Departament de Llenguatges i Sistemes Informatics
UNIVERSITAT POLITÈCNICA DE CATALUNYA

Master Thesis Project

Inferring Best Strategies from the Aggregation of Information from Multiple Agents:

The Cultural Approach

Advisor	Tutor	Candidate
Dr. Ulises Cortés	Dr. Ulises Cortés	Esteve Almirall

June 2010

Table of Contents

1 Introduction

Learning in general and learning in MAS in particular, has been described as a collective process, but one that draws mostly from the direct interaction with other agents. In contrast with that, Cultural Algorithms base their ideas on the way societies learn, by modeling a process of generalization of individual knowledge into a common space.

However, Cultural Algorithms do not stop in modeling of this process of inference, they also address the one of adoption by modeling *how* individual agents select and adopt this common knowledge, incorporating it in their own stock.

Therefore, one of the distinctive characteristics of Cultural Algorithms is its micro-evolutionary perspective, focusing in the mechanisms employed by the agents to generalize and incorporate knowledge. By doing so, we are able to not only apply realistic mechanisms but assess their impact in the system.

This work addresses Cultural Algorithms from a rather particular perspective: the one of complexity, assessing the mechanisms and system dynamics at different levels of complexity by exploring three main aspects.

Firstly a key element of the mechanism of selection: the number of best cases to be investigated in the selection process. In fact, Cultural Algorithms distinguish themselves because of their frugality: a simple observable public signal drives the process of selection and generalization. In the present work we focus on a very concrete aspect of this process, on the optimal number of best cases, uncovering a novel aspect of Cultural Algorithms: their frugality in terms of best cases across all levels of complexity.

Secondly we focus on the dynamics of the whole process, revealing its social aspects in terms of hypothesis building and social confirmation or refutation by adopting it. Adoption is therefore portrayed as a key element in this process, an element that to a great extend is able to amplify and substitute the discovery process done by the agents.

And thirdly, we discuss a mix process of pattern adoption and individual exploration. Taking as granted that some agents will embark themselves in a process of individual exploration we demand again what is the optimal number. Is it better that all agents engage themselves in a costly process of exploration or Cultural Algorithms with their multiplicative power will lesser this requirement? Again we find that if a small fraction of agents engage in exploration it is enough to push the whole population to its best results.

In order to present this research, this document is organized as follows.

Chapter two we will devoted to discuss the motivations of this work.

In chapter three we describe evolutionary algorithms and their operators in order to provide a framework where to build and position this research.

In chapter four we summarize the basics of Cultural Algorithms. Chapter five is devoted to position these Cultural Algorithms in the context of multi-agent systems paying attention to their application

Chapter six introduces NK landscapes, our basic landscape structure where agents conduct their search process.

In chapter seven we present our work modeling pattern-based learning in NK landscapes. This serves as the basis to develop our first results in chapter eight. There we will elucidate on the optimal quantity of best cases upon which to generalize with a unexpected result: less is more, both in terms of the number of the agents selected for generalization and in terms of the complexity of the patterns.

In chapter nine we discuss the case of mixed situations where agents rely both on patterns and on their own exploration capacities. There we will explore the minimum quantity of agents performing the costly process of individual exploration, needed to attain optimal results.

And finally in chapter ten we conclude and present some insights for future work.

2 Motivation

Although learning in MAS is described as a collective experience, most of the times its modeling draws solely or mostly on the results of the interaction between the agents.

This abruptly contrasts with our everyday experience where learning relies, to a great extent, on a large stock of already codified knowledge rather than on the direct interaction among the agents.

If in the course human history this reliance on already codified knowledge had a significant importance, especially since the discovery of writing, during the last decade the size and availability of this stock has increased notably because of the Internet.

Even more, humanity has endowed itself with institutions and organizations devoted to fulfill the role of codifying, preserving and diffusing knowledge since its early days.

Cultural Algorithms are one of the few cases where the modeling of this process, although in a limited way, has been attempted.

However, even in this case, the modeling lacks some of the characteristics that have made it so successful in human populations, notably its frugality in learning only from a rather small subset of the population and a discussion of its dynamics in terms of hypothesis generation and falsification and the relationship between adaptation and discovery.

A deep understanding of this process of collective learning, in all its aspects of generalization and re-adoption of this collective and distilled knowledge, together with its diffusion is a key element to understand how human communities function and how a mixed community of humans and electronic agents could effectively learn. And this is more important now than ever because this process has become not only global and available to large populations but also has largely increased its speed.

This research aims to contribute to cover this gap, elucidating on the frugality of the mechanism while mapping it in a framework characterized by a variable level of complexity of knowledge.

Also seeks to understand the macro dynamics resulting from the micro mechanisms and strategies chosen by the agents.

Nevertheless, as any exercise based on modeling, it portrays a stylized description of reality that misses important points and significant aspects of the real behavior. In this case, while we will focus on individual learning and on the process of generalization and ulterior re-use of these generalizations, learning from other agents is notably

absent. We believe however, that this choice contributes to make our model easier to understand and easier to expose the causality relationships emerging from our simulation exercises without sacrificing any significant result.

3 Evolutionary Algorithms

The evolutionary approach has enjoyed a tremendous success in the last decades. From its applications to related fields such as modeling societal or economic behavior to others seemingly unrelated such as optimization, it has proven its value.

Evolutionary Algorithms have their inspiration in the work of Darwin, around three fundamental concepts: *replication*, *variation* and *natural selection*.

Although replication is fundamental for the survival of a specie, replication does not ensure evolution, because replication produces identical copies. For evolution to exist variation is needed and variation is ensured by two mechanisms. First, caused by the errors in replication process fails to produce identical copies and those result into mutations. And secondly, because of sexual recombination that is itself a product of evolution.

The third basic element of evolution is selection. The adaptation of life forms to the environment causes the *survival of the fittest*, where forms that are better adapted and do not conflict with a certain environment have better chances to survive and reproduce than ones that continuously strive for survival. This has important implications, because the fitness of an individual can only be defined respect to the environment where he lives. Therefore a whale that is very fit in big oceans cannot even survive in rivers or in mainland.

Each organism carries genetic information referred as *genotype*. During his life period, the organism develops traits that constitute its *phenotype*. Both determine the genetic information that is passed to next generation, where organisms can be regarded as mortal vehicles for transmitting potentially immortal genetic information.

