: if an agent A is added to the list of agent B, than B is also added to the list of A. The same applies to removal with tbreak() methods.

  Below is a list of the key methods for SocialNet:

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void addTie(Tie tie)</td>
<td>Add tie to SocialNet.</td>
</tr>
<tr>
<td>List getAllMembers()</td>
<td>Get list of agents with ties with the associated Agent.</td>
</tr>
<tr>
<td>void tbreak(Agent ag2)</td>
<td>Break Tie with the specified Agent.</td>
</tr>
<tr>
<td>void tbreakAll()</td>
<td>Break Ties with all Agent in SocialNet.</td>
</tr>
</tbody>
</table>

- **Tie** — A social tie between two Agent objects.

  It contains information pertaining the social connection between agents. It includes: the identity of the agents, the time of creation, the (modifiable) strength of the social tie, and, optionally, a direction and label.

- **World** — The base class of the meta-model.

  It contains a list of Space. This object is intended to support migration between spaces, possibly distributed on the net, but this feature still needs to be implemented. World sub-classes the class Simulation from package ethos.sim defined in the next section. This allows model code to use all the services from Simulation and World easily.

- **Space** — A physical arena where a set of Body lives.

  This is an abstract class. Models use one of the sub-class: GridSpace — a 2D grid like space; ListSpace — a Space with a variable number of Site objects and unidimensional topology; and VoidSpace, a Space with a single site. A Space contains an unchangeable top population which is automatically created, and is obtained with getTopPop(). Population objects are added to this top population.

  The space topology is indirectly defined by the semantics of List getAllNeighbors(Site site, double range, int dkind). Body objects obtain the list of neighbor sites by calling this method with the site where they live as argument.
Below is a list of the key methods for **Space**:

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Population getTopPop()</td>
<td>Get top Population.</td>
</tr>
<tr>
<td>List getAllNeighbours(Site site, double range, int dkind)</td>
<td>Get list of neighbor Site objects in specified range (using specified kind of direction).</td>
</tr>
<tr>
<td>List getAllSites()</td>
<td>Get list with all Site.</td>
</tr>
<tr>
<td>Site getSite(Point pos)</td>
<td>Get Site for specified position.</td>
</tr>
</tbody>
</table>

- **Site** — A place holder in a **Space** for **Body** objects.
  It maintains a list of **Body** objects contained in a site. Members objects are accessed by class with method List getContent(Class cl).

- **Genome** — An Agent’s one or two chromosome (haploid or diploid) Genome.
  When **Agent** are created with two parent, the **Genome** of the parents is used to create the child’s **Genome**. The Genome sub-class defines how this is performed.

- **Gene** — An individual gene contained in a **Genome**.

### 4.3.2 Package ethos.util

Package ethos.util contain several utilities that are useful in the design of models. We focus here on selection mechanisms as implemented by **Selector** and **Criteria** objects. In the examples at the end of this chapter several examples will appear where they are used, this should make their pragmatic more clear.

- **Criteria** — Interface that specifies a criteria of selection. Its only method is double score(Object obj, Object ctx), that codes for the selection score of an object in the context of another object. This allows for context specific selections, a feature not provided by default by the JAVA API.

- **Selector** — Used to select and filter objects according to a criteria. It uses a roulette-wheel algorithm to select objects in competitive mode.

Below is a list of the key static methods of **Selector** (static keyword omitted):

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>List select(List pool, Criteria crit, Object ctx, List excl, int n)</td>
<td>Select objects by criteria in competitive mode (possibly with duplicates).</td>
</tr>
<tr>
<td>List selectXDups(List pool, Criteria crit, Object ctx, List excl, int n)</td>
<td>Select objects by criteria in competitive mode without duplicates.</td>
</tr>
<tr>
<td>Object selectOne(List pool, Criteria crit, Object ctx, List excl)</td>
<td>Select one object according to competitive criteria (probabilistic).</td>
</tr>
<tr>
<td>List filter(List pool, Criteria crit, Object ctx, List excl)</td>
<td>Filter objects by criteria; select object if score &gt; 0.</td>
</tr>
<tr>
<td>Object getMax(List pool, Criteria crit, Object ctx, List excl)</td>
<td>Get object with highest (best) score in pool.</td>
</tr>
<tr>
<td>Object getMin(List pool, Criteria crit, Object ctx, List excl)</td>
<td>Get index of object with lowest score in pool.</td>
</tr>
</tbody>
</table>
- **TSelector** — Used to select and filter objects according to a criteria. It uses a tournament of defined size to select in competitive mode.

### 4.3.3 Package ethos.sim

This package contains classes that establish the infrastructure for the execution and management of simulations. This includes execution control, parameter setting, exporting of data, and interconnection with a GUI. Package ethos.sim is neutral regarding GUI details, it only provides the infrastructure where a GUI can be plugged in. The package describes next ethos.gui implements a concrete GUI. Figure 4.4 provides a schematic diagram of how package ethos.sim and package ethos.gui interconnect and operates at run-time (explained below). Package ethos.sim also allow for a simulation to be run in batch mode (without a GUI).

![Diagram](image_url)

**Figure 4.4:** Package ethos.sim and ethos.gui.

- **SimRunner** — The interface to be implemented by an object that runs a simulation model.

  It is implemented by Simulation object, but model code should override the required methods. See Simulation description below for the protocol specifying the order of method invocation of SimRunner.

  Below is the complete list of the methods defined in SimRunner:
• **Simulation** — An object encapsulating information about a simulation.

The information includes: the parameters to be set in a GUI or through the command-line; logical parameter groupings to appear in a GUI; parameter walks specifying sensitivity analysis procedures in particular parameters; the viewers to appear in a GUI; data objects to be exported after a simulation execution; the associated **SimRunner**, if different than itself; the data objects file prefix and postfix.

**Simulation** object imposes the following contract with the associated **SimRunner** object: void init(), is called whenever the simulation is made active; void setup(), is called whenever the simulation is setup just before starting to run; void run(int steps, int runs), is called whenever a simulation is going to run; void walkInit(ParamWalk pw), to notify that a parameter walk (sensitivity analysis on some parameter) is to start; void walkNext(ParamWalk pw), is called at each increment of the parameter being analyzed; void walkEnd(ParamWalk pw), is called once the simulation run is completed. The void export() method is called to export data data objects, although the **Simulation** object provides a near universal and useful semantics — writing the data objects to text files. In addition to this, the **Simulation** defines the methods void runInit(), void runEnd(), that are called at the beginning and end of every simulation run. This is useful to gather statistics, such as averages across multiple runs.

**Simulation** maintains a close connection with **DesktopManager** objects. Although, **DesktopManager** controls **Simulation** objects (see below), the **DesktopManager** is created by a **Simulation** object and only if the GUI is to be started.

• **Painter** — Interface for a object that is able to paint other.

This is used to separate objects functionality from visualization inside a viewer.

• **Global** — Global variables of the framework are put here by convention.

The method getRandom() gets the global random variable so all simulation parts use a common seed. The methods echo( ) are used to display debug messages in the text console and in a graphical console if a GUI is open.

• **Observed** — The abstract super class of all observed objects.

**Observed** objects are the ones that have a **Viewer** object presenting some aspect of it in a GUI. By having a top super-class for both observed and viewer objects allows common services and policies to be implemented.

• **Viewer** — A GUI representation of an **Observed**.

• **Param** — A model parameter to be set from the command line or interactively in a GUI.

It refer to a public data field of some object. The static method createPars(obj, regexp) can be used to create a list of **Param** with references to data fields in the object obj, whose name matches the regular expression regexp.
- **ParamGroup** — A logical grouping of **Param** to be displayed collectively in a GUI.

  The method `Simulation.groupUngroupedParams(name)`, can be used to group all currently un-grouped parameter in a new group and giving it a name.

- **ParamWalk** — A node of a parameter walk structure.

  It references a **Param** and a value list to be iterated. This is used to perform automatic sensitive analysis on a parameter (or parameter set) by iterating on the specified list of value, or in `<start,end,step>` like iterations.

### 4.3.4 Package ethos.gui

Package ethos.gui implements a specific GUI. The DesktopManager objects is used as a connection between Simulation objects and GUI classes\(^5\). New GUIs (e.g. with more fancy widgets), can also be created by redefinition of the class factory GUIFactory.

- **DesktopManager** — Manager of interaction between all Desktop components and a Simulation.

  Since all GUI code is and needs to be processed in a dedicated thread, DesktopManager and Simulation in conjunction contain multi-thread management code. GUI controls delivered by DesktopManager to Simulation are interpret by having checkControlStatus() method of Simulation called at the end of every simulation time step by a World object. If the GUI controls have requested to pause a simulation, checkControlStatus() will block the simulation until the GUI gives a command to continue, or to exit. DesktopManager object also commands the Simulation object when to setup, start, stop, and export data objects.

  If Simulation code want to perform an operation in the GUI it should perform it indirectly trough the DesktopManager. For example, to report that an exception occurred while running a simulation a Simulation object invokes the reportException(Exceptio ne) on the DesktopManager to open a error dialog box. This maintains the policy that all simulation code is run by a single thread, and all GUI code is run by another thread (imposed by JAVA Swing usage protocol).

- **GUIFactory** — This is a factory class of all components of a GUI.

  By encapsulating the creation of all the objects of a GUI in a single class makes easier to replace the default GUI by a new one\(^6\).

- **Desktop** — This is the main GUI class.

  It manages and arranges in the screen all the widgets below. Below is a list of widget class implemented and arranged by Desktop:

\(^5\)This corresponds to the implementation of a *Facade* object design pattern (Gamma et al., 1995).

\(^6\)This corresponds to the implementation of a *Abstract Factory* object design pattern (Gamma et al., 1995).
<table>
<thead>
<tr>
<th>Menu Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DesktopMenuBar</td>
<td>The Menu Bar in the GUI Desktop.</td>
</tr>
<tr>
<td>ParamPanel</td>
<td>A Panel with the parameter groupings.</td>
</tr>
<tr>
<td>ParamWalkPanel</td>
<td>A Panel with the parameter walks.</td>
</tr>
<tr>
<td>ProgressPanel</td>
<td>A Panel describing the progress of a simulation.</td>
</tr>
<tr>
<td>RefreshDelayPanel</td>
<td>A Panel for manipulating viewers refresh rate.</td>
</tr>
<tr>
<td>SimCombo</td>
<td>A ComboBox to select between registered Simulation objects.</td>
</tr>
<tr>
<td>SimCtrlPanel</td>
<td>A Panel with buttons for simulation control.</td>
</tr>
<tr>
<td>ViewerFrame</td>
<td>A Frame to encapsulate a Viewer in the Desktop.</td>
</tr>
</tbody>
</table>

- **ViewerManager** — Controls when all registered viewers should be updated.

In the current implementation, it maintains a timer that triggers to indicate that all viewers should be updated to reflect the state of the underlying objects.

### 4.3.5 Package ethos.stats

Package ethos.stats defines the objects used for gathering statistics of simulations runs. The class DataObject is the base class upon which all other statistics objects derives. This allows a uniform interface to be used throughout the framework. DataExporter objects provide another top-level interface that is used to encode file format specific information (e.g., if fields are separated by spaces or commas). At moment a single sub-class, StandardDataExporter, is defined.

Because these features are similar to the ones available in other frameworks, we present below a list of the available classes and short descriptions without further discussion:

- **DataObject** — Top-level class for objects containing simulation statistics.

- **TimeSeries** — A time series of real values, with a specified start time.

- **DataVector** — A vector of arbitrary objects.

- **Histo** — A frequency histogram with a double valued x-axis, with bins of a defined size.

- **BinVector** — A vector of pairs with bins of a specified size on x-axis.

- **DataFrame** — A table like data frame with multiple columns of possibly different value types.

- **Var** — A statistical variable for sampling real value data.

- **Corr** — Linear correlation coefficient between two random variables.

- **Point** — A generic multi-dimensional point.

- **Point2D** — A two dimension Point with double coordinates.

- **Point2DInt** — A two dimension Point with integer coordinates.

- **Point3D** — A two dimension Point with double coordinates.
- VectorPoint A vectorial Point with a arbitrary number of double valued dimensions.

- DataExporter — DataObject exporting facility for some data file format.

- StandardDataExporter — Standard DataObject exporting facility.

4.3.6 Package ethos.viewer

Package ethos.viewer defines GUI Viewer objects either for data objects or for objects of the meta-model. At present we support the following viewers:

- GridSpaceViewer — A Viewer for a GridSpace. The painter of the Agent objects living in the sites is used to paint them.

- PainterFactory — Is used as a general factory for graphical Painter objects.

- Plot — A simple data plotter to plot any type of DataObject whose data elements have the appropriate format, namely, Points with at least two dimensions.

- VectorPlot — A simple vector plotter to draw Wolfram like data plots.

4.4 Demo Application Case Studies

To test the usefulness and generality of our framework, we have re-implemented several agent-based models from the literature with 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 each of these models, summarizing first the structure of the model, followed by a description of how the abstractions of our MAS framework were employed. However, to increase the reader’s acquaintance with Ethos, we first present two toy models with code snippets that exemplify how the overall architecture is used. Important to have in mind in what follows, is that our goal in presenting the different models is not to make a scientific evaluation or replication of them. Rather, the purpose is to see how the different models map on Ethos basic building blocks. In Chapter 5 and chapter 6, we present two new original models whose natural implementation in Ethos further help validate the abstractions of Ethos meta-model.
In the code below, a model top class derives from a World object and overrides a few methods to code model specific functionality. A typical situation is for a model to define a few model parameters that can be manipulated by the GUI user. In the constructor of the top class, we show how this is achieved in Ethos idiom. A regular expression is used to “catch” public member fields in specified objects by calling Param.createParams( ). These are then grouped to appear together in the GUI.

```java
import ethos.model.*; ...

public class Toy1 extends World {
    //model parameters
    public int width = 20;
    public int height = 20;
    public int nAgents = 100;
    public Toy1() {
        super("Toy\_Model\_1");
        addAllParams(Param.createParams( this , "[a-z]*" ));
        groupUngroupedParams("Main");
    }
}
```

Still in the top class, the method setup( ) should be overridden to perform all the model’s initialization code. In the code below, we start by removing all possible objects from a previous execution. Next, we create a GridSpace object that is added to your toy world as its single space, and an AgentPopulation object that is added to that space as a sub-population of the top-level population. We then define an Agent subclass — the only Body objects with your toy world. Agent objects put themselves at a random site in the world, and define their basic behavior as moving to a neighbor site at each simulation time step. Finally, a set of such agents is created and added to the created AgentPopulation.

```java
public void setup() {
    clearAll();
    addSpace(new GridSpace(width, height, true));
    AgentPopulation pop = new AgentPopulation();
    getSpace().getTopPop().addMember(pop);

class MyAgent extends Agent {
    final List allSites = getSpace().getAllSites();
    MyAgent() {
        setSite((Site) allSites.get(Global.getRandom().nextInt(allSites.size())));
    }

    public void act() {
        List ss = getSpace().getAllNeighbors(getSite(), range, dkind);
```
Still in the method `setup()`, we create two viewers. One to visualize the `GridSpace` object created, used to watch agents navigating on the GUI screen. The second viewer is utilized to visualize a non-sense data time series. The viewers are added to the toy world so they can be visualized at the GUI. The data object is added to the toy world so it can be exported to external programs.

```java
GridSpaceViewer gsp = new GridSpaceViewer(getSpace());
addViewer(new Viewer("The Space", gsp, getSpace()));
ts = new TimeSeries("ats");
addDataObject(ts);
Plot plt = new Plot(ts, Plot.LINES);
addViewer(new Viewer("A Time Series", plt, ts));
dv = new DataVector("adv");
addDataObject(dv);
}
```

Finally, in the `main()` method the model is started by creating an instance of your model class. The `start()` starts the GUI, that in turn controls the program’s main thread. The GUI then decides when to call `setup()` and when to run a simulation for a certain number of time steps, and runs (repetitions), and for which specified parameter settings.

```java
public static void main(String[] args) throws IOException {
    Toy1 model = new Toy1();
    model.start(args);
}
```

### 4.4.2 Another Toy World: A Simple Cooperation Game

In the following, we describe how to implement a simple cooperation game. In our game agents are grouped at each time step with a subset of other agents, in order to play a cooperation game and to learn from the resulting payoffs. The actual number of agents playing a game is set in a `TaskEnv` object. If it only one then the agent’s goal is to learn to select the appropriate response for the `TaskEnv`. If it is a positive number, then the focus of the agent is to decide which partners
he wants to play the game with. The partners are selected based on the strength of the social tie that the agent has with them.