Living organisms are built on the basis of set of *chromosomes*, which are strings of DNA (*deoxyribonucleic acid*); chromosomes are the blueprints of a living organism. Chromosomes are functionally divided into genes, blocks of DNA that encode a protein. Each gene has a *locus* in a chromosome and has different settings called *alleles*. In a very simplified model we can associate a gene to a trait, therefore skin or hair color will be associated to a gene. Complex organisms have more than one chromosome; all of them taken together constitute the collection of genetic material that is called a genome.

In nature we can find two forms of reproduction. The first is asexual, were a living organism produces copies of itself. There variations exist because of mutations, because of errors in the copies. Mutations occur by changes alleles in genes or

changing the position of genes, their loci or by deleting some genes. In general any errors that produce an imperfect copy result in a mutation that enhances variation.

The other form of reproduction is sexual recombination. There, genes are exchanged in an operation that can be thought as a crossover of chromosomes.

The fitness of an organism is defined in two forms, first as the probability that this organism will live enough to reproduce, also called viability and secondly as the number of offspring that the organism has, also called fertility.

The key element for variation is the level of interaction between genes, called *epitasis*. Because of this level of interaction, the result between of genetic variations derived from mutations or sexual recombination is hardly predictable, producing the rich explosion of life that fascinated Darwin and all of us.

These ideas were adopted in Computer Science during the 60’s with three but simultaneous different strands of research.

Evolutionary Programming was developed in the 60's by Fogel, Owens and Walsh (Fogel,). They studied systems that on the basis of a given goal, predicted their environment and by doing that showed intelligent behavior. Their models were developed on finite state machines using mutation and selection.

Also during the 60's in the Technical University of Berlin, Rechenberg and Schwefel (Rechenberg,) introduced evolutionary strategies as an approach to optimization for devices. Their approach focused on continuous parameter optimization using mutation and selection.

However, the best well-known approach is Genetic Algorithms. Genetic Algorithms were invented by Holland in the 60's (Holland,). Holland's goal was to study the role of adaptation in nature and to incorporate it into computer science. To that effect, he developed a theoretical framework where chromosomes were represented by strings of bits (zeros and ones) where operators inspired in the genetic model were applied. These basic operators were crossover, mutation and inversion, genes were bits and the alleles were represented the binary digits "0" and "1". Genetic Algorithms was also the only approach-using crossover.

In the last decades there has been a big interest in evolutionary approaches and the barriers between the three approaches have diluted and disappeared. Also, the field has been extended and evolutionary algorithms have found a place in many areas such as in machine learning, *etc* ...

3.1 Evolutionary Algorithms. The Basics

Without referring to any particular case, a template for a generic evolutionary algorithm is presented in Algorithm 3.

Algorithm 3 Generic Evolutionary Algorithm

```
Input   : P  - population at t
Output: P  - population at  t+1

function EvolutionaryAlgorithm : (P ) → P
    t=0
    evolution = true
    IniPopulation(P(t))
    CalculateFitness( P(t))

    repeat
            P' = SelectForVariation(P(t))   - P' ⊆ P
            Mutate(P')
            Recombine(P')
            CalculateFitness( P(t))
            P(t+1) = SelectForSurvival( P(t))
            t = t + 1
    until  evolution = false
    return P
end
```

There, we can observe how initially, the fitness of a population is calculated. The calculation of fitness is sometimes inherent to the model, as in the case of using landscapes such as NK models, but many times and specially when using evolutionary algorithms to solve optimization problems, correspond to the objective function. In other cases, such as in evolutionary economics (Dosi, 1988), fitness corresponds to a function that aims to map, in stylized form, the most relevant aspects of the problem. In all cases, how fitness is calculated is a central aspect of any evolutionary algorithm and determines much of it.

The next step of any evolutionary algorithm is the selection of a part of the population, where genetic operators: *mutation*, *crossover* and *inversion* will be applied. Three families of methods are commonly used for selecting for variation.

The first one takes into account the fitness of each individual and selects it on the base of its fitness. Therefore individuals with high fitness are more probable to be selected. Proportionate fitness selection, also known as *Roulette Wheel Selection*, mimics nature in the sense that individuals with a higher fitness are more prone to reproduce and

therefore their genes to cross-over or mutate. The probability of an individual s_i to be selected will be given by,

$$p(s_i) = \frac{f(s_i)}{\sum_j f(s_j)}$$

However, fitness proportionality also has problems as a selection method, being the most common the fact that as variance of fitness decreases the sampling becomes more random. Maintaining the selection pressure independent of the variance of fitness values in a population is without any doubt interesting. A proposal that achieves that objective is the *linear ranking model* (17), where the probability of selecting an individual s_i is given by,

$$p(s_i) = p_{max} - (p_{max} - p_{min})\frac{i-1}{n-1}$$

where *n* is the size of a population ordered by fitness where *i* denotes this ordering, p_{max} and p_{min} denote the maximum and minimum selection probability. As in the case of selection proportional to fitness, ranking methods can be extended to non-linear functions.

A third method of selection is tournament selection (Miller and Golberg, 1995) where *k* individuals are selected at random from the population, being *k* the tournament size, then with probability *p* the best individual from the pool is selected, with probability $p\ (1-p)$ the second individual, with probability $p\ (1-p)^2$ the third individual and so on. A 1-way tournament, denoting the case of *p* =1 is equivalent to random selection.

To this sample, selected by using one of these three families of methodologies, genetic operators are applied and after that, a second type of selection must be performed, the one determining which individuals will survive and which one will not.

The simplest way to approach the issue is through the *imitation* of biological systems. This is precisely what generational replacement does. There, parents are being replaced by their offspring. It is common to find this approach associated with a fitness proportionate selection.

Even if the dynamics of short-lived species resembles this approach where the whole population is replaced by its offspring, this is not true for long-lived ones. There, each generation produces only a few new members that live concurrently with their parents. Steady state selection aims to reproduce this approach, several variations exist like *oldest replacement* or *worst replacement.*

There are also two classes of Evolutionary Strategies (ES) used in nature-inspired search and optimization algorithms, known as $(\mu, \lambda) - ES$ and $(\mu + \lambda) - ES$. In both

cases μ denotes the number of parents and λ the offspring. In the first case μ parents are replaced by the best of the λ offspring and in the second case the best μ individuals are chosen from a temporary population comprising the μ parents and the λ offspring.

Furthermore, other strategies and combinations of strategies exist like for example, elitism (only the best individuals survive) or duplicate checking (identical copies are not included).

3.1.1 Genetic Operators

Mutation and recombination depend on the use of genetic operators to obtain the candidate solutions. Although in our work we will only use operators in bit strings, we will also discuss genetic operators in continuous variables. The use of genetic operators can also be extended to other structures like finite state machines or trees.