Below we show the code for a subclass of SimpleTaskEnv, which in turn is a subclass of TaskEnv. SimpleTaskEnv is useful when a single (ideal) response is to be associated with a TaskEnv context. The context here is coded as a Long value. The setupPayoffs() method is used to setup the benefits and costs of agents participating in the game. Here, \( R \) is a parameter for the reward and \( I \) a parameter for the cost or investment. If the action chosen by an agent is equal to the response associated with the TaskEnv then that is considered as a cooperation act and costs \( I \). If the response differs the cost is 0. The benefits from each player action take a value proportional to the number of cooperators (Axelrod, 1985, 1997a; Boyd & Richerson, 1985).

```java
class MyTaskEnv extends SimpleTaskEnv {
    public MyTaskEnv(long c, long r, int n) {
        super(new Long(c), new Long(r), n);
    }
    public void setupPayoffs(List ags) {
        int coop = 0;
        for (Iterator i = ags.iterator(); i.hasNext(); ) {
            Agent ag = (Agent) i.next();
            LongAction a = (LongAction) ag.getLastAction();
            if (a.getResponseID() == getResponseID()) {
                coop++;
                ag.setCost(I);
            } else {
                ag.setCost(0);
            }
        }
        for (Iterator i = ags.iterator(); i.hasNext(); ) {
            Agent ag = (Agent) i.next();
            ag.setBenefit(R*coop/getNAgents());
        }
    }
}
```

The method perform() codes for the sequence of events performed in each collective action. It follows a very typical sequence: First, the actions chosen by each agent set in the TaskEnv are computed with setupActions(); Then with method setupPayoffs() the payoffs to each agent are computed and memorized; Next, it is noted that agents have used all the time available in the current time step (so they do not perform more than a single action per time step); Finally, the strength of the social ties is updated based on the payoff obtained. As discussed next, this makes agents interact more with agents they had positive experiences with — here modelled as payoffs.
Below, we define an Agent subclass for the agents in our toy model. In the constructor method, the type of control for the agent to use is set. We use a ListControl with a maximum list size set to NA. After some more parameter settings of the ListControl object, a single initial action is added to the control. This allows the ListControl to derive other actions from this first one by mutation. Next a social network is created and added to the list of the agent social networks.

```java
public void perform() {
    setupActions();
    setupPayoffs();
    updateByPayoff();
    useTime();
    updateAllTiesStrength(0);
}
}
```

In the method act() we code what each agent does. First, a set of partners is selected using a Selector object by invoking a method that performs a probabilistic selection of a certain number of agents. The class BondCriteria implements the Criteria interface, and defines the probability of an agent to be selected as a partner by another agent — which defines the context of selection. This probability is the score obtained by calling method psel() in MyAgent class. The score is 0 if the agent already performed an action in the current time step, or the strength of the bond with the other agent otherwise.

```java
class MyAgent extends Agent {

    public MyAgent() {
        setControl(new ListControl(NA));
        getControl().setLearningRate(A);
        ((ListControl) getControl()).setMutationRate(PM);
        LongAction a = new LongAction(0, 0, 0.1);
        a.setValidBits(1);
        ((ListControl)getControl()).addAction(a);
        addSocialNet(new SocialNet(this, 0));
        setQuality(Global.getRandom().nextGaussian() * 10);
    }...

    public void act(TaskEnv te) {
        List partners = Selector.selectXDups(pop.getAllMembers(),
            BondCriteria.crit, te, getThisList(), te.getNAgents() - 1);
        if (partners.size() < te.getNAgents() - 1) {
            return;
        }
        te.clearAgents();
    }
```
te.addAgent(this);
te.addAllAgents(partners);
te.perform();
}

public double psel(Agent ag) {
    if (!hasFreeTime()) {
        return 0;
    }
    return getSocialNet(0).getTieStrength(ag);
}

static class BondCriteria implements Criteria {
    static BondCriteria crit = new BondCriteria();
    public double score(Object obj, Object ctx) {
        MyAgent ag1 = (MyAgent) obj;
        MyAgent ag2 = (MyAgent) ctx;
        return ag1.psel(ag2);
    }
}

Finally, an AgentPopulation subclass is defined to assign TaskEnv to agents. Here a random TaskEnv is selected from the ones created in the constructor of the AgentPopulation subclass. This is performed in actOne() which is invoked for every agent that has free time. In the constructor the scheduling policy is also set, and the agents are created.

class MyAgentPopulation extends AgentPopulation {
    MyAgentPopulation() {
        setSchedulingTimeUsage(Clock.SCHD_FREETIME);
        for (int i = 0; i < nAgents; i++) {
            addMember(new MyAgent());
        }
        for (int i = 0; i < nTaskEnv; i++) {
            addTaskEnv(new MyTaskEnv(i, i, i + 1));
        }
    }
    protected void actOne(Agent ag) {
        TaskEnv te = getTaskEnv(Global.getRandom().nextInt(getAllTaskEnvs().size()));
        ag.act(te);
    }
}
A point that should draw the reader’s attention at this stage, is that the code for this simple cooperation is evenly distributed between the defined MyTaskEnv, MyAgent, and MyAgentPopulation subclasses. As a rule of thumb: Population subclasses should code operations (or data) relative to a large population of agents; TaskEnv subclasses should code for operations relative to a temporary assemblies of agents to perform a collective action (usually for the duration of single time step); Agent subclasses should code operations specific to a single agent. Although this may seem unnecessarily cumbersome, it allows the different model parts to be logically separated. For simple models the advantages from following this practice might not be too substantial, but for more complex models the gains in ease of design and clarity may be substantial. Moreover, specific models may not need to subclass all three classes.

4.4.3 Higgs’s Mimetic Transition

Paul G. Higgs’s mimetic transition model is an agent-based model of gene-meme co-evolution, and was designed to study the conditions under which the capacity for learning memes can evolve, even if there is no mechanism to distinguish between memes that increase biological fitness and those that decrease it instead (Higgs, 2000). The model uses non-overlapping generations of agents which have two biological parents and a set of cultural parents from which they learn memes, and do so with a probability proportional to their learning ability. Agents are also able, with some small probability, to invent new memes. Higgs’s results show that, for a wide range of parameter values, the capacity for learning evolves in a phase-transition, where the capacity for learning evolves initially very slowly, but, after a critical point, increases very fast.

Although we have several theoretical reservations about Higgs’ model, we consider useful for illustrative purposes to see how such model can be implemented in Ethos (see (Aunger, 2000) for general reservations on the use of the concept of meme in modeling human social behavior and culture). We transcribe here near in full the model specification from section 2 of Higgs’ paper:

[...] There is a fixed population size of $N$ individuals and generations are treated as non-overlapping. Each individual in the population has a biological fitness $w$ and a cultural fitness $v$. The biological fitness determines the probability of reproduction and, hence, of passing genes to the next generation. The cultural fitness determines the probability of being imitated and passing memes to the next generation. [...] It is assumed that both the biological and cultural fitness of an individual are determined by the set of memes that he or she knows. Each meme ($m$) has a biological fitness effect $w_m = 1 + s_m$, and a cultural fitness effect $v_m = 1 + c_m$. These quantities represent the effects of the meme on the fitness of the individual. For each meme the values of $s_m$ and $c_m$ are determined from normal distribution with mean zero and standard deviations of $\sigma_s$ and $\sigma_c$. [...] The biological and cultural fitness of the individual are $w = \Pi_m w_m$ and $v = \Pi_m v_m$ where the products are over the
set of memes known by the individual. A 'naive' individual knowing no memes has \( w = v = 1. \) [...] The ability to learn by imitation is determined using the simplest possible, one-locus, diploid genetic system. Variant alleles are present at this locus, each of which specifies a learning ability \( l. \) [...] Mutation occurs with a probability \( u \) per gene per generation. [...] Whilst genes are inherited from biological parents only, memes are inherited from both biological parents and cultural parents. Cultural parents are defined as non-related individuals in the parental generation from whom the children copy memes. Each new individual created in the population has two biological parents randomly chosen from the previous generation with a probability proportional to their biological fitness (i.e. roulette wheel selection) and \( K \) cultural parents randomly chosen from the previous generation with a probability proportional to their cultural fitness. The \( K + 2 \) adult models for any one child must all be different. [...] An individual with a learning ability \( l \) has a probability \( L(l) = 1 - exp(-l) \) of successfully learning a meme from any of its \( K + 2 \) parents. [...] In order to determine the set of memes known by an individual in a new generation, each of the meme of the first parent in considered in turn and acquired with probability \( L(l). \) [...] After attempting to learn memes by imitation, an individual also has a probability \( p_{\text{inv}} \) of inventing new memes. [...] 

We model this by defining an Agent subclass implementing most of the model code. Following Higgs's specification, each agent is defined to have a learning ability, and both a biological and a cultural fitness value. The constructor with two parameter specifies how an agent obtains its learning ability from two parent agents. The list of memes an agent knows about is stored as a list of attributes.

```java
class MyAgent extends Agent {
    double biofit = 0;
    double cultfit = 0;
    double l = L0;
    MyAgent() {}
    MyAgent(MyAgent par1, MyAgent par2) {
        super(par1, par2);
        double l = (par1.l + par2.l) / 2;
        if (Global.getRandom().nextDouble() <= U) {
            l = 1 + (Global.getRandom().nextGaussian() * SD_DL + DL);
        }
    }
...
}
```

In the method act(), which is called at every simulation time step by the containing Population, the list of cultural teachers for an agent is created. First, its biological parents are added to the list. Next, a set of \( K \) randomly selected members using the roulette-wheel method of the Selector object is added to the list of teachers. The method learnFromTeachers() performs a two
part nested iteration over all agents (the teachers) and all memes, to garner which memes are
learnt by the agent. This is followed by a probabilistic check to watch if the agent is able to
invent a new meme. Finally, the agent’s biological and cultural fitness is computed based on the
fitness of the set of all memes known to it. The memes proper, are trivially defined in a class
with two members — the cultural and biological contribution of the meme (not shown). The
values are set randomly according to the model’s specification.

```java
public void act() {
    if (getParent1() == null) { // first generation
        return;
    }
    Population pop = getParent1().getPopulation();
    List teachers = new ArrayList();
    teachers.add(getParent1());
    teachers.add(getParent2());
    List teachers2 = Selector.selectXDups(pop.getAllMembers(),
        CulturalFitnessCriteria.crit, null, null, K);
    teachers.addAll(teachers2);
    learnFromTeachers(teachers);
    if (Global.getRandom().nextDouble() <= PINV) {
        addAttr(new Meme());
    }
    computeFitness();
}
```

Two criteria are defined: one for agent’s cultural fitness and another for the agent’s biological
fitness. This is coded below, very similarly to the code presented in the previous cooperation

game example.

```java
static class CulturalFitnessCriteria implements Criteria {
    static CulturalFitnessCriteria crit = new CulturalFitnessCriteria();
    public double score(Object obj, Object ctx) {
        MyAgent ag = (MyAgent) obj;
        return ag.cultfit;
    }
}

static class BioFitnessCriteria implements Criteria {
    static BioFitnessCriteria crit = new BioFitnessCriteria();
    public double score(Object obj, Object ctx) {
        MyAgent ag = (MyAgent) obj;
        return ag.biofit;
    }
}
An AgentPopulation subclass is defined to perform the inter-generational step and to gather statistics. Below, we see that the BioFitnessCriteria criteria is used to select $2 \times N$ agents to be the parents of the next generation. After they are so selected they are removed from the population, and the offspring is created to replace them using the two argument constructor of MyAgent.

```java
public void act() {
    super.act();
    getStats();
    List ags = Selector.select(getAllMembers(),
            BioFitnessCriteria.crit, null, null, 2*N);
    removeAllMembers();
    for (int i = 0; i < N; i++) {
        Agent ag = new MyAgent((MyAgent) ags.get(i*2),
                (MyAgent) ags.get(i*2 + 1));
        addMember(ag);
    }
}
```

The process of gathering statistics is straightforward. Below, three statistical variables are employed to compute the mean of three different statistics. Following the Higgs paper: the mean biological fitness of the population, the mean number of memes learnt, and the mean learning ability. Once these three values are computed they are added to three TimeSeries objects, that can be exported for plotting in another program or displayed on a viewer in the ETHOS GUI.

```java
public void getStats() {
    Var vfit = new Var();
    Var vnmemes = new Var();
    Var vlabil = new Var();
    for (int i = 0; i < N; i++) {
        vfit.add(((MyAgent)getMember(i)).biofit);
        vnmemes.add(((MyAgent)getMember(i)).getAllAttrs().size());
        vlabil.add(((MyAgent)getMember(i)).1);
    }
}
```
A meme is trivially defined as a class with two members — the cultural and biological contribution of the meme. The values are set randomly according to the model specification.