Let us assume in the following that *A* and *B* are binary strings representing a candidate solution, $A, B \in \{0,1\}^n$.

One-point crossover

This operator works by randomly selecting a point *p* that will be used to cut the bit strings in two parts. The operator connects the head of the first bit string with the tail of the second bit string (or vice versa). Therefore, one point-crossover produces two solutions.

$$A'_i = \begin{cases} A_i \ if \ i \leq p \\ B_i \ if \ i > p \end{cases} \quad and \quad B'_i = \begin{cases} B_i \ if \ i \leq p \\ A_i \ if \ i > p \end{cases}$$

The following example illustrates the one-point crossover operator,

$$\begin{matrix} A = 10011 \mid 100 \\ B = 01101 \mid 010 \end{matrix} \rightarrow \begin{matrix} A' = 10011 \mid 010 \\ B' = 01101 \mid 100 \end{matrix}$$

Two-point crossover

If instead of cutting in one point, we cut in two points we obtain a new operator: two-point crossover that can be easily generalized to a k-point crossover. For the sake of the example we will discuss the two-point version.

In the two-point version we will have to randomly chosen cutting points p_1 and p_2 $(p_1 \leq p_2)$ resulting in three pieces. As before two solutions *A'* i *B'* will be generated.

$$A'_i = \begin{cases} B_i \ if \ p_1 \leq i \leq p_2 \\ A_i \ otherwise \end{cases} \quad and \quad B'_i = \begin{cases} A_i \ if \ p_1 i \leq p_2 \\ B_i \ otherwise \end{cases}$$

Again we will use an example to illustrate how the two-point crossover works,

$$\begin{matrix} A = 100 \mid 11 \mid 100 \\ B = 011 \mid 01 \mid 010 \end{matrix} \rightarrow \begin{matrix} A' = 100 \mid 01 \mid 100 \\ B' = 011 \mid 11 \mid 010 \end{matrix}$$

Uniform crossover

Uniform crossover is again a generalization of crossover. In that case a mask is used with a 1 indicating the position where the bit has to be copied and a 0 otherwise.

$$A_i' = \begin{cases} B_i \; if \; M_i = 1 \\ A_i \; otherwise \end{cases} \quad and \quad B_i' = \begin{cases} A_i \; if \; M_i = 1 \\ B_i \; otherwise \end{cases}$$

The following example illustrates how it works,

$$\begin{matrix} A = 10011100 \\ B = 01101010 \end{matrix} \rightarrow M = 01010101 \; \begin{matrix} A' = 11001000 \\ B' = 00111110 \end{matrix}$$

Bit flip mutation

The bit flip operation consists simply in flipping one or more genes in the genome or, in the case of bit strings, changing their value from 0 to 1 or vice versa. For example

$$A = 01011\underline{0}11 \rightarrow A' = 01011\underline{1}11$$

The bit flip operator is predominantly implemented in two ways. One way is to predefine the number of bits to flip and select the loci randomly. The other one consists in predefining a rate in which each component is flipped. The effect could be similar depending on how these two mechanisms are implemented. For example, a mutation of a single bit randomly selected will have in the limit the same effect than a mutation of each bit with a rate of *1/n*, being n the string length.

Inversion

Inversion is the second mutation operator. In that case two points in the bit string are chosen and the enclosed sub-bit string is reversed.

$$A = 010|110|11 \rightarrow A' = 010|011|11$$

We have discussed two types of operators: crossover and mutation. In genetic algorithms the bulk of the work is carried on by crossover operators. Mutation is normally applied to the offspring, after crossover with the objective of adding diversity to the population, preventing that way a premature convergence of the population or the system.

4 Basics of Cultural Algorithms

Genetic algorithms (Holland, 1975) are a powerful tool for mapping biological evolution and, as we have already discussed, its use extends far beyond of a modeling exercise to areas such as optimization or in general a widespread use in Artificial Intelligence.

However, powerful as they are, they are not adequate to model cultural evolution that is a fundamental trait of human societies. Although there are points of coincidence, cultural evolution has its own differential characteristics. Among them the fact that is orders of magnitude faster than genetic evolution and the existence of a shared belief space that allows individuals to directly incorporate knowledge from it without having to relearn everything from scratch either by themselves or by model other individuals.

There have been however, attempts to have a more direct map of genetic algorithms to cultural spaces. Perhaps the best well known of these attempts is *memes*, an adaptation of the Greek work *mimema* (meaning something imitated). Richard Dawkings introduced the notion of *Memes* in the extremely popular book ***The Selfish Gene*** (Dawkins, 1976). Dawkins defined *memes* as counterparts of genes in the cultural world. Memes are therefore the unit of cultural transmission or imitation such as a piece of thought, a fragment of music and so on. Memes propagate through imitation and evolve when three conditions exist:

1) variation, when change is introduced to existing elements either by errors in copying them or by recombination,

2) replication, when the capacity to make copies of memes exist, and

3) differential fitness, so the fittest to a certain environment will survive.

Memes focus on a micro-evolutionary perspective of the cultural evolutionary process, by paying attention only to the transmission of ideas, behaviors, traits, etc. between individuals of a population and discarding a macro perspective where a shared space of traits, behaviors and ideas is maintained. Cultural algorithms attempt to model both levels.

4.1 Cultural Algorithms

Eduard B. Tylor was the first to introduce the term *Culture* back in 1881 in his book **Primitive Culture** (Tylor, 1881). There, he described culture as "*that complex whole that includes knowledge, art, morals, customs, and any other capabilities and habits*

acquired by man as a member of society". These early approaches to culture where characterized with this type of understanding, leading to classifications into groups. For example, George Murdoch (Murdoch, 1975) produced a catalogue of 565 cultures based upon 30 different characteristics.

The research in Cybernetics and System Theory during the 60's brought with them a new understanding of culture, where the concept of group and feedback mechanism was central. Culture was regarded as a system that interacted with its environment with positive and negative feedbacks that amplify and counteract behavioral deviations of individuals within a cultural group (Flannery, 1968). Also in the 60's Cultural Ecology emerged as a discipline concerned with the nature of the interaction between the cultural system and its environment.

In the 70's we saw a renew emphasis on presenting culture as an information system and a rising concern on the flow of this information. For example, for Geertz "*Culture is the fabric of meaning in terms of which human beings interpret their experience and guide their actions*" (Geertz, 1973). In the same line of thought, Durham defined culture as a "*system of symbolically encoded conceptual phenomenon that are socially and historically transmitted within and between populations*" (Durham, 1990).

This conceptualization of culture as a space of encoded concepts available to the whole population is the departing point of Cultural Algorithms in contrast to other mechanisms such as memes.