```java
class Meme {
    double biofit;
    double cultfit;
    public Meme() {
        biofit = 1 + Global.getRandom().nextGaussian() * SD_S;
        cultfit = 1 + Global.getRandom().nextGaussian() * SD_C;
    }
}
```

Although Higgs’s model is a relatively simple, its implementation is far from trivial. The use of ETHOS abstraction allows the model (together with the basic JAVA language, run-time, and API) to be expressed in a natural form. By using well tested code such as Selector objects, and a pre-set domain ontology for agents and populations of agents, the solution to the implementation unfolds in a straightforward way.

4.5 Comparison with other frameworks

ETHOS currently adds three main modelling constructs to the ones provided by widely used MAS frameworks for agent-based modelling, such as SWARM (Minar et al., 1996), REPast (Collier, 2002), and ASCAPE (Brookings, 2000). Namely, behavior selection, social influence, and relationship management. It also provides a simpler and yet flexible event management scheme than these frameworks.

The behavior or action selection mechanism can be as simple or as sophisticated as the modeller desires. It can be fully implemented in the code of the agents class, or it can be implemented through control objects that implement non-trivial action selection mechanisms (Bryson, 2000b, 2000a). At this stage ETHOS provides only a simple evolutionary strategy based object (ListControl), but others can be easily added. This type of mechanisms allow a large class of models to be implemented, such as game-theoretical ones with learning (Young, 1998).
The social influence mechanism is accommodated by having control objects update their state based not only on the reference agent payoff, but also on others' payoff or other information. Although further experience is required to see which types of social influence are more suited to study which phenomena, this is clearly a key feature to support in modelling culture transmission and behavioral traditions (Heyes & Bennett G. Galef, 1996). For example, in a recent paper Noble and Franks studied the relationship between environmental structures and imitation and other simpler forms of learning (Noble & Franks, 2003). They found that imitation does not always benefit the individual, and other forms of social learning might be used instead. As far as Ethos is concerned, the motivating drive is to provide a principled manner for modelling inter-agent influence and communication, without requiring ad hoc implementations by model developers. As different mechanisms with well documented semantics are provided, this will allow models to experiment with them and see in what scientifically relevant ways that affects their simulations’ results.

Social relationships are partly supported in some of the MAS mentioned above through generic graph-like data structures. Ethos takes this a step further. Giving a semantic interpretation to agents’ social relationships, it permits the automation of some processes, such as parent-offspring relationships management, or the dynamics of social network growth. Here too, additional new features might be incorporated as we gain experience using Ethos. Such features might include analysis of social structure, and automatic generation of networks with a certain type of topology (e.g. small-world networks (Watts, 1999)).

Finally, comparing the Ethos event dispatching scheme with that of other MAS, we believe Ethos to propitiate a balanced tradeoff between simplicity of use and flexibility of operation. The event scheduling scheme of SWARM and Repast, which is based on the scheduling of events for arbitrary points in time, while flexible, is inclined to produce event driven code. As most system programmers will attest, this tends to produce code hard to understand and prone to error (Tanenbaum, 1987; Coulouris, Dollimore, & Kindberg, 1994). In contraposition to that, Ascape uses a mechanism of simple iteration of behavior rules at the grain level of agent aggregates. As illustrated by the model examples presented here this may become over restrictive. Sometimes control is required simultaneously both at the level of the individual and at the level of the aggregate (population). While this can also be implemented in Ascape by having aggregates with only one agent we find that preternatural. While Ascape provides a domain ontology that maps naturally to computational abstractions, Ethos takes the inverse approach. It maps a domain ontology which is natural for the social scientist and maps it into the computational infrastructure.

While all features mentioned above can potentially be implemented on top of these other MAS, providing them “off the shelf” simplifies modelers’ work. In any case, we plan to bridge Ethos’s core features with other MAS because some users might prefer (to continue) to use them.
Chapter 5

Human Mate Choice: Case Study I

When making your choice in life, do not neglect to live.

Samuel Johnson, in The Little Zen companion

5.1 Introduction

Perhaps the most important set of adaptations that humans and other animals are endowed with are those related to mating and reproduction (Bateson, 1983; Symons, 1979; Barkow et al., 1992; Buss, 1994). Although individuals can promote their genes indirectly by helping kin (as in social insects), a more direct strategy is to mate with a member of the opposite sex and produce offspring (Alcock, 1997). Therefore, it is expected that evolution would provide individuals in sexual species with specially designed psychological mechanisms that allow them to perform these reproductive tasks efficiently. Discerning the structure of such mechanisms is crucial if we hope to understand the behavior and minds of humans and other animals\(^1\).

Within the set of such mechanisms associated with mating and reproduction, the ones related to mate choice are central. Because individuals of the opposite sex tend to vary in their quality as suitable mates (e.g. due to genetic makeup, social status, or parental skills), and because mate quality strongly influences offspring quality, mate choice decisions are crucial to the fitness of offspring (Bateson, 1983). This is especially true when these decisions are performed in highly competitive settings, as is the case in monogamous mating systems where partner sharing is not beneficial and therefore not easily tolerated by (at least) one of the sexes. In such cases, both sexes tend to be highly choosy, leading to a process of mutual mate choice where each individual strives to get the best possible mate for itself.

It is thus not surprising that many computational and mathematical models have been proposed\(^1\) \footnote{This chapter is partly based on published articles (Simão & Todd, 2001, 2002a, 2002b, 2003).}
in the biological and social science literature that address the issue of mate choice behavior (Kalick & Hamilton, 1986; Todd & Miller, 1999; Parker, 1983; McNamara & Collins, 1990; Johnstone, 1997; Bergstrom & Real, 2000). These models allow important conclusions to be drawn about mating behavior of animals and humans, but very often they rely on assumptions that fail to hold in realistic mating environments, particularly for humans. Typical assumptions of models of mutual mate choice are that individuals search and encounter mates sequentially (usually without the ability to go back to, or “recall,” earlier mates), and that individuals make their mating choices as a single, irreversible decision whether to mate with an individual or not (Kalick & Hamilton, 1986; Todd & Miller, 1999; Parker, 1983; McNamara & Collins, 1990; Johnstone, 1997). This conflicts with the fact that humans use extensive courtship periods to establish long-term sexual/romantic relationships, and that this allows individuals to engage in relationships in tentative ways — possibly switching to better alternatives if they become available in the future (Buss, 1994; McKnight & Phillips, 1988; Weisfeld, 1999). As we show in this chapter, the existence of a non-negligible courtship time and access to potential alternative partners has, indeed, significant consequences for the strategic behavior of individuals when choosing mates.

Taking the opposite extreme from sequential choice, some earlier models make the unrealistic assumption that the complete set of potential mates in known instantaneously and is common to all members of the same sex (Bergstrom & Real, 2000). More generally, these models often assume that individuals seeking mates have complete and accurate information about the distribution of qualities of potential partners, about their own quality, and sometimes even about the preferences of other individuals (Parker, 1983; McNamara & Collins, 1990; Johnstone, 1997; Bergstrom & Real, 2000). Given that such information is typically not available in the real world, it is not surprising that most of these models are of limited empirical validity.

In this chapter, we present a conceptual framework for modeling mate choice in the context of long-term relationships with extended courtship periods, with particular emphasis on the human case. (The results can also be generalized to other animals where similar assumptions hold.) Based on an evolutionary functional analysis, we develop an agent-based model that captures key aspects of this adaptive problem. By this, we mean making appropriate proposals for the top-level goals that individuals should care about, particularly ones that contribute to biological fitness (e.g., be choosy about the quality of mating partners, and take one’s own limited reproductive lifetime into consideration), and then postulating psychologically plausible behavior rules that satisfy these goals (Gigerenzer et al., 1999; Gigerenzer & Selten, 2001). We do not assume, however, that behavior rules are fully specified by genetic instructions (Oyama, 2000). Culturally transmitted behavior rules can also arise that enhance or retard genetically shaped drives and values (Durham, 1990; Boyd & Richerson, 1985).

Our model differs from previous computational and mathematical models of mutual mate choice in that it relies on more realistic assumptions about the specifics of the human social environment and the nature of human psychological constraints. In particular, we assume that individuals
have dynamic, growing social networks of potential partners, instead of meeting these partners sequentially or having complete information about all of them instantaneously. Furthermore, our model postulates choice mechanisms (decision rules or heuristics and strategies that combine them) that are more psychologically plausible than those in previous models. We argue that individuals can make simple, efficient, and robust mating decisions by using heuristics that exploit the specifics of the adaptive problem domain rather than attempting to perform complex optimizations, thus constituting an example of ecological rationality (Todd, Fiddick, & Krauss, 2000). In addition to arguing for this new conceptual framework for understanding mating decisions, we demonstrate the empirical power of our model by showing that its predictions about overall relationship patterns observable at the population level fit data from the social sciences much better than competing models.

This chapter is organized as follows. In section 5.2, we review work in the area of computational and mathematical modeling of mutual mate choice behavior in humans and animals. We focus only on models of two-sided matching (rather than one-sided search) because they are the ones more relevant to the study of human behavior. In section 5.3, we establish a conceptual framework for the study and computational modeling of human mate choice. Capitalizing on this framework, we present in framework, we present in section 5.4 a model of (human) mate choice based on non-negligible courtship periods. Section 5.5 describes the different decision rules used by agents, and section 5.7 compares their performance. Section 5.8 presents the model predictions and makes an analysis of these predictions in the face of known empirical evidence. Finally, in section 5.9, we compare our model results with those of previous models and discuss future research directions.

5.2 Previous Work on Models of Mate Choice

Seminal studies of human mate choice using computer simulations were conducted by Kalick and Hamilton in the 1980's (Kalick & Hamilton, 1986). They started with the fact that observations of many human populations show that individuals in couples are highly correlated in attractiveness (correlations between 0.4 and 0.6 in different studies). This finding had led social scientists in the 1960's to propose the "matching hypothesis" that people actively seek a mate matched to them in attractiveness. But this seems to contradict experimental data indicating that people in general tend to prefer more physically attractive individuals as prospective partners (i.e., not taking their own attractiveness into account). To explain this apparent contradiction, Kalick and Hamilton set up individual-based simulations to study the relationship between individual-level preferences and population-level patterns. In their simulations, randomly selected individuals with particular attractiveness values are paired up sequentially in "dates." Both individuals in a date then use a probabilistic acceptance criterion to decide whether or not they accept each other, and, if both agree, they mate and leave the population. A discounting factor was introduced to make individuals less choosy with time.
Kalick and Hamilton's results demonstrated that universal preferences for high attractiveness, as opposed to preferences for similarity in attractiveness (matching), can produce realistic degrees of intra-couple correlation of attractiveness (.55). This is because higher attractiveness individuals tend to pair (and leave the mating pool) earlier than lower quality individuals, leaving the lower quality individuals with no option other than mating amongst themselves (Burley, 1983). One critique of Kalick and Hamilton's support for preferences for high attractiveness was that an unrealistically high number of dates (evaluations of members of the opposite sex) was required in the model for a realistic intra-couple attractiveness correlation to be obtained and a significant percentage of the population to mate (e.g., it took 40 "dates" for the correlation to reach .43 and 86% of the individuals to mate) (Aron, 1988).

More recently, Todd and Miller used a similar type of simulation to explore the efficacy of different individual rules for searching through a sequence of encountered potential mates (Todd & Miller, 1999). They were particularly interested in whether individuals could make reasonably good (satisficing) mate choices without having to check many potential partners. In their model, an "adolescence" (learning) period is used by individuals to adjust an aspiration level based on the feedback provided by the mating offers and rejections of potential mates they encounter. After the adolescence period, individuals make mating offers to everyone they meet who exceeds their aspiration level, and whenever both individuals in a pair make mutual offers to each other, they mate and are removed from the population. Todd and Miller's results showed that simple learning rules can adjust individual aspiration levels quickly (e.g., after an adolescence comprising 12 dates or partner-assessments) to yield mated pairs of highly matched mate value. However, these learning rules typically left an unrealistically large proportion of the population unmated (e.g., over 50%).

The animal behavior literature is rich in studies of mate choice, with the book Mate Choice (Bateson, 1983) setting the stage for the work done later in the area. In particular, the chapter by Parker (Parker, 1983) made a provisional formal analysis of (optimal) mating decisions when an infinite-horizon rate-maximization of matings is expected, with individuals alternating between searching and "processing" time. His work was further refined by McNamara and Collins with a full game-theoretical analysis of the problem (McNamara & Collins, 1990). They described a single stable strategy (a Nash equilibrium) where each sex is partitioned into a finite (or countable) sequence of categories with decreasing-quality intervals such that members of each category end up mated with members of the corresponding category in the opposite sex. In some cases, low quality members of one the sexes could end up never mating.

The problem of learning the distribution of qualities of available mates during sequential search was tackled by Mazalov and colleagues (Mazalov, Perrin, & Dombrovsky, 1996). Their model of single-sex discrimination showed how individuals could learn the mean and variance of the mate quality distribution by incremental updating with each new potential mate seen, and how this information could be used at the same time to set a varying threshold for mate acceptance. They found that such learning could be advantageous compared to using a fixed strategy if there
is enough variation in the distributions that can be encountered, and if the learning time is long enough. We also assume that such conditions hold in the mating situations we consider, and hence we include learning in our strategies; however, we avoid the full-optimization approach and the assumptions of fixed search horizons, fixed population distribution, and no search costs that Mazalov et al. adopted.

Considering more realistic constraints than the models just described, Johnstone presented a model of mutual choice where individuals have a limited time to mate (the duration of a breeding season) (Johnstone, 1997). In his model, individuals encounter each other in random pairs and must decide whether or not to mate with this one partner for the duration of the breeding season. There is also a cost associated with delaying the mating decision, and the distribution of available mates changes over time. Using a numeric method (iterative best-response), Johnstone computed optimal aspiration levels as a function of both an individual’s quality and the time left in the breeding season. His results showed that as the breeding season progresses, high quality individuals tend to become less choosy, whereas lower quality individuals initially tend to increase their level of choosiness, but after a certain period also become less choosy with time. This initial increase in choosiness of lower quality individuals arises as a way of exploiting the decrease in choosiness of high quality individuals.

The above models, while representing theoretical advances, are limited through their reliance on unrealistic assumptions such as full information or constant search costs which are unlikely in environments where the rate of encounters is not deterministic. Similar kinds of assumptions are also found in other mate choice models presented in the animal behavior literature, when either one or both sexes discriminate between partners (Real, 1990; Dombrovsky & Perrin, 1994; Johnstone, Reynolds, & Deutsch, 1996). This often has the effect of removing most of the relevant problem structure and therefore hampering the empirical validity of the models (Pepper & Smuts, 2000; Hammerstein & Riechert, 1988; Hammerstein, 2001; Simão & Todd, 2001). As Todd and Miller’s work argues, these kinds of assumptions are neither psychologically or ecologically plausible nor necessary for building useful models (Todd & Miller, 1999; Todd, 1996). More generally, it is likely that animals, including humans, use simple decision mechanisms that exploit the rich information structure present in their task-environments, rather than adhering to the normative, optimizing approach typically used in behavioral research (Reed, 1996; Gigerenzer et al., 1999).

Similar criticisms can be aimed at models of two-sided matching presented in the economics literature. In a recent paper, Bergstrom and Real review some of this work and suggest how it can provide insights for the study of animal behavior (Bergstrom & Real, 2000). Yet, they focus only on models where the set of all potential partners is defined and known before hand. This has the effect of making researchers concentrate mainly on issues of global pairing stability. Because real-world scenarios are more likely to involve dynamic social networks, global stability is only temporarily (or never) obtained (Epstein & Axtell, 1996). Therefore, emphasizing full knowledge and stability only diverts attention from other issues of greater empirical relevance.
5.3 A Framework for Modeling Human Mate Choice

In this section, we summarize some of the key aspects that should be considered when modeling human mate choice from an adaptive perspective. This is used as the rationale for design decisions in the model presented in the next section. (As mentioned earlier, these considerations also apply to other species, particularly those that have a tendency to mate in monogamous pairs and that aggregate in groups or clusters, such as some species of birds.)

5.3.1 The Nature of Preferences

A central question to be asked when studying human mating is the nature of interpersonal attraction. From an evolutionary perspective, it is expected that humans (like other animals) have traits that influence their ability to survive, reproduce, and successfully raise their offspring. They will also present some degree of variation in those features. Thus, it is reasonable to assume that evolution would endow individuals with the capability to discriminate among potential partners, preferring the ones with better traits. We can conceptualize preferences as being based on some combination of the different relevant features into an overall *mate value* or *mate quality*, as suggested by Donald Symons (Symons, 1979). Thus, as a first approximation, we can build useful models by relying only on a one-dimensional quality feature. (see section 5.9 for discussion of multidimensional qualities).

5.3.2 Courtship Processes

Much evidence exists for a universal tendency in humans to establish long-term sexual/romantic relationships. Although there is also evidence that humans like sexual variety and when possible will engage in short-term sexual relationships (Buss & Schmitt, 1993; Buss, 1994), several researchers argue that this is only a complementary strategy and not an alternative to the first pattern (Miller & Fishkin, 1997; Zeifman & Hazan, 1997). Once long-term relationships are established, both men and women substantially invest in the offspring that result from them (Symons, 1979). However, following the general pattern among mammals, human females have a much higher *minimal parental investment* than do males (Trivers, 1972). Thus, women are under selective pressure to be particularly careful to avoid choosing as mates men who would desert them after mating (Dawkins, 1976/1990). One way to be careful is to impose a costly courtship process on men, during which women can evaluate a man’s commitment and willingness to invest in the relationship (and offspring). During the courtship process, men are expected to provide women with resources and, even more importantly, spend “quality” time with them (Buss, 1994; McKnight & Phillips, 1988). Time is an especially good predictor of commitment, because although a resourceful man can give gifts to several women, he can only be physically in one place at a time.
The issue of tactical assessment of partner willingness to commit to a relationship has been widely studied in game-theoretical models presented in the literature within the category battle of the sexes — where fast or coy females (i.e., quick or slow to mate) and helpful or non-helpful males compete against each other (Dawkins, 1976/1990; Schuster & Sigmund, 1981; Mylius, 1999; Wachtmeister & Enquist, 1999). Of particular importance to mate choice research, is the result that female cogness can evolve and invade a population, provided that two condition are met: first, a combination of helpful and non-helpful males must be present in the population; and second, females must obtain increasing information about the likelihood that a male will desert her after mating as courtship progresses — so that she is trading breeding season or reproductive lifetime for information (Wachtmeister & Enquist, 1999; Wachtmeister, 2000). Both of these conditions are typically valid for humans (Weisfeld, 1999; Buss, 1994). Therefore, in this chapter we make the assumption that individuals delay mating until a courtship period is completed.

An important side-effect of the courtship period in mating processes is that it can be used strategically as an opportunity to switch to a better partner if one becomes available. This is true not only for women, but also for men, because they are also choosy in selecting partners for long-term relationships. The decision to switch can be influenced by several factors, including time and investment already made in the current relationship, how much better the alternative partner is than the current one, and how likely it is that the alternative partner will not themselves switch later to a better partner. From all these considerations, it follows that human mate choice is better modeled as a process that takes time to complete, rather than as a single atomic event. Contrary to all previously proposed models of mate choice that we are aware of, the model that we present in section 5.4 is the only one that incorporates this aspect of courtship within a background of a realistic social ecology.

5.3.3 Time Pressure to Mate

Everything else being equal, the earlier individuals mate, the better off they will be from an evolutionary point of view. This is because an individual's (reproductive) lifetime is limited, so that the earlier they mate, the more offspring they can potentially produce. Moreover, in an uncertain and risky environment, the possibility of premature death is always present. Although these arguments may hold less in modern societies, human mate search strategies were designed by evolution with these factors firmly in place (Barkow et al., 1992). All sexual animals, not just humans, have limited time to find a mate and reproduce, but many arrange their reproduction periods in a non-continuous way — usually in a form of breeding seasons, when the conditions for mating and reproduction are most suitable (Johnstone, 1997; Krebs & Davies, 1993). The model of human mate choice that we present in section 5.4 can be adapted to animals with a breeding season by equating that time period with the limited reproductive lifetime in humans.

---

This would also require the additional modification of not replacing mated individuals, so that the breeding pool shrinks over time (see section 5.4).
5.3.4 Interaction Possibilities

Despite the possibility of switching partners during a courtship process (as discussed in section 5.3.2), there are several reasons why an individual might consider delaying entering into a relationship. When engaged in an ongoing relationship, the possibilities of meeting and interacting with individuals of the opposite sex might become significantly reduced (e.g., due to mate guarding, and the requirement to invest "quality" time and other resources in current the relationship). Moreover, changing partners might have inherent costs (e.g., retaliation by the current partner). This means that individuals should be sensitive to the quality of those that they accept as tentative mates, even when they can easily switch to another mate later. This choosiness can be accomplished either by setting an aspiration level for the minimal quality acceptable for tentative mates (the typical assumption made by the models discussed in section 5.2), or by starting relationships with a low level of commitment and increasing it progressively, or a combination of both. In section 5.4 we will compare the performance of all three kinds of strategies.

5.3.5 Estimating One’s Own Quality

In addition to needing to evaluate the mate quality of others, it may also be useful for individuals to perform a (rough) estimate of their own mate value. This information can be used in deciding whether or not to initiate a courtship process (e.g., to aim at others with a similar mate value), and how much to invest in that courtship. This estimate can be based on at least two sources of information: the outcomes of past interactions with members of the same sex, and the outcomes of interactions with members of the opposite sex. The latter may provide more accurate information because it represents a direct window on the preferences of the opposite sex (Todd & Miller, 1999). In species like humans, where full maturity and maximum reproductive potential is reached after considerable time, namely, at the middle or end of adolescence in the case of females (Low, 1997), the opportunity is created for individuals to assess their mate quality (as perceived by others). In humans this takes the form of “flirting” games where individuals probe others for interest via short-term, low intimacy, contacts without engaging in a formal or socially recognized relationship (Montemayor, Adams, & Gullotta, 1994; Weisfeld, 1999). Barnacle geese, Branta leucopsis, have been reported also to engage in trial partnerships while they are young, although the adaptive reasons might not be the same as in humans (Jeugd & Blaakmeer, 2001). This trial testing can be done with little cost in terms of lost reproductive lifetime, exactly because maturity has not yet been reached during this period. In the model presented in the next section, we incorporate this aspect of human mating by introducing discrete time frames corresponding to juvenile and flirting periods where individuals assess their mate quality before engaging in a costly courtship process.
5.4  A Model of Mate Choice with Courtship

To take into account all of the important aspects of the adaptive problem just laid out, we have developed a new model of human mate choice. By incorporating a broader range of factors than previous models, we lay down here the foundations for a better understanding of sexual partnership formation in humans. We have also been able to account for a wider spread of empirical results, as we show in the next section. Our model is intended to capture scenarios where individuals only have partial knowledge of mating opportunities and are not aware of competitors. This will be the typical case in medium to large communities where the network of acquaintances is sparse.