Cultural Algorithms were introduced by Reynolds (Reynolds, 1994) aiming to model the cultural evolution process from both a micro-evolutionary perspective in terms of transmission of knowledge, behaviors or traits among individuals of a population and from a macro-evolutionary perspective in terms of generalization of this knowledge from individual experiences. A key element supporting on one side this generalization and on the other its influence to individuals is a dual inheritance system that enables both generalization and influence.

In the macro-evolutionary level individuals create a *mappa*, a generalization of their beliefs, knowledge and preferences. Individual *mappa* can be merged creating group *mappa*, the dual inheritance system allows communication between these two levels. In cultural algorithms these two levels are mapped into two spaces, a population space and a belief space. In the population space we can find individuals that evolve in the conventional way of evolutionary computation. The belief space contains information that individuals have acquired during the evolutionary process and is available to the whole population. A communication protocol connects both spaces and determines the exchanges and what information is retained. For example, the communication protocol can establish that only individuals with the best fitness can update the belief space.

Cultural algorithms have been successfully applied in a number of domains, such as multi-objective optimization (Coelo and Becerra, 2003) and constraint based problems (Reynolds, Michalevicz and Cavaretta, 1995, Jin & Reynolds, 200). Of special interest in our work is their use in multi-agent systems to which we will devote a section.

4.2 A Computational Framework

As we just mentioned, the data part of the framework is supported by two spaces. One of those corresponding to individuals consists of a set of traits and behaviors. And another one corresponds to the whole population that generalizes these individual traits and behaviors.

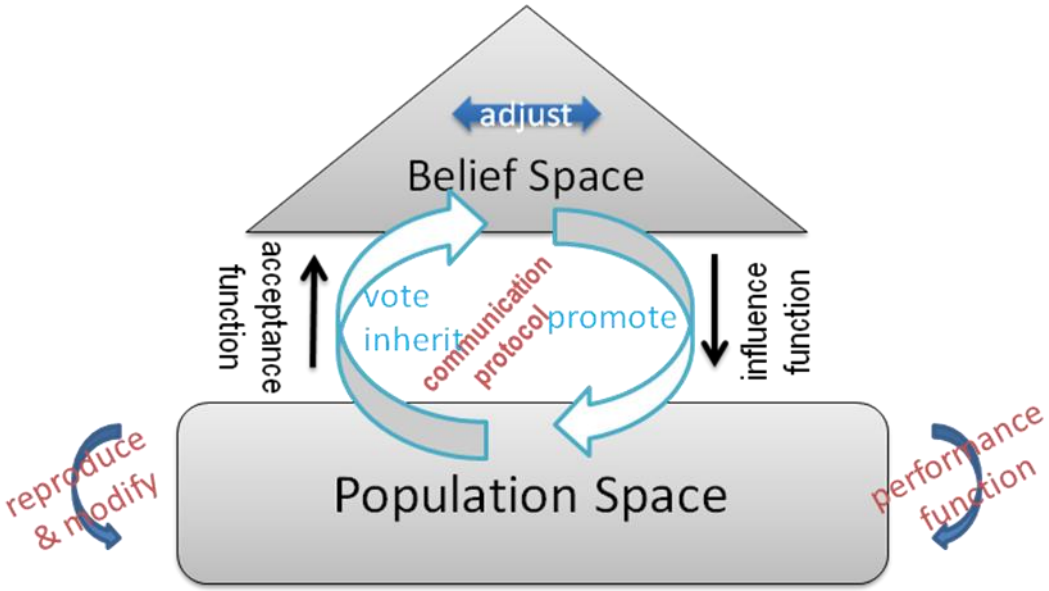


Fig 2. Belief and Population Spaces in Cultural Algorithms and their mechanisms

In Figure 2 we can appreciate a representation of the underlying mechanisms in cultural algorithms.

Individuals in the population space are evaluated by means of the performance function and each individual produces a generalized map based on their actual experience (this process is called outlining). All these beliefs, coming from individual generalizations, are combined in the belief space. If the combined performance of these generalizations passes the acceptance function then they are incorporated to the belief space, if not they are discarded, that way the belief space is adjusted.

At its turn, the belief space can be used to modify the performance of individuals of the population, modify the set of available traits, enforce beliefs that have been discarded because of lack of support in the acceptance function, etc. These processes are mapped by the influence function.

Once the influence function has been applied and the population evaluated a process of reproduction is applied. In this part, cultural algorithms follow quite precisely their genetic counterparts. First a part of the population is selected as parents of the new generation who by using genetic operators produce offspring.

Communication protocols are an important aspect of the mechanism, because they determine the relation between the population space and the belief space by means of the acceptance and influence functions.

A commonly use protocol is `Vote-Inherit-Promote`. In this protocol first the performance of an individual in the population space is evaluated by voting. After those are aggregated in the belief space and pruned using the acceptance function. In this protocol, the belief space inherits the aggregate individual performance that finally is used to promote these individuals in the population space giving those more chances of being selected and reproduce.

A representation of this mechanism in *pseudo*-code is presented in the following algorithm,

Algorithm 4 Generic Cultural Algorithm

```
Input   : P  - population at t
Output: P  - population at  t+1

function CulturalAlgorithm : (P ) → P
    t=0
    evolution = true
    IniPopulation(P(t))
    IniBeliefSpace( B(t))

    repeat
            EvaluatePopulation( P(t))
            AdjustBeliefSpace( B(t), Acept( P(t))
            InfluencePopulation( P(t), B(t) )
            P' = selectForVariation(P(t))   - P' ⊆ P
            P=Evolve(P',P)
            t = t + 1
    until  evolution = false
    return P
end
```

Here we can observe the same elements described before arranged in a procedural fashion.

First both population and belief space are initialized and a loop repeats until the simulation ends.

In the main loop, we begin first evaluating the population and assigning a fitness to each individual. After the process of outlining and the construction of the belief space is carried out by the `AdjustBeliefSpace` function, on the basis of the current version of the belief space and the acceptance function applied to the population, selecting the generalizations of beliefs from the population that enjoy enough support according to the acceptance function.

`InfluencePopulation` maps the effect of the influence function where on the basis of the belief space, performance, traits or beliefs of the population are modified.

Finally, `SelectForVariation` and `Evolve` that support not only variation but also survival perform the evolution of the population, the part that more closely corresponds to genetic algorithms.

4.3 Version-Space Guided Genetic Algorithm (VGA)

There are many variations in terms of implementation of the framework presented above. One of such implementation is Version-Space Guided Genetic Algorithm (Reynolds, 1994), where the micro-evolutionary process is modeled using genetic algorithm while the belief space represents `schemadata` or generalizations of individual strategies.

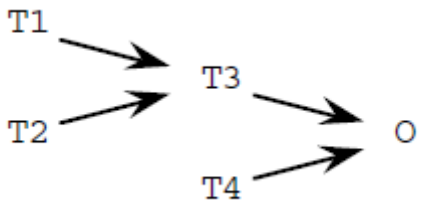
A `schemadata` in terms of genetic algorithms is a template that identifies a set of similar strings. For example, if we consider only binary strings the schema 0**1*01 represents binary strings of length 7 with 0 at positions 1 and 6 and 1 at positions 4 and 7. Any implementation of a cultural algorithm must choose a way to implement the evolution in the population spaces and the beliefs in the belief space, together with the acceptance and influence functions.

With respect to the evolution in the population space, it can be formalized in many different ways, Evolutionary Programming and Genetic Algorithms among them. VGA uses genetic algorithms for this purpose among other things because of its roots in cultural simulations where understanding the acceleration rate of this evolution is highly relevant. The schema theorem (Holland, 1975) in genetic algorithms provides a basis that can be used as a benchmark for this purpose.

Regarding the belief space, VGA uses a lattice or Version Spaces (Mitchell, 1978), where each schemadata represents a set of individuals and is organized in a tree. The root of this tree represents the whole population. This representation fits pretty well

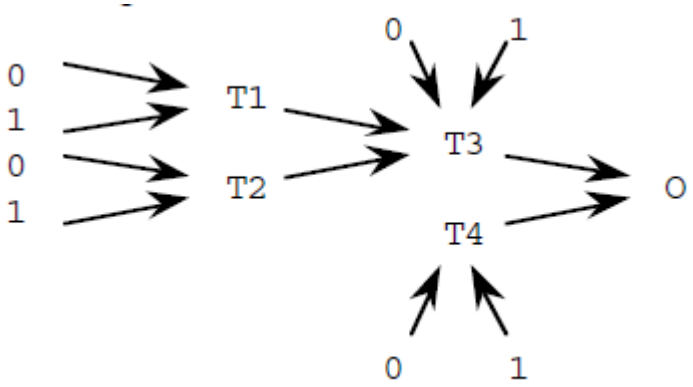
with the idea that a key aspect in the development of a culture is the ability and the need for classification and abstraction of common characteristics.

An example of this tree structures is presented below.



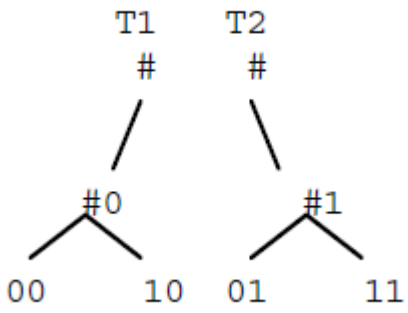
Note that in this case there are two prerequisite structures: T3 is supported by T1 and T2 and T4 has no prerequisites. These types of structures are represented by ordered sequences; normally post-order transversal is preferred, producing T1, T2, T3, T4.

Augmenting the graph with leaf nodes containing all possible values, such as produces concrete specifications,



Therefore, all possible chromosomes are obtained by traversing the nodes. In this example, chromosome 0110 will be obtained by choosing a 0 for T1, a 1 for T2, a 1 for T3 and a 0 for T4.

The belief space is a representation of all series of unique combinations resulting from traversing these trees, for example,



Individual mappa is constructed by subsets of the belief space with a process called Outlining that depends on two functions, one that determines the acceptability and another that determines the performance.

The process of outlining is one of generalizing solutions. Through this process two parts of the mappa are produced with linear complexity, the G-Mappa, corresponding to the generalization and the S-Mappa that describes the specialization set. This can be easily observed in the simple case presented above. There we have two instances 00 and 01, which produce a generalization #0 because a 1 in the first position is not acceptable.

5 Cultural Algorithms & Multi-Agent Systems

Human evolution has distinctive characteristics that many times, are not adequately mapped with the traditional toolset of Artificial Intelligence. In fact, human societies have endowed themselves of a mechanism of storage and transmission of information pretty unique: culture. Culture is independent of the individuals, being potentially (and thanks to Internet this potentiality is becoming real) available to all individuals. Adaptation is therefore not taking place only in terms of genetic but also in terms of cultural information. This key aspect provides not only with the possibility of a faster rate of adaptation but also with the capacity of storing much more information than in its genetic counterpart.

In a cultural system, knowledge is stored and evolves not only at individual level, but also at societal level. Thus, cultural systems can be viewed as dual inheritance systems where knowledge is stored and evolves at two levels: a) at individual level and b) at the level of acquired beliefs, both levels interact and co-evolve (Durham, 1991).

Cultural algorithms were developed by Reynolds (1994, 1997) with the purpose of modeling cultural evolution. Therefore, because the mechanism that they aimed at modeling is inherently agent-based and distributed, cultural algorithms even if designed with an intention that lies far away from Multi-Agent systems, belong since the beginning to this category. The earliest models of cultural algorithms (Reynolds, 1979) were motivated by existing evolutionary computational models were an individual consisted of a vector of traits and was affected by other individuals. The basic operators used were analogous to the ones in genetic algorithms. However other socially motivated operators, such as *copying*, were introduced (Boyd and Richerson, 1985).

The first model was used to show how the agriculture could have evolved as an accumulation of small-scale resource scheduling decisions, taking as an example the Valley of Oaxaca in Mexico. In these early models the whole population was used (the acceptance function was 100%) and the probability of applying an operator was proportional to the increase in fitness, performance that this new operator produced when incorporated. Likewise, it deceased with decreased performance. The influence function was based on the current beliefs of the agents about the utility of the operators.

A second generation of models was characterized by using a lattice to represent the belief space where leafs nodes were explicit population members. Learning in the

belief space was done using Version Space algorithms (Mitchell, 1997 whose information structure we briefly presented in the previous section. These algorithms are able to identify regions in the lattice were the likelihood for encountering individuals with a fitness above the average is higher.

Also the process of generalization-specialization was accelerated by using local search procedures that systematically changed some bits in order to finds out which ones resulted in higher fitness and which ones were irrelevant. The resulting description of individuals contained do not cares "#" and were called stable classes because their performance did not change for the specified bits. The further integration of these individuals in the belief space resulted in even more generalized classes.

A schema theorem for Version Space Guided Genetic Algorithm was developed by Reynolds (1978 to illustrate how culture can accelerate learning in genetic algorithms.