We begin by establishing a population of constant size with $P$ males and $P \cdot R$ females (so $R$ is the sex ratio of females to males). Individuals of both sexes have a one dimensional quality parameter $q_i$, randomly generated from a normal distribution with mean $\mu$ and standard deviation $\sigma$, truncated such that $0 < Q_{\text{min}} \leq q_i \leq Q_{\text{max}}$. Time is modeled as a sequence of discrete steps. Pairs of males and females meet at a certain stochastic rate: in 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 constant model parameter specifying the maximum meeting rate, and $p_i$ is an individual-specific discount factor that is dependent on the individual's interaction capability (described below). The specific individual $j$ to be met is chosen randomly with a probability proportional to $p_j$. The meeting rate is assumed not to be dependent on the sex ratio $R$, because we do not want to conflate possible effects of partner availability and meeting rates. Thus, the only effect of $R$ is to change the relative size of the male and female pools.

Each individual maintains a list of the potential mates already met — the alternatives list. The alternatives list has a maximum size of $N$, the number of opposite-sex individuals an agent can maintain in his or her social network and possibly make courting proposals to. 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). Moreover, if the individual met is already present in an individual's alternatives list, no new individual is added to its social network. Because each meeting has the effect of putting a male and a female into each other's social network, the average number of new individuals met in each time step by each agent is therefore $2 \times Y - \epsilon$, where $\epsilon$ is a discounting factor due to repeated encounters. Overall, this models a scenario where individuals go out to meet new people of the opposite sex, sometimes successfully, sometimes not, and occasionally encountering people whom they already know.

Within the alternatives list, one member can have the “special status” of being the individual's current partner (or “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 are courting is regarded as the courtship time $\epsilon_t$. Each individual as a
specific minimum courtship time \( K_i \), which specifies how long it takes for the individual to fully commit him/herself to a relationship and become willing to mate (metaphorically, to fall in love with its partner). If both courting individuals “fall in love” in this way, then they mate, and do not consider further dating opportunities. First, we consider \( K_i \) to be constant and equal for all individuals. This simplifies the process of comparing different behavior strategies. Afterwards, in section 5.8, we drop this assumption. Every individual has a maximum reproductive lifetime of \( L \) time steps, so some individuals may be never be able to find a mate.

The interaction capability \( p_i \) of an individual is negatively correlated with the degree of involvement in the current courtship process (and therefore with \( c_i \)). This is intended to model increasing levels of intimacy and exclusivity as courtship progresses (Nowak & Vallacher, 1998). Specifically, 

\[
p_i \in [0, 1] = \max \{0, 1 - \left( \frac{c_i}{K_i} \right)^I \},
\]

where \( I > 0 \) is a model constant that defines the shape of the “intimacy curve”.

In 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, he/she has to decide whether to continue to court the same partner or try to court another individual. In the next section we specify the exact decision functions used by the agents in our model.

Note that, although an individual can make requests to court several others in each time step, he/she can only court one individual at a particular point in time. Moreover, if an individual decides to switch to another partner, the \( c_i \) of the current pair is reset to 0, meaning that if this pair ever ends up courting again in the future, they will have to restart the courtship process from scratch. Figure 5.1 depicts a graphical representation of the model structure.

If individual \( i \) requests to court individual \( j \) and individual \( j \) simultaneously requests to court individual \( i \), they will start courting (i.e. the decision must be mutual). If an individual is accepted by several others as a date in a particular time step, then he/she will court the one with the highest quality. Any abandoned (“dumped”) individual will be left with no partner (unless he/she also successfully tried to court somebody else). If no individual that \( i \) requests to court accepts him/her, and if the partner of \( i \) did not start courting someone else, then the pair remains courting and their courtship time \( c_i \) is incremented by 1. Date requests of individuals are processed sequentially so that if an individual was about to court an alternative and that alternative starts to court someone else, the first individual considers the next best alternative.

In figure 5.2, we present more formally the matching (proposal making) algorithm used in each time step in pseudo-code. For each agent we define a proposals list which is the sub-set
of individuals of the interaction list he/she decides to propose to (according to the decision function in use). This proposal list is sorted by decreasing quality of its members to allow a simple instantiation of preferences for higher quality individuals. Once all proposals lists are created (not shown in figure 5.2), the algorithm starts by marking that all agents (of both sexes) will first propose to the best alternative in the proposals list (i.e., the highest quality individual). Next, for each agent \( i \) the algorithm finds the best individual \( j \) in \( i \)'s proposal list that is proposing to him/her. If \( j \) does not correspond to \( i \)'s current proposal, \( i \) now proposes to \( j \) (thus reducing his/her ambition level). Since \( j \) might also change proposals when this procedure is applied to him/her, the algorithm is repeated until no agent changes proposals. In particular, since a proposal first made by \( i \) might have been rejected (not matched) by some \( j \) only because \( j \) was aspiring to high, this allows agents to go back to better alternatives and retry proposals. This looping is also terminated if necessary after a maximum (large) number of iterations, to avoid possible endless loops when there is no global ordering of preferences. Because in our model we are assuming that preferences are universal and perceptions are error-free, this is not strictly necessary here. In this case, the algorithm has the property that, when terminated, each agent will be paired with the highest quality agent that has reciprocated the courting proposal.

To accommodate an adolescence or flirting period in which individuals try to gather information about and assess their quality (as discussed in section 5.3.5), all individuals go through an initial phase of 4 time steps when they are introduced into the population. In this phase, they can make proposals and accept them or reject them, as described above, but they will not start a courtship process — in other words, they still remain single. The outcomes of these interactions are used solely for the purpose of allowing individuals to estimate their own quality. In the next section, we will describe a simple rule to perform that estimation in an effective way. Individuals in the adolescent flirting phase will preferentially meet other individuals in the same phase because they
for (i: all agents) {
    i proposes to best alternative in its proposals list
}

boolean nomatch = false;
int nloops = 0;
do {
    nomatch = false;
    for (i: all agents) {
        for (j: all members in proposals list of i (sorted by quality)) {
            if (j is proposing to i) {
                break; // current value of j is preserved
            }
        }
        if (i isn't proposing to j) {
            nomatch = true;
        }
    }
    i proposes to j;
}
nloops++;
} while (nomatch && nloops < MAX_LOOP);

Figure 5.2: Pseudo-code for matching / proposal making algorithm.
are the only ones receptive to courting proposals that do not involve exclusivity and increased levels of intimacy (the reverse applies to individuals who have already completed the flirting phase).

One of the things we want to explore is strategic scenarios where the adolescent flirting period of individuals does not completely coincide with the initial lifestage of low current reproductive value (see section 5.3.5). To allow such possibilities in our model, we define a juvenile period of J time steps at the beginning of an individual's lifetime where his/her current reproductive value is considered negligible. Figure 5.3 depicts graphically the possible life-histories of agents as structured by the model, and the possible relationships between time periods. In the simplest case, A is equal to J (top time line), which means that the only kind of behavior performed in the juvenile period is flirting, with real courtship (pair-formation) attempts starting immediately after. When A is smaller than J (middle time-line), individuals are allowed to start courting before the juvenile period ends to save time and try to reap the advantages of mating earlier (i.e., sooner after the juvenile period ends). The only constraint to be maintained is that mating does not occur until the juvenile period is completed. This is implemented by simply enforcing the inequality $A + K \geq J$. Finally, when A is greater than J (bottom time-line), individuals extend flirting behavior beyond the juvenile period to try to assess more accurately their own quality. In section 5.7.2, we will explore the strategic implications of using these alternative life-histories, to see whether less or more information-gathering is useful when balanced against more or less time left to reproduce once mated. As follows from the above, individuals that are unable to mate will die (and be removed from the population) when they reach age $J + L$.

5.5 Possible Strategies for Choosing and Switching Partners

To evaluate the success of decision rules and strategies, we assign mated individuals a fitness value $F(q_d, t) = q_d \times \frac{(J+L)-t}{L}$, where $q_d$ is the quality of the individual's partner (date) and $t$ is the individual's age at mating time. Thus we instantiate a payoff for high quality (rather than
quality-matching), along with time pressure to mate early (in addition to the limited lifetime). The juvenile period constant $J$ is added to the reproductive lifetime constant $L$, because we assume that postponing mating during $J$ is costless. A more realistic version of the fitness function should also incorporate a dependency on the age of the partner, in such a way that the oldest individual of the couple dictates the time available for the production of offspring (Kenrick & Keefe, 1992). To simplify matters, and because we will not address here the issue of age difference within couples, we deliberately ignore this. This has the important advantage of keeping preferences unidimensional (as discussed in section 5.3.1).

Each mate choice strategy consists of two decision rules, one used when the agent is in the single state and the other in the courting or dating state. As a notational convention, all decision rules defined below for the single state will be prefixed by $S$, and for the courting (dating) state by $D$. Strategies will be prefixed by $\Gamma$. Thus, a strategy $\Gamma_a$ is fully specified by a pair $(S_b, D_c)$, where $b$ and $c$ are identifiers (names) for the two specific decision rules in use, and $a$ is the identifier for the strategy as a whole.

In the following sections, we introduce several decision rules that are combined in different ways to create strategies. We start by introducing a naive decision rule for the single state $S_{\text{naive}}$ and a partner switching dating rule $D_{\text{swap}}$, which are combined to create a strategy named $\Gamma_{\text{swap}}$. The key feature of $\Gamma_{\text{swap}}$ is that it does not require agents to estimate their own quality to make mating decisions. Next, we present a heuristic by which individuals can estimate their own quality. This heuristic is combined with two different decision rules, $S_{\text{rational}}$ and $S_{\text{frugal}}$, that dynamically set aspiration levels specifying the minimal quality accepted in a partner. $S_{\text{frugal}}$ differs in character from $S_{\text{rational}}$, due to its greater simplicity and indirect estimation of environmental parameters. A dummy decision rule that never swaps partners, $D_{\text{null}}$, is then combined with these two decision rules to create two strategies — $\Gamma_{\text{rational}}$ and $\Gamma_{\text{frugal}}$, based purely on aspiration levels. Finally, a mixed strategy $\Gamma_{\text{mix}}$, which both uses aspiration levels and performs partner switching, is introduced. After the presentation of the strategies, we will perform a thorough evaluation of their relative performance.

### 5.5.1 Switching Partners

A naive decision rule to initiate relationships when single can be defined as follows:

$$S_{\text{naive}}(t_a) = \begin{cases} 0 & \text{if } t_a + K > J + L \\ 1 & \text{otherwise} \end{cases} \quad (5.1)$$

In the above, $t_a$ is age of the prospective alternative date. The decision rule specifies that a single individual will propose to any agent encountered as long as that agent is not too old to complete courtship.
Next, given the previously defined fitness function $F$, we can establish the following decision rule for individuals to (try to) switch partners whenever they are already involved in a courtship process:

$$D_{\text{swap}}(q_a, q_d, c_t, t, t_a, t_d) = \begin{cases} 
0 & \text{if} \quad t + K > J + L \\
0 & \text{if} \quad t_a + K > J + L \\
1 & \text{if} \quad F(q_a, t + K) > F(q_d, t + K - c_t) \\
0 & \text{otherwise}
\end{cases} \quad (5.2)$$

In the above equation, $q_a$ is quality of the alternative partner considered, $q_d$ is the quality of the current date, $c_t$ the current courtship time, and $t, t_d, t_a$ the ages of the focal individual, its date, and the alternative, respectively. The individual conditions are evaluated sequentially from top to bottom, and only when a condition does not hold will the next one be evaluated. The first and second conditions of the decision rule declare that if there is not enough time left to carry out a full courtship period (within one's own lifetime or that of the alternative), then the switch should not be attempted. The third condition declares that if the expected fitness of mating with the alternative (calculated using the total required courtship time $K$) is greater than the expected fitness of mating with the current date (calculated using the remaining courtship time $K - c_t$), then switching should be attempted. Note that because we defined $F(q, t)$ as a linear function of $t$ this implies that the third condition simplifies to $q_a > q_d \times (1 - \frac{c_t}{K})$. Finally, if none of the above conditions hold the current courtship process should continue undisturbed.

The more complete form of the third condition above would take into account the risk of the current date or the alternative abandoning the individual, but here we assume all individuals are insensitive to this risk. Specifically, fitness outcomes could be multiplied by the probability that a courtship process would be successfully completed and could also take into account the expected residual fitness (i.e., the fitness if the courtship is aborted). Making the swap decision without trying to compute all these terms goes in line with a view of agents with bounded rationality that do not bother to attempt to predict many aspects of an uncertain world and instead exploit the specifics of the problem domain (Todd et al., 2000; Pfeifer & Scheier, 1999; Gigerenzer & Selten, 2001). In this case, we are reducing a game-theoretic problem to an individual decision problem. This reduction might not be too problematic because an initial courting acceptance already implies some degree of certainty that the courtship will succeed. We do not make the claim that computing the likelihood of future rejection cannot be done, or that humans do not compute it, but rather that reasonably good mating decisions can be made without this extensive computation\(^3\).

Finally, a first strategy can be defined as $T_{\text{swap}} = (S_{\text{naive}}, D_{\text{swap}})$. As mentioned above, the distinguishing feature of this strategy is that it does not require individuals to estimate their

\(^3\)It is possible for instance to imagine a decision mechanism that will tend to accept dates of higher quality (despite the higher risk of later abandonment), unless reliable information about future rejection becomes available.
own quality. Consequently, agents using $\Gamma_{\text{swap}}$ remain idle during the juvenile period.

### 5.5.2 Flirting to Set Initial Aspiration Levels

We now turn to a second, slightly more sophisticated class of strategies. As we discussed in section 5.3.4, when there are costs involved in entering and staying in a relationship, a natural type of 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 (proposed to) as dates. Typically, this aspiration level will 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. Rationally bounded agents should not be assumed to have information about their own relative quality automatically, because their rank is relative to all other individuals in the population. Instead, we model agents who must estimate their quality dynamically and use it to perform mating decisions as they go along. As mentioned earlier, the flirting period allows agents to make a first estimation of their quality and set their aspiration levels to a corresponding (here equal) level (Todd & Miller, 1999).

Specifically, an individual $i$ starts out being totally non-discriminating by setting their self-quality estimate $q_i^*$ to 0. In each time step of the flirting period, $i$ proposes to all individuals $j$ in its interaction list that have a quality $q_j$ greater than or equal to $q_i^*$ (independent of whether or not it has proposed $j$ before, and what the outcome of such a proposal was). If a proposal is accepted by $j$ (i.e., matched by a corresponding proposal), and $q_j$ is strictly greater than the current value of $q_i^*$, then the following update rule is used:

$$q_{i_{\text{new}}}^* = q_{i_{\text{old}}}^* \cdot (1 - \alpha) + q_j \cdot \alpha \quad (5.3)$$

In the above, $\alpha$ corresponds to the learning rate (we use the value .2 in our simulations). This updating procedure can be interpreted, metaphorically, as individuals trying to climb a ladder of qualities. At the end of the flirting period, (on average) the higher the quality of individuals the higher they have climbed in terms of their $q_i^*$ self-estimate. Thus, due to the requirement of mutual acceptance, the final value for $q_i^*$ will tend to approximate the actual individual quality $q_i$, has long as a reasonable number of individuals are met during the adolescent flirting period. This final $q_i^*$ value is then used as an initial aspiration level after the flirting period\(^4\). Because of this, we will borrow the symbol $q_i^*$ and use it also to designate the aspiration level of an agent $i$ after the flirting period.

\(^4\)Similar kinds of heuristics are presented in (Todd & Miller, 1999), but these also use information from encounters that are unlikely to happen in the real world (namely, those where neither of the parties is interested in flirting or courting). Moreover, these heuristics are not combined with other update rules that regulate the drop in value of $q^*$ to compensate for wasted reproductive lifetime if partners are not found (as we do below).
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 failure to mate after the flirting period to an inflated or inadequate value of the aspiration level — which, in turn, should prompt a drop in the value of $q^*$. Below, we describe two ways to perform this (downward) update of $q^*$. The first one is based on a probabilistic estimate, which to some extent endorses the typical computation- and information-intense approach used in normative theories of decision making (Plous, 1993). The second employs a simple heuristic that dispenses altogether with such computations (Gigerenzer et al., 1999). It is worth noting that this kind of aspiration-dropping mechanism has been identified in animals other than humans. For example, female cockroaches of the species Nauphoeta cinerea have been found to have an internal biological clock that makes them less choosy as they get older (Moore & Moore, 2001).

Setting Aspiration Levels the Rational Way

A reasonable way to regulate the drop in the value of $q^*$ over time is to try predict the time $t_w$ it will take for a partner of the desired quality to be obtained, and check if it pays off to wait that period instead of proposing to court a lower quality alternative already available. This computation of $t_w$ can be done by keeping track of the (social) environment and continuously estimating relevant pieces of information, such as the rate at which individuals are met ($\hat{y}$), the fraction of individuals met that are single ($s$), and the mean and standard deviation of their quality distribution ($\mu, \sigma$). To compute $\hat{y}$ we simply divide the number of individuals met during an individual’s life so far by its age (thus assuming lifetime uniformity in the meeting rate). To compute $s$ we check the state of individuals when they are meet. To compute $\hat{\mu}$ and $\hat{\sigma}$, we assume that the set of all individuals ever met (currently in the alternatives list or not) constitutes a representative population sample. We further postulate a range of qualities that the agent considers desirable, but still attainable, as $[q^*, \hat{q}]$, where $\hat{q} = \min ([q^* + \hat{\sigma}, \hat{Q}_{max}]$ and $\hat{Q}_{max}$ is the quality of the best individual ever observed. As a result agents still prefer others with quality of at least $q^*$, but do not take into account whether or not individuals are likely to reciprocate their proposals.

If agents make the assumption that qualities are normally distributed, the proportion $f$ of individuals met whose quality falls within such an interval range can be approximated as: $f = N(\hat{q}, \mu, \sigma) - N(q^*, \mu, \sigma)$, where the function $N$ stands for the cumulative normal distribution (with the specified mean and standard deviation). Given the definition of $f$, the mean meeting rate $\hat{y}$, and the proportion of singles $s$, we can thus approximate the average value for $t_w$ as follows:
\[
\bar{t}_w = \frac{1}{\rho \cdot f \cdot \hat{y} \cdot s}
\]  
(5.4)

where \( \rho \) corresponds to a residual uncertainty factor that measures the likelihood that the found partner will accept the courtship proposal (because its aspiration level is not higher than the agent’s quality). As a simplification, we will assign \( \rho \) a constant value independent of the agent’s quality (see discussion below). We can now define an update rule for \( q^* \) as follows:

\[
q_{\text{new}}^* = \begin{cases} 
q_b & \text{if } F(q^*, t + K + \bar{t}_w) \leq F(q_b, t + K) \\
q_{\text{old}}^* & \text{otherwise}
\end{cases}
\]  
(5.5)

In the above, \( q_b \) represents the quality of the best individual in the alternatives list whose quality is lower than \( q^* \). This update rule essentially states that an individual will drop his/her aspirations to the level of the best known individual whenever waiting for a better alternative does not provide any fitness benefits. To be more rigorous, and compliant with expected utility maximization approaches, different values of \( \bar{t}_w \) and fitness gains could be averaged in the above equation for the different qualities within the interval range \([q^*, \hat{q}]\). Still, if we assume that \( \hat{q} \) and \( q^* \) differ only by a small amount (in the present case no more than \( \hat{\sigma} \)), the approximation of \( \bar{t}_w \) is reasonable, because there will be little variation between the relative frequencies \( f \) of individual quality values within this small interval range.

With this update rule, and the update rule used in the flirting period, we can now define an aspiration level-based decision rule for agents as follows:

\[
S_{\text{rational}}(q_a, q^*, t, t_a) = \begin{cases} 
0 & \text{if } t_a + K > J + L \\
1 & \text{if } q_a \geq q^* \\
0 & \text{otherwise}
\end{cases}
\]  
(5.6)

The rule essentially states that agents will propose to any individual above the sought minimal quality, provided that they are not too old. All variables have the same meaning as before. We further define a strategy \( D_{\text{null}} \) that never changes partners once courtship starts. This allows us to define a strategy based purely on dynamically computed aspiration levels, \( \Gamma_{\text{rational}} = (S_{\text{rational}}, D_{\text{null}}) \).

We could further refine the update rule for \( q^* \) by introducing additional factors in the computation of the residual uncertainty \( \rho \) (e.g., the quality of the agent making the decision, an estimation of the distribution of aspiration levels of other individuals as a function of their quality and age, the accuracy of the rule used in the flirting period to estimate \( q^* \), and others). But instead we will next pursue a different route: finding a simple but efficient update heuristic for \( q^* \) (Gigerenzer et al., 1999).
An alternative approach to regulate the drop in the value of the aspiration level $q^*$ is to keep track of the time an individual has been waiting for a partner and to lower his/her aspiration when a waiting time threshold $t_{max}$ is reached. This approach is more parsimonious than the one presented in the previous section, because estimating an appropriate value for $t_{max}$ does not require knowing environmental factors such as the meeting rate or distribution of qualities. Specifically, we will define this threshold $t_{max}$ as a fixed fraction $\kappa$ of the maximum waiting time $t'_{max}$ for which there are fitness gains by mating with an individual with quality $q^*$ rather than $q_b$ (as defined above). Intuitively, the constant $\kappa$ can be interpreted as a risk factor that specifies how much the individual is willing to bet in the attempt to court an individual with the current minimal sought quality — rather than the best (attainable) alternative that is likely to be already available to him/her. The value $t'_{max}$ can be computed straightforwardly by solving the algebraic equation: $F(q^*, t + K + t'_{max}) = F(q_b, t + K)$, where $t$ is the current age of the agent. This yields:

$$t_{max} = \kappa \times t'_{max} = \kappa \times [(J + L) - (t + K)] \times (1 - \frac{q_b}{q^*})$$  \hspace{1cm} (5.7)

The constant $\kappa$ can be interpreted, once again metaphorically, as a risk factor that specifies how much the individual is willing to bet on its current aspiration level — the minimal quality partner sought — rather than the best (attainable) alternative that is likely to be already available to him/her. Although the above expression appears complex and therefore as difficult to implement in an agent as $S_{\text{rational}}$ if we assume that evolution would endow agents with “innate” knowledge of the approximate value of the parameters $J$ and $L$, then the information-gathering demands on the agent are minimal. Specifically, an agent only needs to keep track of the highest quality individual seen so far.

We can now define a decision rule $S_{\text{frugal}}$ as equivalent to $S_{\text{rational}}$ but with the following update rule for $q^*$:

$$q^*_\text{new} = \begin{cases} q_b & \text{if } t_w > t_{max} \\ q^*_{\text{old}} & \text{otherwise} \end{cases}$$  \hspace{1cm} (5.8)

In the above, $t_w$ represents the number of time steps an individual is waiting for a date of the minimal sought quality. Whenever the value of $q^*$ is changed, $t_w$ is reset to 0. A key feature of this strategy is that although statistically speaking the time an individual will have to wait for a partner of the sought minimal quality is independent of the time he/she is already waiting, this value reflects (albeit in a highly aggregate way) many relevant aspects of the social environment. Namely, it provides a summary of the effects of the quality of the agent, the accuracy of the quality estimate, the meeting rate, the availability of partners, and the lost fitness due to wasted
reproductive time — all without making explicit observations or computations of these values.

Finally, a new strategy $\Gamma_{\text{frugal}}$ can be defined as $\Gamma_{\text{frugal}} = (S_{\text{frugal}}, D_{\text{null}})$. This is again a strategy based purely on aspiration levels, which never makes agents change partners after they start courtship.

### 5.5.3 Combining Aspiration Levels with Partner Switching

To investigate the advantages of combining partner switching strategies with aspiration level strategies, we define a new strategy $\Gamma_{\text{mix}} = (S_{\text{frugal}}, D_{\text{swap}})$. This strategy combines $D_{\text{swap}}$ with a slight variation of $S_{\text{frugal}}$ in which the update rule is modified such that outcomes of broken relationships are also used in setting the values for $q^*$: Specifically, if an agent was previously courting and the partner took the initiative of breaking the relationship, $q^*$ is updated to $\omega \cdot q_d$, where $q_d$ is the quality of the agent’s departing partner and $\omega \in [0, 1]$ is a correction factor to decrease the agent’s expectations to slightly below the quality of that partner (we will use the value .8). This 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 $q_d$. This will be the case whether the value of $q^*$ goes down or goes up — as might happen when the agent was lucky getting a high quality mate originally. Additionally, because the partner switching decision rule $D_{\text{swap}}$ gives partnerships a tentative character, it makes sense to have $q^*$ drop in value faster by decreasing the risk factor $\kappa$ in equation 5.7.

### 5.6 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 $\text{SNET\_AQ}$, 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 a integer set to be 0 or 1. Agents also set the one-dimensional quality attributed when created. This is used as primary information when other agents decide whether or not to date the agent.

Each agent maintains a reference to an 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 criteria that gives all agents an equal probability of choice (given they are not know already).
```java
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) getSocNet(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,
                getSocNet(SNET_AQ).getAllMembers());
            getSocNet(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 sub-population. It takes as arguments in the constructor the identifier of the sex and the number of agents created. In 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 agent that exceed the maximum set age.

```java
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 reason, we omitted here the details of the proposal making and matching algorithm. Full specification can be found in (Simão & Todd, 2002a).