This second generation also introduced the recombination of lattices in the belief space, further accelerating the process of cultural evolution. Another important change was in the acceptance function that used to take all individuals in the previous generation and in this one, only the ones above average, resulting in a 50% of the population.

Is in this second generation of Cultural Algorithms when we encounter a notable increase in applications that leave the original orientation of modeling human populations to a wide range of fields, from optimization to program understanding (Reynolds and Sverdlik, 1995) or concept learning problems (Reynolds and Maletic, 1993).

The third generation of cultural algorithms developed as a response to the need of dealing with real-value genetic algorithms (Michalewicz, 1994). There are many strategies that support real-value traits but we will focus on Cultural Algorithms with Evolutionary Programming because it has become the more common. In this case, ranges of real values represented traits in the agents. The belief space consisted of the traits, in form of real values, coming from the best individuals that surpassed a threshold. Typical acceptance rates are around 20% (Bäck, 1996). The evolution among the population is driven by the generation of offspring solutions by applying mutation operators and tournaments for carrying on the selection process.

The applications expanded to cover other areas beyond optimization. They included learning among cooperating agents (Reynolds and Chung, 1997), data mining of large scale spatial-temporal databases (AlShehri, 1998), re-engineering rule based systems (Reynolds and Sternberg, 1997), gradient approaches to image segmentation (Reynolds and Rolnick, 1995) among others.

5.1 Agent-Based Historical Reconstructions

One of the most evident uses of Cultural Algorithms is to simulate the behavior of human populations, representing how the knowledge of each individual is shared among the whole population (Kobti, Reynolds and Kohler, 2003). The first model of this type was built by Kohler (2000) attempting to explain the evolution and eventual disappearance of the pre-Hispanic Anasazi settlement in Mesa Verde (Colorado, USA). The aim of this first model was to understand the disappearance from the region. Later this initial model was transformed into a multi-agent simulation embedded in a Cultural Algorithm framework in order to explore how cultural learning and various socio-economic factors affect the population (Kobti, Reynolds and Kohler, 2003).

In this model, each household has individual characteristics that are shared among the population. In addition each household is endowed of local knowledge between a radius, determining the maximum distance that they can travel and the move frequency. In addition to that, social ties were maintained together with rules for marriage.

Reciprocal exchange strategies were added to the model in 2004 (Kobti, Reynolds and Kohler, 2004). They were representing rules for exchange food and services, using a finite state machine that represented diverse stages of each individual, from satisfied or philanthropic, or hungry to death.

Cooperation was mapped as a consequence of the beliefs of the agents who kept track of the most successful interactions favoring them over the less successful ones. Agents are initially randomly assigned to areas representing the likelihood for selection for cooperation. The model was further extended to allow for food resources and the sharing of them (Kobti and Reynolds, 2005).

5.2 Cultural Algorithms in Engineering and Management

Cultural Algorithms have also been used in engineering and management (Dasgupta and Michaalewicz, 1997; Ostrowxki, Schleis and Reynolds, 2003) in areas as diverse as decision rule-based systems (Sternberg and Reynolds, 1997) or decision tree-based knowledge discovery (Al-Sheher, 1997).

One of the examples of usage in commercial settings is for modeling evolving pricing strategies for vehicle sales (Ostrowski, Tassier, Everson and Reynolds, 2002). In that case a multi-agent system to model Oem Equipment Manufacturer market was developed using `MarketScape`, a java based agent framework designed to model economic environments. Four types of agents were present: consumer, vehicles, manufacturers and dealers. The model showed that optimal strategies were generated faster using Cultural Algorithm based programming than genetic programming.

Cultural based algorithms have also been used in reengineering. Reengineering, when done manually, is a very labour intensive approach, the use of Cultural Algorithms proved that this task can be automated (Rychtyckyj, Ostrowski, Schleis and Reynolds, 2003). This work was done using the Direct Labour Management System (DLMS), used by Ford to manage vehicle assembly.

Another example of the use of Cultural Algorithms is for learning user behaviors. A concrete case was its use for learning the correct usage of safety restraints for children (Kobti, Snowdon, Rahahman, Dunlop and Kent, 2006). In that case a *correct* knowledge, obtained from surveys, is maintained together with the knowledge in the belief space and a distance between both is calculated. An interesting experiment was done in this case with the addition of a social network component based on kinship and local neighbourhood. The research showed that learning from an expert is best because the social network made the entire system more resilient to change.

5.3 Cultural Swarms

Cultural swarming in Cultural Algorithms emerged as a consequence of the discovery that problem solving phases emerge in the belief space (Reynolds and Saleem, 2003) and its prove by performing experiments using a population represented by a Canonical Particle Swarm model (Iacoban, Reynolds and Brewster, 2003).

Reynolds and Saleem (2003) presented three phases of swarm emergence, each of them dominated by the knowledge source that is most successful in generating good solutions. The first phase is the coarse grain phase, dominated by topographical and situational knowledge. The second phase is the fine-grain phase, dominated by situational knowledge. Finally, the third phase is the backtracking phase where all sources of knowledge contribute equally. The authors presented the hypothesis that various knowledge sources were interacting at the cultural level, effectively guiding problem-solving decisions.

5.4 Other applications of Cultural Algorithms

A variety of applications have taken advantage of Cultural Algorithms besides the main categories that we have just discussed. Among them, maybe the most unexpected of them are games, from board games (Ochoa, Padilla, Gonzalez, Castro Hal, 2008; Ochoa et al., 2007) to racing games (Kinnaird-Heether, Reynolds, 2009).

However optimization, in all its forms, has certainly been a fructiferous area, such as Constraint Satisfaction Problems are especially suited to Evolutionary Programming

and therefore have been also an area for Cultural Algorithms (Becerra and Coello, 2004; Rynolds, Michalewicz and Cavaretta, 1995; Jin and Reynolds, 1999), multi-objective optimization (Coello and Landa Becerra, 2003) or even Fuzzy Clustering (Alami, Benameur and Imarani, 2007).

Finally, learning has obviously been also an area of work for cultural algorithms (Curran 2006).

6 NK Landscapes

NK landscapes were devised, by Stuart Kauffman, as a model for species fitness, mapping the states of a genome onto a scalar fitness (Kauffman and Levin 1987, Kauffman 1989, Kauffman 1993). The NK model provides a fitness landscape whose ruggedness can be *tuned* by a single parameter: `K`, this is one of the beauties of the model and part of the reason why it has been widely used.