```java
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 your mate choice model(s) using ETHOS with the one done from scratch we found some interesting tradeoff. On one side, using ETHOS required us to be aware of ETHOS meta-model abstractions, and how to properly used them. On the other hand, the existence of a conceptual landscape from which objects can be pick 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.

### 5.7 Results — Comparing the Strategies

In this section, we investigate how the strategies just specified perform over a wide range of parameter settings, and what qualitative and quantitative aspects of those strategies explain the differences in performance. More specifically, we explore the strategic role that courtship plays in mate choice behavior, looking at the advantages of switching partners during courtship. We also ask how the simplicity of decision rules for setting aspiration levels impacts their efficiency and robustness, and what the consequences are of using combined strategies that rely both on aspiration levels and partner switching.

In table 5.1, we present a summary of the model parameters with the (default) values used in the simulations and an indication of the rationale for those choices. For those model parameters where the actual value is likely to be highly contingent on the specifics of particular environmental ecologies, we present the interval ranges for which we performed sensitivity analysis. We
set 10 time steps to correspond to one 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 without 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. Note that for the results shown in the next section, all of the strategies use a courtship period of \( K = 10 \), even when no swapping is allowed (in \( \Gamma_{\text{rational}} \) and \( \Gamma_{\text{frugal}} \); this is to reflect the cost of courtship for assessing the likelihood of desertion, as discussed in section 5.3.2, even though we do not explicitly model desertion here. Furthermore, we start by assuming that the flirting period \( A \) coincides with the juvenile period \( J \), thus instantiating the first agent life-history depicted in figure 5.3 (we consider other life-histories later). Each simulation was run until 20000 agents were created, and the results shown correspond to averages across 10 runs (except when mentioned otherwise).

### 5.7.1 Strategy Comparison Across Parameter Settings

To compare the efficiency of the strategies, we use as a heuristic measure the average fitness \( F(q_d, t) = q_d \times \frac{(J + L - 1)}{L} \) of individuals with quality equal to or above average \( (q_i \geq \mu) \). Figure 5.4a shows these measures for the four strategies defined in section 5.5: \( \Gamma_{\text{swap}} \), \( \Gamma_{\text{rational}} \) (with uncertainty factor \( \rho = .5 \)), \( \Gamma_{\text{frugal}} \) (with risk factor \( \kappa = .3 \)), and \( \Gamma_{\text{mix}} \) (with a lower risk factor \( \kappa = .1 \) to take into consideration the possibility of switching partners as discussed in section 5.5.3), with the parameter settings otherwise as specified in the previous section. The rationale for this measure is that more efficient strategies should move the fitness of individuals of high quality further above chance level (i.e., the level obtained by mating with the first individual encountered) than less efficient ones that select lower quality individuals. With our current parameter settings random mating would give all individuals a fitness value of about 8.
Figure 5.4: Fitness comparisons between the mate choice strategies.  a) Average fitness of individuals with quality above average using each strategy.  b) Average fitness of individuals using each strategy, as a function of their mate quality.

In fact, as can be seen from figure 5.4b, the small gains in the fitness of high-quality individuals above the chance level is achieved in all strategies at the expense of a much greater reduction in the fitness of low quality individuals.  Furthermore, since higher quality individuals have higher fitness they will deliver more replicas (offspring) to the next generation.  Thus, the strategies that perform better for these individuals are the ones most likely to invade a population over evolutionary time.  While a complete analysis of strategy performance and evolutionary stability would require a full-fledged game-theoretic analysis, because the fitness of individuals is contingent on the relative frequency of the different strategies existing in the population (Smith, 1982; Dugatkin & Reeve, 1998); here we focus on the above heuristic measurement as a considerable simplification, but still useful first step.

Figure 5.4a shows several important results.  First, $\Gamma_{\text{swap}}$ performs slightly better than $\Gamma_{\text{rational}}$.  This is interesting because with $\Gamma_{\text{swap}}$ individuals do not try to estimate their own quality, but instead only try to retain partners and switch to better ones.  Furthermore, $\Gamma_{\text{swap}}$ performs efficiently in spite of a relatively small courtship time, along with a realistic reduction in interaction possibilities as courtship progresses.

To analyze the extent to which this result holds across parameter values, we present in figures 5.5a and 5.5b how the performance of $\Gamma_{\text{swap}}$ varies as a function of meeting rate $Y$ and size of alternatives list $N$ (keeping the courtship period $K$ constant and equal to 10).  As can be seen when $N = 5$ (figure 5.5a), $\Gamma_{\text{swap}}$ performs almost at the same level as $\Gamma_{\text{frugal}}$ (and better than $\Gamma_{\text{rational}}$), for all values of $Y$ presented — and with slightly increasing performance as $Y$ increases.  On the other hand, with $Y = .5$ (figure 5.5b), $\Gamma_{\text{swap}}$ requires a minimal value of $N = 5$ to reach a performance close to $\Gamma_{\text{frugal}}$ — with no important changes in performance after $N$ reaches the value 6.  Thus, $\Gamma_{\text{swap}}$ can work as an effective mating strategy, but at the expense of keeping track of (at least) a small number of alternatives.  If agents could strategically choose which individuals to keep track of (e.g., only the ones with higher quality than the current
Figure 5.5: Average fitness for individuals with above-average quality \((q > \mu)\) using the four strategies. a) Comparing performance across changes in meeting rate \(Y\) (with \(N = 5\)). b) Comparing performance across changes in maximum number of simultaneous alternatives \(N\) (with \(Y = .5\)).

partner), then this memory requirement could be reduced. Elsewhere we have shown that \(\Gamma_{\text{swap}}\) also performs rather well, in terms of producing mated pairs with high interpair quality correlation (see beginning of section 5.2), for most values of \(Y\), with very little gain from increasing courtship time beyond a small value (Simão & Todd, 2001). This indicates that \(\Gamma_{\text{swap}}\) allows individual individuals to make good mating decisions with reasonably few encounters. Intuitively, the power of \(\Gamma_{\text{swap}}\) arises from the fact that holding partners shields individuals against the uncertainty of whether or not they will be able to find better (and attainable) alternatives soon enough in the future.

Moving away, temporarily, from our focus on humans, we can see the relevance of these results on the effectiveness of \(\Gamma_{\text{swap}}\) for other animal species. First, the life-history of many species is such that their juvenile period cannot be easily used by individuals to evaluate their own quality accurately without losing important reproductive time. This will be the case especially if an individual’s absolute and relative quality varies between breeding periods — as can occur for seasonal birds (Johnstone, 1997; Alcock, 1997; Krebs & Davies, 1993). Second, the social and ecological constraints might be such that the relevant inter-sex interactions are too rare to allow a good estimation of one’s own quality (although some animals may also rely on internal “gauges” such as health condition). Thus, aspiration-level-based strategies may not make sense for many species, but \(\Gamma_{\text{swap}}\) is a good alternative.

Wittenberger (Wittenberger, 1983) proposed a related mating tactic available to many animals, called the **sequential-comparison tactic**: search for mates only until there is a drop in the quality of the next individual found, and then attempt to go back to the previous individual. If we think of previously visited individuals as tentative dates, then this strategy has some parallels with \(\Gamma_{\text{swap}}\) (when \(N = 2\)). The main difference is that the trigger for mating in the sequential-comparison tactic is finding a lower quality individual, while in \(\Gamma_{\text{swap}}\) a fixed courtship time is
used. Because both strategies present drawbacks that the other can address — namely, \( \Gamma_{\text{swap}} \) requires several individuals to be met or be available during the courtship period to perform well, and the sequential-comparison strategy is prone to prolonged search times if the meeting rate is very low and might lead to bad decisions due to precocious mating — combinations of the two kinds of strategies can be envisioned (e.g., use a threshold time to find the next alternative, and use the quality of the alternatives found as a trigger for mating). Empirical studies have indicated that some species use even more elaborate strategies, involving the sampling of increasingly restricted subsets of individuals before a choice is made (Patricelli & Borgia, 2001).

Another interesting aspect to be observed from figures 5.4 and 5.5 is that \( \Gamma_{\text{frugal}} \) performs better than \( \Gamma_{\text{rational}} \) for many parameter values. In figure 5.5b we can see that, when \( Y = .5 \), this holds when \( N < 8 \). Above \( N = 8 \) \( \Gamma_{\text{rational}} \) gains an advantage because the update rule it uses potentially changes the aspiration level \( q^* \) in every time step, and so can profit by seeing a larger and more representative pool of alternatives. In contrast to this, \( \Gamma_{\text{frugal}} \) only updates \( q^* \) after a certain waiting time has elapsed, and so is virtually unaffected by changes in \( N \).

In a similar vein, figure 5.5a that shows that \( \Gamma_{\text{frugal}} \) is very robust to changes in the value of \( Y \), while \( \Gamma_{\text{rational}} \) exhibits a steep decrease in performance for values of \( Y \) greater than .5. This is somewhat surprising given the fact that the strategy directly estimates this meeting rate itself (see section 5.5.2). But because the increased meeting rate makes all individuals more picky, waiting longer before lowering their aspiration levels (see equation 5.4), this reduces the likelihood that the alternatives an agent meets that match or surpass its aspiration level will accept the agent in turn. It turns out that assigning a constant value to the uncertainty parameter \( \rho \) in equation 5.4 is insufficient to implement a good strategy across the range of values for \( Y \). The value \( \rho = .5 \) chosen at the beginning of this section provides the best performance for \( Y = .5 \), but for higher values of \( Y \) we found that smaller values of \( \rho \) deliver better performance. On the other hand, and for small values of \( Y \) higher values of \( \rho \) perform better. This shows that \( \Gamma_{\text{rational}} \) is very sensitive to proper settings of \( \rho \).

A tentative conclusion to derive from these results is that \( \Gamma_{\text{rational}} \), by trying to guess what the future might bring in terms of partnerships, becomes very sensitive to variations or errors in the estimation of the wide range of factors involved in this calculation. As mentioned, this could be remedied by taking into account more factors in the computation of the parameter \( \rho \) in equation 5.4, in a sense creating a more complex (implicit) model the world. This would require, though, that agents collect even more information (unless a fixed value of \( \rho \) is adequate for all parameter values, or environmental settings, the agent might be exposed to). Not only would this be temporally and computationally expensive, but it would also be highly prone to overfitting (Gigerenzer et al., 1999). \( \Gamma_{\text{frugal}} \), on the other hand, by exploiting the information available in a single good piece of evidence — the time the agent is waiting for a partner — becomes robust to possible environmental fluctuations.

A further key aspect to draw from figure 5.4a is that \( \Gamma_{\text{mix}} \), by combining the advantages of
\( S_{\text{frugal}} \) in setting minimal aspiration levels with the ability to swap partners during courtship as directed by \( D_{\text{swap}} \), is able to outperform the individual use of these rules in the two strategies \( \Gamma_{\text{frugal}} \) and \( \Gamma_{\text{swap}} \). From figures 5.5a and 5.5b, we can see that this result is robust across changes in \( Y \) and \( N \). Intuitively, \( \Gamma_{\text{mix}} \) performs better because setting aspiration levels allows low-quality individuals to be avoided, while swapping allows the agent to not be too picky about the quality of the first accepted partner because of the possibility of later switching to a better partner. This result suggests that delaying mating with courtship — often interpreted in formal models of mating behavior solely as a signal of male commitment (Dawkins, 1976/1990; Schuster & Sigmund, 1981; Mylius, 1999; Wachtmeister & Enquist, 1999) — can also be used effectively (by both sexes) for mate selection. Moreover, given the dual composition of \( \Gamma_{\text{mix}} \) and its good performance, it might be argued that (at least for complex social species like humans) mate choice behavior and its underlying psychological mechanisms could be composed of not just one strategic component, but several elements which are activated in different periods or contexts in the individual’s life. Together, the combination of these elements is orchestrated to produce effective life-span mating behavior. Although in this chapter we explore only strategic behavior that lasts until the time of first mating, we are currently working on further models of mate choice that extend the time-frame under consideration (e.g., including rules to decide if a partner should be deserted once offspring have been produced).

5.7.2 Testing the Utility of Courtship with Alternative Life-histories

In the previous section, we found that being able to switch partners during a courtship period is superior to courtship without partner-switching (that is, \( \Gamma_{\text{mix}} \) outperformed \( \Gamma_{\text{frugal}} \) and \( \Gamma_{\text{rational}} \)). This was not so surprising, as being able to swap upwards in mate quality should clearly be beneficial. But that result did indicate another adaptive role for courtship — holding on to good potential mates, at least until a better one is found — beyond the usual proposal of courtship for testing male fidelity (see section 5.3.2). Here we want to test this role for courtship more thoroughly, asking whether the ability to hold and switch partners can actually make using a courtship period better than not using any courtship period. That is, we will compare the mixed strategy \( \Gamma_{\text{mix}} \) with a courtship period including partner switching, against a variation of the \( \Gamma_{\text{frugal}} \) strategy that needs no courtship at all and just allows individuals to mate immediately after the juvenile period. Can delaying mating via courtship possibly outperform immediate mating, given the time cost paid by the former? If so, then we can infer that courtship can have an important strategic role in mate choice through the possibility to hold good partners and switch to better ones, in addition to its usefulness in assessing mate “honesty” (intention of not deserting after mating).

To test this question, we introduce two strategy variants that draw upon the different life-histories presented in figure 5.3. First, we define the strategy \( \Gamma_{\text{frugal}}^- \) to be equivalent to \( \Gamma_{\text{frugal}} \) (\( \kappa = .3 \)), but with the courtship period removed by setting \( K = 0 \) (thus corresponding to a degenerate case of the first life-history presented in figure 5.3). Second, \( \Gamma_{\text{mix}}^- \) is defined as
equivalent to $\Gamma_{\text{mix}} \ (\kappa = .1)$, with the exception that it makes use of a reduced flirting period 
$A = J - K(= 20)$ so that courtship can begin while individuals are not yet fully mature (thus 
(corresponding to the second life-history presented in figure 5.3). This means that $\Gamma_{\text{mix}}$ does 
not pay as big a cost in terms of mating delay as would $\Gamma_{\text{mix}}$ (otherwise there are no benefits 
of delaying mating and using the courtship period to switch partners — compare the plots for 
$\Gamma_{\text{mix}}$ in figure 5.5 where the fitness values are for most parameter setting below 8.4 with those 
in figure 5.6 for $\Gamma_{\text{frugal}}^-$ where the values are higher than this.) With these two strategies, we 
can compare mate choice without courtship (i.e., beginning immediately following the juvenile 
period) using $\Gamma_{\text{frugal}}^-$ with mate choice including courtship and partner-switching using $\Gamma_{\text{mix}}^-$.

Figures 5.6a and b depict the performance of the strategy variants $\Gamma_{\text{mix}}^-$ and $\Gamma_{\text{frugal}}^-$ (along 
with $\Gamma_{\text{frugal}}^+$, which is described below), as a function of $Y$ (with $N = 5$) and $N$ (with $Y = .5$) 
as in figure 5.5. As can be seen, for all values of $Y \geq .2$ and $N \geq 3$, $\Gamma_{\text{mix}}^-$ always performs 
better than $\Gamma_{\text{frugal}}^-$. Only for very low meeting rates does the $\Gamma_{\text{mix}}^-$ strategy of courting and 
partner-switching not prove beneficial. This is an important result, because it indicates that 
even in the ideal case where no courtship is required to evaluate the commitment of potential 
mates (i.e., here with $K = 0$), the costs of courtship in terms of delayed mating can still be 
outweighed by its benefits in finding and switching to better partners. In other words, courtship 
can serve not only the function of selecting for honest mates, but also selecting for good mates. 
Thus, models of the battle of the sexes (see section 5.3.2) and models of sequential mate choice 
can profit from integration into more complex and realistic theories of reproductive behavior.

Finally, we can ask whether simply extending the adolescent flirting period ($A$) of $\Gamma_{\text{frugal}}^-$ could 
make this courtship-less strategy outperform the $\Gamma_{\text{mix}}^-$ strategy with courtship. Will the oppor-
tunity to learn longer and set a better-informed aspiration level be as effective as the opportunity 
to hold onto and switch partners during courtship? To find out, we created the strategy $\Gamma_{\text{frugal}}^+$, 
defined to be equivalent to $\Gamma_{\text{frugal}}^-$ but with an extended flirting period $A = 40$ (and still with 
$K = 0$). This strategy corresponds to a degenerate case of the third life-history in figure 5.3. 
By comparing the performance curves of $\Gamma_{\text{frugal}}^+$ and $\Gamma_{\text{frugal}}^-$ in figure 5.6 with those of $\Gamma_{\text{frugal}}^-$ 
in figure 5.5, we see that extending the flirting period beyond the juvenile period (using time 
otherwise used for courtship) is not particularly advantageous. This is because only a small 
number of encounters are required for reasonable setup of the aspiration level, and therefore 
$\Gamma_{\text{frugal}}^+$ delays mating unnecessarily. Note that this is in contrast to the findings of Mazalov et 
al. that longer learning leads to better mate choice (Mazalov et al., 1996). This result indicates 
that partner switching during courtship plays a qualitatively different role from just obtaining 
better estimates of one's own quality during an equivalent time period.
Figure 5.6: Average fitness for individuals with above-average quality \( q > \mu \) using four variant strategies. a) Comparing performance across changes in meeting rate \( Y \) (with \( N = 5 \)). b) Comparing performance across changes in maximum number of simultaneous alternatives \( N \) (with \( Y = .5 \)).