The NK model is a model of a genome with `N` genes. Each gene has `A` alleles, mostly `A=2`, representing a binary genetic code. Each of the `N` genes constituting the genome can interact with other `K` genes. These epistatic interactions are in fact the ones responsible for the ruggedness of the landscape.

In its simplest model, when `A=2`, each gene *i* of `N` depends of other `K` genes that interact with it epistatically. Thus for each possible combination 2^{K+1} a random number is drawn from a uniform distribution between 0.0 and 1.0, this will be the contribution of gene *i* depending on the other `K` genes with whom it interacts.

The next step in the model is to define the dependencies between the genes composing the genome *N*. Three different ways have been mostly used in the literature: random assignment (Kauffman, 1993), sequential (where the successive `K` genes are assigned to `i`) (Levinthal 1997) and nearest-neighbour (where its flanking `K/2` neighbours to either side are chosen) (Kauffman 1993).

Once selected both the dependences between genes and the contribution of each gene, depending on the combination of `K+1` genes are maintained through the whole construction of the landscape.

Finally, the fitness of each possible genotype is the average contribution of its genes:

$$W = \frac{1}{N}\sum_{i=1}^{N} w_i$$

where, w_i stands for the contribution of each gene, depending on k other genes besides itself, and *W* represents the fitness of the whole genome.

The NK model has been used to represent different scenarios. We are going to follow Rivkin's (Rivkin, 200) and Levinthal's (Levinthal, 1997) approach and use it to represent the strategy of a firm and its reward in the market given its strategy.

Hence in our case, a strategy s will be represented vector $\{s_1, s_2, \ldots s_n\}$ where each component can be activated or not, producing a total of 2^N possible configurations.

The value of each strategy component s_i will depend of other `K` components. So for each 2^K possible combinations a value will be drawn from a uniform probability distribution [0..1] and the contribution of each component s_i will be assigned taking in consideration the other `K` components.

The overall value associated to a strategy s (a point in the NK landscape) is the average over the `N` value contributions.

$$P(s) = \frac{\sum_{i=1}^{N} C_i(s_i; s_{i1},...,s_{iK})}{N}$$

When `K=0` each component depends only of itself, so for every point in the landscape differing of only one component, the maximum difference in fitness will be `1/N`, resulting therefore in a smooth landscape. Contrary to that, when `K` is big, each component depends of `K` others, so given two points in the landscape that differ of only one component their maximum difference in terms of fitness will be `k+1/N` consequently resulting in a roughed landscape.

So the number of local maxima – a point with fitness greater than all its neighbours, *a.k.a*. all other points that only differ from it in one component – will depend on the value of `K`. For `K=0` only one local (and global) maxima will exist (the one whose each component of N has greater value).

On the other side, for `K=N-1` the number of local optima is very large. The probability P_s that a given strategy is a local optima is just the probability whose fitness is greater than its N neighbours, so:

$$P_s = \frac{1}{N+1}$$

and the total number of existing strategies in a landscape of parameter N is 2^N, so the expected number of local optima for one component neighbours is

$$S_1 = \frac{2^N}{N+1}$$

that give us a fairly large number. For example for a strategy of 16 components, we will have only one local maxima for `K=0` and 3,855 for `K=15`.

To get a feeling of how the various landscapes look like, it is helpful to observe the two extreme cases of the NK model, when `K=0` and when `K=N-1` (Kauffman, 1993).

Properties of *K=0* landscapes

- There is an only one local/global maximum.

- The landscape is smooth and neighbouring points (1-opt neighbours) are highly correlated. The fitness of 1-opt neighbours differs no more than $\frac{1}{N}$.
- The number of fitted decreases at each step a 1-opt local search.
- The average number of steps to reach local/global maxima is $\frac{N}{2}$, therefore $O(N)$.

Properties of *K=N-1* landscapes

If `K=N-1`, the contribution of each component - gene - depends on all the other components, resulting in a very rugged landscape.

- The expected number of local optima is $\frac{2^N}{N+1}$.
- The expected number of fitter neighbours divides by $\frac{1}{2}$ after each iteration of a 1-opt local search.
- The number of steps to reach a local optimum is expected to be in $O(\log N)$.
- The expected number of solutions to examine to reach a 1-opt local optimum is proportional to *N.*
- Starting from an arbitrary point in the landscape only a fraction of local optima can be reached by 1-opt local search. The upper bound is given by $N^{\log_2(N-1)/2}$.
- Global optima can only be reached - performing a 1-opt local search - from a small number of starting points $2^{(\log_2 N)^2/2}$.

Furthermore, for increasing values of N, the fitness of local optima decreases towards $\frac{1}{2}$. Kauffman calls this phenomenon a *complexity catastrophe* (Kauffman, 1993). In order to avoid it and be able to compare fitness values coming from different landscapes we will normalize the landscapes used in our simulations to (0..1]. Therefore global optima always will have a value of 1.

Table 1 – Number of local peaks

Panel A: Number of Distinct local Peaks

	N = 4	***N = 8***	***N =12***	***N =16***
K = 0	**1** (0)	**1** (0)	**1** (0)	**1** (0)
K = 1	**1.78** (0.74)	**2.76** (1.72)	**4.26** (2.91)	**8.86** (7.85)
K = 2	**2.36** (0.72)	**4.96** (1.71)	**11.04** (4.35)	**24.68** (12.27)
K = 3	**3.4** (1.23)	**8.3** (1.87)	**23.28** (5.86)	**66.56** (20.24)

K = 4	**12.08** (2.21)	**39.70** (7.40)	**132.34** (21.61)
K = 5	**17.08** (2.52)	**59.70** (7.58)	**223.56** (21.16)
K = 6	**22.28** (2.86)	**90.30** (8.30)	**360.48** (27.16)
K = 7	**28.80** (3.07)	**121.10** (8.45)	**546.60** (38.32)
K = 8		**158.94** (9.56)	**765.68** (44.58)
K = 9		**205.20** (8.48)	**1034.14** (41.58)
K= 10		**255.00** (12.03)	**1357.04** (37.03)
K =11		**314.68** (10.76)	**1736.54** (35.80)
K =12			**2173.72** (44.94)
K =13			**2670.70** (41.92)
K = 14			**3221.34** (38.93)
K = 15			**3857.34** (37.29)

Panel B: Proportion of local Peaks vs. points

	N = 4 *16 points*	***N = 8*** *256 points*	***N =12*** *4,096 points*	***N =16*** ***65,536 points***
K = 0	**6.250%**	**0.391%**	**0.024%**	**0.001%**
K = 1	**11.125 %**	**1.078%**	**0.104%**	**0.013%**
K = 2	**11.750 %**	**1.937%**	**0.269%**	**0.038%**
K = 3	**21.250 %**	**3.273%**	**0.568%**	**0.102%**
K = 4		**4.719%**	**0.969%**	**0.202%**
K = 5		**6.672%**	**1.457%**	**0.341%**
K = 6		**8.703%**	**2.205%**	**0.550%**
K = 7		**11.250 %**	**2.956%**	**0.834%**
K = 8			**3.880%**	**1.168%**
K = 9			**5.010%**	**1.578%**
K= 10			**6.226%**	**2.071%**
K =11			**7.683%**	**2.650%**
K =12				**3.317%**
K =13				**4.075%**
K = 14				**4.915%**
K = 15				**5.882%**

For each level of N and K fifty landscapes were created and local peaks – within 1 mutation distance – were counted exhaustively. Numbers shown on panel A, corresponds to the mean number of local peaks found and the standard deviation (below). Panel B shows the proportion of local maxima respect to the total number of positions in the landscape for every `N`, `K` value.