5.8 Testing Model Predictions

In the model analysis so far we considered individual’s courtship time \( K_i \) to be constant across individuals and life span. This was useful for the purpose of comparing strategies. In this section, we are interested in investigating model predictions. Namely, the kind of global patterns that emerge from the individual decision rules and the social interaction model and see how well they account for the self-organization of real human mating populations. Because the abrupt transition made from a flirting period to a “true” mate search period is unnatural (given what is known about the social ecology of human mating (Montemayor et al., 1994; Weisfeld, 1999)), in this section we drop that assumption. That is, we set the juvenile period parameter \( J = 0 \), so that agents can mate whenever the courtship time is completed, and we assume that agents can update their aspiration level continuously. This is as if the flirting period parameter \( A = L \), but when dating agents make proposals only if the used strategy specifies to do so. Due to its high performance, robustness, and plausible psychological assumptions, we will take the strategy \( \Gamma_{\text{mix}} \) as a reasonable candidate to model human behavior (at the level of abstraction of interest to us). In the rest of this section, we use \( \Gamma_{\text{mix}} \) with the same base parameter settings as specified at the beginning of section 5.7.

Because individual fertility is lower at very young ages and reproductive lifetime is finite, the costs of delaying mating is higher later in life. Thus, 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 again the constant model parameter defining \( K_i \) at age 0.

We set the male population size parameter \( P = 100 \), and the female to male sex ratio \( R = 1 \) (initially), 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 as a tenth of a 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 do not have 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.

5.8.1 Patterns of correlation in quality

Figure 5.7 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).

Additionally, 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. 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, 2002a)), the percentage of mated individuals drops slightly, but it's 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, 2002a), 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 mates 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 behavioral 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 et al., 1999).

5.8.2 Relationship Stability

Our simulations also gave some indication that the average duration of terminated relationships is negatively correlated with the difference in quality between courting partners, as one might expect. To more clearly highlight this trend, we run a simulation with a much higher number of runs — 50. In figure 5.8a, we show the results for a particular setting of $Y(= .5)$ and $K(= 10)$. We can see that the higher the difference is in the qualities within a couple, the more unstable (less durable) the relationship is. Moreover, if individuals differ too much in quality they will never court. (The absolute quality of individuals taken alone is a less important predictor of the duration of relationships.) Similar findings were obtained in a number of empirical studies (see (Kalick & Hamilton, 1986) for a short review).

On a more theoretical level, this result matches insights from well-established (but non-formal) social psychology theories about factors underlying satisfaction and stability in relationships, such as the theory of interdependence in close relationships and its extension, the *Investment*
Figure 5.8: a) Average duration of relationships; b) Average number of breaks \( K = 10; Y = .5 \).

Model (Berscheid & Reis, 1998; Rusbult, Martz, & Agnew, 1998; Rusbult & Buunk, 1993). In this theory, three factors are identified as associated with commitment to a relationship, and consequently its stability: satisfaction, quality of alternatives, and investment. Subjective satisfaction and investment are known to be positively correlated with relationship stability, while quality of alternatives is negatively correlated. Although in Rusbult’s Investment Model these three factors have a broad holistic interpretation and they have not been interpreted as aspects of individual adaptive strategies for finding good mates, our model suggests that such an interpretation is reasonable. As depicted in figure 5.8a, the relative quality of alternatives (compare to the current partner) is the most important variable in controlling partner switching behavior — and therefore the termination of relationships. Furthermore, if we equate courtship time in our model with investment, then we can also functionally explain the positive correlation between an individual’s investment in a relationship and their commitment to maintain that relationship. Regarding the degree of satisfaction in relationships, it is empirically known that intensity of romantic feelings is higher when an individual perceives his/her partner as more attractive (Bunk, 1996; McKnight & Phillips, 1988). This is consistent with an evolutionary functional interpretation, such as the one we endorse here, where individuals seek and have a preference for high quality partners.

We also found that in our model, as expected, lower quality individuals are more likely to be “dumped” by their partners, with higher quality individuals taking the initiative of breaking relationships (figure 5.8b). Furthermore, lower quality individuals on average need more courtships and more time to find a mate. This is because these individuals are more likely to be courting somebody with a higher quality, who will often take the initiative of breaking off the relationship.
Figure 5.9: Mean mating time as a function of individuals quality (with $Y = .1$), for a fixed value of courtship
time $K = 40$.

5.8.3 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). 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 5.9).
This occurs because high-quality individuals mate assortatively with similarly high-quality individuals,
who are few 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. 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 5.10).
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 sufficient to generate the empirically-observed
age-at-marriage curve.

To match the demographic data through their individual-level mate search model, Todd and

$^5$In the analyses that follow, we consider demographically-observable marriage as a stand-in for long-term
mating behavior.

$^6$When we made the quality distribution uniform ($q \in [Q_{\text{min}}, Q_{\text{max}}]$), 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.
Figure 5.10: 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.

Figure 5.11: Mean mating time as a function of the (females to males) sex ratio $R$ on logarithmic scale (with $Y = .1$ and $K = 40$).

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 of 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 5.10). This form of individual differences is reasonable to build into the model, because variations in socio-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.
5.8.4 Effects of skewed sex ratios

An additional factor that can affect the timing of marriage (or mating) is the female-to-male sex ratio in a particular population. Social scientists have proposed two opposing hypotheses concerning the effect of sex ratio on marriage, motivated on both empirical and theoretical grounds, and so we wanted to test whether our model would lend further theoretical support to one or the other. Marital search theory (Lloyd & South, 1996; Oppenheimer, 1988) proposes that an excess of members of one sex should accelerate the transition to first marriage by members of the opposite sex (e.g., an excess of females should make males marry earlier), because of the increased opportunities to find a suitable mate. Conversely, among the more common sex, the mean age at marriage may go up, because some will marry the rarer sex early but others will marry late or never. Imbalanced sex ratio theory (Guttentag & Secord, 1983), in contrast, argues that an excess in the relative number of women should reduce men’s motivation to commit to marriage, and so lead to later marriage (for both men and women). (When men are more common than women, this theory makes roughly the same prediction as marital search theory, that those men who marry at all will marry earlier.) While our model cannot address the motivational mechanism proposed in imbalanced sex ratio theory (because we assume constant motivation to mate monogamously), we can assess whether individual search in the presence of an imbalanced sex ratio would lead to earlier or later marriage for those who marry.

To implement an imbalanced sex ratio in our model, we vary the relative number of males and females in the population by manipulating the model parameter $R$, as specified in the beginning of section 5.4. An alternative or additional arrangement could be to change the size of the social networks of each sex (e.g., make men meet and remember more women, or vice versa). However, in earlier work (Simão & Todd, 2002a) we found that once the social network reached a reasonable size (around 8 acquaintances), there was little change in search performance, so this means of introducing imbalanced sex ratios into the model would have little impact.

We show the effect of sex ratio on mean mating time of those who find a mate in Figure 5.11, where we the sex ratio ($R$) on the x-axis on logarithmic scale to highlight the symmetry of the results for imbalanced sex ratios above and below 1. The results are averaged across 10 runs, each with $P$ males and $R \times P$ females. Error bars represent standard deviation of the mean mating times from each run. (This is different from Figure 5.9 where the error bars represent the standard deviation of mating time for all individuals in all runs). Individuals who are unable to mate are left out of the calculation. We vary the sex ratio parameter $R$ over a very wide range (beyond what is typically observed in natural populations, though historically there has been such variation — see (Guttentag & Secord, 1983)) to more clearly observe the general trend. The male population size parameter $P$ is kept fixed at 100, as before, so that when $R = .8$ for example there are $.8 \times P = 80$ females. The clear result in Figure 5.11 is that the mean mating time decreases when the sex ratio is imbalanced (away from $R = 1.0$). When $R < 1$, the deficit of females means that some males remain without mates. This has the effect of actually reducing the mean mating time of those who mate, because those males who mate are of higher
quality on average (as shown in Figure 5.12), and they typically find mates earlier (as shown in Figure 5.9). Thus, by removing the very low quality males who are now unable to win a mate from the marriage pool, the average marriage time is shifted downwards. In addition, because the mean quality of mated males increases, females are less likely to break relationships, which also leads to less delay before mating. These same factors (but from the female perspective) lead to decreased age at mating when \( R > 1.0 \).

Thus, while the overall strength of the effect of changing sex ratios on mean mating time is relatively small (compare the mean mating time differences of up to 8 in Figure 5.11 with the mating time range from 40 to 120 in Figure 5.9), the consistent decrease in mating time is in line with the predictions of marital search theory and not with those of imbalanced sex ratio theory. It is understandable that results stemming from a model based on individual search like ours would match the predictions of a search-based theory; but it is also important to point out that the simplicity of our model prevents us from being too conclusive about this suggestive consistency. With more realistic assumptions (and complexity) regarding differences between male and female search behavior and goals, our model could perhaps support aspects of imbalanced sex ratio theory instead. (For example, if we were to introduce into the model the possibility for individuals to divorce and remarry, and a substantial number of low quality individuals later married higher-quality divorcees, then this could weaken the timing effects suggested by marital search theory.)

The current picture is nonetheless filled out by considering the mean quality of males and females in mated couples as the sex ratio changes, shown in Figure 5.12. When there is a deficit of females \( (R < 1) \), the average quality of the males who are able to mate grows increasingly higher than \( \mu \) as \( R \) gets smaller, while the average quality for mated females stays fixed at \( \mu \) because they are all able to mate. When there is an excess of females \( (R > 1) \) the reverse occurs: All males are able to mate, and the average quality of mated females increases as \( R \) gets larger. Finally, whenever \( R \) deviates from the fully balanced sex ratio \( (R = 1) \), the correlation between the qualities of individuals in mated pairs gets smaller (not plotted). This occurs because the effective quality variation among mated individuals of the more common sex gets smaller, which implies reduced linear correlation between the sexes (Aron, 1988). For example, with \( R = 1 \) the correlation between the qualities of individuals in mated pairs is .64, while with \( R = .6 \) the correlation coefficient drops to .49.

5.9 Discussion and Future Work

In the previous sections, we presented a detailed performance analysis of alternative mating strategies and selected the most efficient one as a tentative candidate to explain (part of) human

\(^{7}\)All of these sex-ratio-related effects emerge even in a simplified model where courtship time is constant for each individual (i.e., it does not change with age), and does not vary across individuals, as was introduced to explore the distribution of age at marriage.
Figure 5.12: Mean quality of mated individuals as a function of the (females to males) sex ratio $R$ on logarithmic scale (with $Y = .1$ and $K = 40$).

mating behavior. Methodologically speaking, we embraced an iterative process of designing and evaluating plausible psychological mechanisms as an attempt to reverse-engineer the functional structure of the mechanisms making up the human mind. This approach goes very much in the programmatic direction of *evolutionary psychology*, in the strict sense defined by Tooby and Cosmides more than one decade ago but still little explored (Tooby & Cosmides, 1989).

Our results show that our model better fits empirical data concerning patterns of human mating than do previous models. Our model demonstrates that individuals can make successful mating pairs after a relatively few encounters with potential mates. In contrast, 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). This is because individuals in their model do not try estimate their mate quality and use it in tuning their (probabilistic) aspiration level. Still, our model confirms Kalick and Hamilton’s intuitions that a great deal of assortative mating in human populations can be explained by a common preference for the most attractive or “best” mates (also called “type” preferences). This does not exclude the possibility that for some quality dimensions there may be “homotypic” or “like prefers like” preferences instead (e.g. height (Ellis, 1992)).

In our model, most individuals are able to find mates. Todd and Miller’s model only produced unrealistically low proportions of individuals mated (Todd & Miller, 1999), because the aspiration-guided individuals miss too many opportunities, rather than taking an initial mate and possibly swapping later to a better one. Individuals in our simulations used simple heuristics to learn about the qualities of available partners during mutual search with unknown costs and no fixed time horizon. The learning model of Mazalov and colleagues concerned optimized single-sex searching with no search costs, fixed environmental distribution of mates, and known search time (Mazalov et al., 1996). Finally, while Johnstone’s model produces statistics similar to ours, this is only accomplished 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).
Beginning with such unrealistic assumptions does not allow us to learn much about the actual design of the psychological mechanisms regulating mating decisions. Our model and the strategies described here, on the other hand, rely on plausible socio-ecological assumptions and feasible psychological designs. In particular, we found that the use of ecologically valid information such as “waiting time” allows individuals to make efficient and robust decisions without requiring substantial information gathering or computation. Moreover, the possibility of switching (tentative) partners during courtship periods adds to the mating success of individuals. Overall, our methodological commitment to psychologically and environmentally plausible mate choice mechanisms allowed us to make a set of substantial predictions that matched empirical data, and thereby better understand the nature of the adaptive problem.

We are currently working to further develop our conceptual framework of human mate choice and to extend our model in several directions. First, we are interested in studying the nature of preferences and the effect of including extra preference dimensions in the models (e.g., age of partners). Multidimensional preferences are likely to make it harder to set up appropriate aspiration levels, because different rules will apply to different dimensions and trade-offs will often be involved. Therefore, we expect to find that the complementary ability to switch partners during courtship is even more advantageous in this case. This is so, because the tradeoffs in different dimensions can be made based on comparisons between specific pairs of values (the values of the current partner and the alternative), instead of trying to estimate appropriate aspiration levels for all dimensions by taking into account what the future might bring. Second, because partnerships frequently do not last for the complete reproductive period of individuals — for example, people get divorced — we are currently working on a model that includes strategic behavior beyond the first mating and is conditional on the number and paternity of existing offspring (and can also be affected by contraceptive use). Third, we plan to explore asymmetric models where the evolutionary pressures and strategies of males and females differ. We also aim to explore the conditions in which different mating systems characteristic of human populations emerge, and the extent to which cultural distinctions and similarities can be captured in a broader model.
Chapter 6

The Cultural Evolution of Human Preferences: Case Study II

There is nothing either good or bad but thinking makes it so.

William Shakespeare, *Hamlet*

6.1 Introduction

What do miniskirts, afro haircuts, and body tattoos have in common? They are all forms of body accessories that have had a characteristic fashion like career. They emerge out of obscurity and spread through a population very fast, only to shortly after they reached their maximum popularity to vanish again from the cultural landscape, sometimes to resurgence again long after. For example, in the USA, turtleneck shirts where first popularized in the 1920s, enjoyed a resurgence in the 1960s, and became popular again in the 1990s (Long, 2002)\(^1\).

Although contemporary history of popular culture is riddled which such fashion-like phenomena, current scientific proposals to explain such forms of collective behavior are in many way unsatisfactory or limited in scope. One classical explanation was proposed by Georg Simmel’s (Simmel, 1957). According to Simmel, humans are endowed with two opposing drives or instincts: on the one hand they tend to *imitate* those individuals they admire, and on another they tend to *distinguish* themselves from those to whom they are indifferent or who they despise (Benvenuto, 2000). From this it follows that, in a stratified society, characteristic traits of upper social classes should spread down to lower classes by imitation. As this occurs, individuals in upper-classes will want to distinguish themselves from those in lower social classes. This makes them drop previously desired traits and replace them by others.

\(^1\) This chapter is partly based on published or submitted articles (Simão & Pereira, 2003d, 2003b).
While Simmel’s effect is logically sufficient to make fashion-like collective behavior to emerge (see Pedone & Conte, 2000, 2001), for a computer simulation, its assumptions present some problems. First, postulating the existence of instincts raises an important ontological question (Bateson, 2000): What do we mean, exactly, by instinct? Can’t it be used to label any behavioral regularity or pattern? If so, we should then further identify the neuro-psychological mechanisms that might generate the behavior. For example, while one can rationalize the adaptive advantages of imitating somebody else (Boyd & Richerson, 1985), the advantages involved in distinguishing oneself are much less clear. Second, how do the imitation and distinguishing behavior relate to the set of values and norms shared by a population? Could the upper-classes embrace any characteristic trait, or are these individuals also somewhat constrained by what is acceptable? While, as observers, one can retrospectively find some cultural traits strange or even bizarre (given our own cultural background), it seems that even upper class individuals have been constrained in their choices by their peers and members of their social network (e.g. by virtue of negative social reinforcement that would discourage too extravagant a behavior (Bandura, 1985)). Moreover, lower class individuals would certainly not imitate everything the upper-classes might envision. Clearly, some cultural traits are more likely to be found more often than others. That is, the existing values and norms of individuals in the lower classes also constrain what they imitate and what spreads through the population. Finally, how do individuals learn what cultural specific traits are characteristic, and how can they identify others has being members of higher class? Without having an appropriate theory of the way in which preferences and meaning are associated with symbols and traits, one ends up with a circular explanation for the emergence of fashion.

Economists too have hurdled with the problem of explaining the strange career of products that have their value change by fashion-like behavior (Bergman, 2002; Banerjee, 1992; Bikhchandani, Hirshleifer, & Welch, 1992). According to standard micro-economic theory, the demand for a product should mostly depend on the supply and an individual’s own evaluation of the product (Frank, 2003). In general, however, such theoretical framework is insufficient to explain the rapid spread of product consumption and the rapid change in individual preference characteristic of fashions. Thus, to explain this kind of phenomena, additional factors (often designated as externalities (Schelling, 1978)) are required. One such proposal is that the value of some products is proportional to the total number of its users (e.g. as in telephone or other communication systems). A high number generates a positive feedback loop that leads to rapid spread of products once an initial slow inception phase is surpassed (Katz & Shapiro, 1985). A related proposal, employed to explain the introduction and spread of innovation in markets, is to assume that the likelihood that individuals buy a product is dependent not only on its price (and therefore, its intrinsic value), but also on the number of individuals that have bought it already (Holanda, Bazzan, Gerolamo, H. A. Franco, & Martins, 2003). These explanations are however limited, because they do not explain why many products suddenly lose their value. As put by Margo Bergman: “[…] Most products are still useful to people even if they are the only ones who own them. […], miniskirts are still clothing, and therefore useful, even if they aren’t popular”.

To fix this, this author as proposed a model where in addition to considering imitation effects in product acquisition, she considers that an individual’s interest in a valued product decays with time (Bergman, 2002). While this generates a fashion-like characteristic behavior, this is no more than an ad-hoc fix in the basic model assumptions, as it does not explain why individuals’ valuation decays in the first place, neither why it could increase later in time.

An alternative type of hypothesis, proposed by some economists, is based on the assumption that individuals often imitate others blindly, possibly ignoring information available to them. This can create *information cascades* (Banerjee, 1992; Bikhchandani et al., 1992), that lead to the spreading of bad products if the initial choice is poor or ill-advised. Thus, when the product’s “real” value is later discovered and the attending information spreads, this creates a sudden drop in product use. The later kind of explanation presents two shortcomings though. First, similarly to the previous explanations, it does not apply to cases where the product is still functional and its “real” value is high. Most importantly, it assumes that there is such thing as a “real” value that can be independently assessed by the isolated individual (at least given enough information, resources, and time), without really explaining where this value comes from.

The above discussion suggests that one possible avenue to explain the emergence of fashion-like collective behavior is to identify a plausible (neuro-psychological) mechanism that would explain how individuals attribute value to traits or products, and how they might change over time. With this mechanism one could then model a population of agents with some such psychology, and examine what kind of macro patterns it would generate. Fortunately, we know of (at least) one such mechanism: classical (or Pavlovian) conditioning (LeDoux, 1998; Rolls, 1999).

In classical conditioning, previously neutral stimuli (here traits) are made to trigger emotional or behavioral responses, by recurrent association in space and/or time between them and other stimuli that, taken alone, already produced such responses (LeDoux, 1998; Rolls, 1999). A neutral stimulus is said become *conditioned* by other (unconditioned) stimuli. Although this kind of neuro-psychological process is often setup in controlled laboratory environments for experimental scientific studies, one can find evidence that the same or similar processes are also operating in more natural settings. For example, studies have shown that when physically attractive people are rated in other attributes, such as linguistic or speech skills, they receive from observers higher scores than below average attractiveness of individuals (Rosenthal & Simmerman, 1978; Kunda, 1999). And an all too well known example of “application” of conditioning mechanisms, is the marketing strategy of associating value charged stimuli (e.g. sex or status symbols) with the products or brands producers want to sell (Quart, 2003). In fact, it is reasonably well understood that learning to trigger similar responses when presented with similar or spatio-temporal correlated stimuli is a basic organizational principle of cognitive systems — both in artificial AI systems (Holland, Holyoak, Nisbett, & Thagard, 1986), and in natural brains (Arbib et al., 1998).

In this paper, we present an agent-based model where a population of agents observe others’
traits, and change their preferences for those traits by a rule akin to that of classic conditioning. This allows us to study the macro-level dynamics of preferences change and trait usage in the population. Our results show that fashion-like collective behavior can emerge out of this individual neuro-psychological mechanism, within a wide range of plausible assumptions about the level of social stratification within a population. While agents themselves change their preferences in a very natural and reasonable way — making the plausible assumption that conditioning is in general of adaptive value (Miller & Todd, 1990; Todd & Miller, 1991) — at the population level the pattern of change is found to be quite capricious and seemingly irrational.

The rest of this chapter is organized as follows: In section 6.2, we describe our model design. In section 6.4, we present the model’s results. In section 6.5, we broaden the discussion to relate our results with those of previous models, and see how our results and approach can be utilized in the wider context of cultural preference change.

### 6.2 A Model of Fashion Emergence by Conditioning

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 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\(^2\). 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 \( \sigma^2 \). 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 esthetical 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 the agent’s subjective perception of what is the value of not having the trait \( v_0^i \), or of having the trait \( v_1^i \). The values of \( v_0^i \) and \( v_1^i \) 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: \( < q_i, t_i, v_0^i, v_1^i > \).

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 \( (v_1^i) \) of having the trait is changed by an amount proportional to the average of the qualities of the

\(^2\)On the specific topic of mating preference this has been abstracted by Donald Symons as *mate value* or *mate quality* (Symons, 1979).
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 $\alpha$, 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:

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

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 $\alpha$. Formally:

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

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: $max\{3\alpha^2 - |q_i - q_j|\}^{-E, 0.001}$, 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 and low qualities have little chance of interacting amongst them, while individuals with equal 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 models (numerically illustrated in the next section).

Each 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. Once an agent changes the trait usage, we assume that it takes at least $D$ time steps until he can switch 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.
Additionally, to allow the model to escape stationary situations where all or none of the agents have the trait, we define a parameter $\theta$ 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 (possibly sub-consciously (LeDoux, 1998)). At night, back home, they sleep on the subject. In the morning, with their new sense of aesthetics properly in place, they make a fresh new decision of whether or not to wear the trait during the day. Since in this version of the model updates and actions of agent are asynchronous, to keep this metaphor holding, we should assume that each agent is performing its updates and trait changes in different days.

### 6.3 Implementation in Ethos

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

In the constructor, each agent is assigned a random quality and the subjective trait values initialized. The method 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 then it contain as agents the list of models that the Agent uses to update the subjective trait values. The other case is when Agents check for trait change.

```java
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) {
```
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 model agents, taking the average quality, and updating the trait valuations.

```java
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 methods `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 non of the two previous cases apply, a switch is made to the trait value that the agent values most.

```java
public void changeItems() {
    if (lastChange >= 0 && getTime() <= lastChange + D) {
        return;
    }
    if ((R != 0) && Global.getRandom().nextDouble() <= R) {
        item = (Item) allItems.get(0);
    }
}
```
Global.getRandom().nextInt(allItems.size());
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 to make 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 criteria QualityCriteria. Once models are obtained they are added to a TaskEnv an 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);
t.e.clearAgents();
```java
te.addAllAgents(models);
tag.act(te);
}
else if (phase == 1) {
tag.act(null);
}
}

public List getModels(Agent ag) {
    List models = Selector.select(getAllMembers(), QualityCriteria.crit,
tag, 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);
    }
}
```

### 6.4 Results

To study the model dynamics, we set the population size parameter $P = 50$, and the model set size $N = 5$. The standard deviation of agents' quality $\sigma^2$ 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 the initial value that agents attribute to having or not the trait equal to $-3 \times \sigma$. 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 6.1, we present a summary of the model’s parameters and the base values we assign them here.

In Figure 6.1, we depict a bit map (Wolfram diagram) representing trait usage across time. In the 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
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Value(s)</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P$</td>
<td>population size</td>
<td>50</td>
<td>small sample</td>
</tr>
<tr>
<td>$N$</td>
<td>number of models</td>
<td>5</td>
<td>small</td>
</tr>
<tr>
<td>$E$</td>
<td>assortment</td>
<td>4</td>
<td>$r \approx 0.75$</td>
</tr>
<tr>
<td>$\alpha$</td>
<td>1 - learning rate</td>
<td>0.2</td>
<td>fast learning</td>
</tr>
<tr>
<td>$\sigma$</td>
<td>standard deviation</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>$D$</td>
<td>delay</td>
<td>2</td>
<td>cognitive or material</td>
</tr>
</tbody>
</table>

Table 6.1: Parameter settings.

Figure 6.1: Bit map of trait usage across time ($D = 4$).

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 6.2 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. However, due to the inertia of agents in switching traits, as specified by parameter $D$, it takes some time until 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 and lower quality individuals. In Figure 6.3, 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, 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 called 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 6.3 with Figure 6.1, 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 behavior with the 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 do not have the trait of high quality individuals. This drives the model behavior into a continuum oscillation between near full trait usage to near full trait avoidance. This can be verified in Figure 6.2.

To test the robustness of this result we start by varying the value of the delay parameter \( D \). In
Figure 6.3: Bit map of trait usage across time ($D = 4$) with deterministic selection of model.

Figure 6.4: Bit map of trait usage across time ($D = 10$).

Figure 6.4, we present a bit map of trait usage across time with $D$ set to 10, and in Figure 6.5 with $D$ set to 0. As can be seen upon inspecting Figure 6.4, with large value of $D$ the behavior of the model is qualitatively very similar for most values — it produces wave-like patterns across all ranges of agents’ quality. On the other hand, in Figure 6.4 we can see that when $D = 0$ (no enforced delay between trait switches) the patterns are not so clear and appear almost random. This is so because agents respond faster to devaluations of trait usage. We also systematically varied the learning rate parameter, and the number of models, and we found no important changes in model behavior. Thus, a tentative conclusion that we take from our simulation results is that cognitive inertia or material resources is an important factor in establishing the duration of a ”fashion”-like phenomena. Although, in a real setting many more factors will be involved.
6.5 Discussion and Future Work

Our model show how a constant oscillation between a near full trait usage and near full avoidance can occur. It postulates only very simple psychological mechanisms of valuation or conditioning, a (normal) quality distribution of agents, some criteria for the selection of model agents, and most importantly a cognitive delay or material cost between two successive switches. The act of switching traits can be interpreted as imitation, but to the extent that one assumes the switching of traits corresponds to the invocation of pre-existing behavior routines, then postulating imitation as a psychological mechanism is not absolutely required (Noble & Todd, 2002). In any case, we do not need to postulate the existence of two "drives\” as in Simmel’s effect, one for imitation and another for wanting to be different.

Thus, we put forward a new mechanism based on conditioning to explain fashion-like collective behavior. This is an alternative to other standard explanations such as the Simmel effect, or postulating irrational agents, as discussed in the introduction section. Still, much work needs to be done. In our simple model the imposed delay was the most important factor in establishing the period of fashion waves. It would be interesting to see under what conditions the period of the waves can be decoupled from direct influence of model parameters and instead emerge from the model dynamics.

A important addition to the model is to consider agents with multiple changeable traits, and traits with multiple values. This would most probably change the model dynamics for each individual trait. (See (Axelrod, 1997b) for a similar model design.) Another aspect to consider would be to add differential abilities to switch traits based on an agent’s quality, representing some form of social class empowering.

Figure 6.5: Bit map of trait usage across time ($D = 0$).
Chapter 7

Conclusions

It is good to have an end to journey toward; but it is the journey that matters, in the end.

Ursula K. Le Guin, in The Little Zen companion

In this thesis, we addressed the issue of how to devise computational tools for modeling and simulation of human social behavior and cultural processes. In chapter 4, we described Ethos, a MAS framework devised to support the development of agent-based models of human social behavior and culture change. We described its main abstractions and how they fit together. To test the usefulness of the framework we presented several examples that use its abstractions, including a model previously published in the literature. Additionally, in chapter 5 and chapter 6 we presented two novel models that make extensive use of Ethos and show how substantially different models can be expressed naturally with its abstractions. Overall, we concluded that Ethos's features can simplify the modeling process of a wide range of agent-based models intended to study human social behavior. The intrinsic tradeoff is that the modeler needs to be aware of the specifics of the framework. We hope our approach can motivate other researchers both to use the proposed abstractions, and/or devise alternative sets of abstractions that prove more useful than the ones provided in Ethos.

In chapter 5, we have shown how an evolutionary (functional) analysis of mate choice can be combined with an agent-based modeling approach to gain insights into the processes underlying human sexual/romantic relationships. In particular, by building extended courtship processes, the ability to switch partners, and other key aspects of the mating game into our model, we have accounted for existing data and generated predictions in ways unattainable by earlier models or verbal theories. We hope that our work in this model will motivate further research on more realistic and informative models of the psychological mechanisms underlying social behavior in humans (and other species). This should in turn help social psychology to escape the dangers of theory-blind empiricism and turn more to theory-guided experiments. Finally, we hope that
the design and engineering insights that come from building computational models will help push forward a new view of ecologically rational social cognition: social beings as effective decision makers using simple mechanisms tuned to specific environmental contexts, rather than as general-purpose utility maximization calculators pounding at all social problems with the same big stick.

In chapter 6, we have presented a simple model that shows how mechanisms of neuro-psychological conditioning at the individual level can generate the emergence of fashion-like collective behavior. The model shows that for a wide range of parameter values, trait usage can oscillate continuously between periods of near full trait use to near total trait avoidance. This model provides an alternative to other explanations proposed in the literature to address the issue of transient preferences. Still, some further work comparing the model results with empirical data on preferences change would be desirable.

To keep testing the usefulness of Ethos’s main abstractions (possibly extending and refining them), we plan to implement additional models of human social behavior, 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 the framework which 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 than just those provided by current MAS. Still, further work is required to learn which of Ethos’s features are most useful and which others should be added.

Overall, in light of the experiments presented in this thesis, we conclude that the task of building computational agent-based models of human social behavior and culture is both a feasible and useful task. Feasible because the design of software frameworks for simulation of multi-agent systems considerably simplifies the task of building models. Modelers can this way focus on the specifics of the models’ design using the structuring and development tools provided by the framework without worrying too much about underlying computational details. Useful because these models can be used to gain considerable insights about the nature of the processes of human social interaction. This includes both a general understanding of the dynamics generated by the model and actual qualitative predictions about population level patterns. It became clear from our experience, that the ultimate responsibility for adequacy in model building is with the model designer which must ensure that the used abstractions are appropriate to the target domain. Simulation frameworks can provided some help by deploying an intuitive ontology that focuses the modeler’s work, but a case by case analysis of the domain is always required. The models of human mate choice and cultural change presented in this thesis showed how this symbiosis
between generic simulation frameworks and careful analysis of specific problem domains can be successfully worked out. We hope our approach can entice other researchers to follow the same path.
References