As K increases becomes easier for the agents to get trap into local maxima, making more difficult an incremental improvement strategy.

In NK landscapes the number of local peaks grows exponentially as K grows (Table 1) and their grow ratio increases with N (Figure 1). But as N increases, a second regime can be observed, the ratio of local maxima versus total number of points for K maximal (K = N−1) is 1/N+1 and this percentage decreases with a ratio 1− 1/N as N increases (Table 1). So, local maxima become perceptually scarce as the landscape grows.

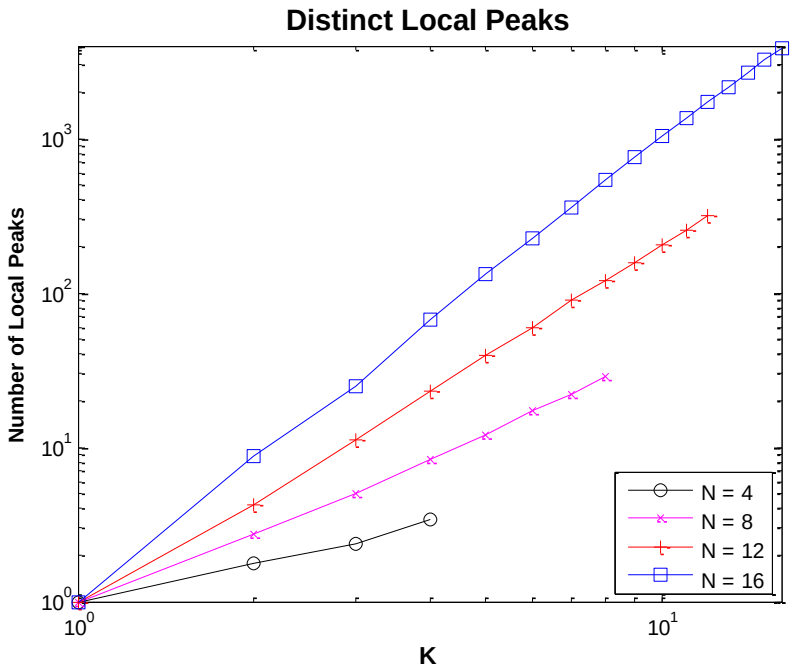


Fig 1. For each level of N and K fifty landscapes were created and local peaks – within 1 mutation distance – were counted exhaustively. A figure with a log/log representation of the results is show for every level of N.

The number of local peaks grows exponentially with K, but as N grows they are a fraction of the total 2^N points.

For NK landscapes as N and K grow two regimes can be observed. As K grows for a certain N, the landscape becomes rugged and it is easier for the agents to get trapped into local maxima, resulting in a lower success for an incremental improvement strategy.

On the other side, as N grows, the size of the landscape grows faster than its number of local maxima, providing larger travelling spaces where incremental improvement agents can bet on more strategies without getting immediately trapped into a local peak.

One very well known characteristic of NK landscapes is that local peak height decreases as K increases (Table 2) effectively flattening the landscape.

This effect can be avoided normalizing by $\sqrt{N}$ if desired, but in many cases is not relevant. For example, if we are using a greedy search strategy where the highest peak is the one selected, the relative height of peaks does not have any effect. On the contrary, if our agents use a fitness strategy, where peaks are selected on the basis of their relative height (or the difference in height with respect to our position) then differences in relative peak height will result in different probability distributions.

Table 2 – Peak Height

	N = 4	***N = 8***	***N =12***	***N =16***
K = 0	**0.669** (0.04)	**0.669** (0.02)	**0.668** (0.07)	**0.670** (0.04)
K = 3	**0.649** (0.09)	**0.663** (0.06)	**0.661** (0.05)	**0.657** (0.04)
K = 5		**0.654** (0.06)	**0.657** (0.05)	**0.659** (0.04)
K = 7		**0.649** (0.06)	**0.647** (0.05)	**0.648** (0.04)
K = 9			**0.647** (0.04)	**0.645** (0.04)
K =11			**0.638** (0.05)	**0.639** (0.04)
K =13				**0.636** (0.04)
K = 15				**0.630** (0.04)

One hundred landscapes (random interactions) were created for different levels of N and K and local peaks were – within 1 mutation distance - examined exhaustively. The mean peak height is shown in bold and below the standard deviation of the average is shown in parenthesis below each data point.

The average height of local maxima falls as K increases, or as the landscape becomes more complex. This effect must be taken into account in simulations and can be corrected normalizing by $\sqrt{N}$.

Is difficult to assess if this fall of peak height reflects in any way a real world economic characteristic where more complex and interrelated strategies result in lower rewards (fitness) than simpler ones. Simpler strategies have usually lower entry barriers that result in more crowed points with higher concurrence levels than strategies with a higher degree of epistatic interactions. These differences in concurrence levels due to lower entry barriers make it difficult to assess if in real economies, a higher level of interaction between strategy components result in lower local peak fitness. Either way,

this effect is not relevant to the simulations and the work carried out in this paper, but must be taken into consideration for future work.

The distribution of local maxima in NK landscapes is a relevant characteristic that will clearly affect *how* agents carry out search. We will use the Hamming distance (Hamming, 1986), known as the difference in information components, strategy components in that case, between to bit strings (strategies in that case). So if two strategies difference in 1 component, its Hamming distance will be 1.

As we can see in Table 3, the average distance between local maxima approaches N/2 as K increases which clearly corresponds with the random distribution upon which the NK spaces are built. One striking factor, already reported in (Kauffman, 1993) is the fact that for K low (K=1, K=2) peaks are closer, approaching N/3. This could be consistent with many real world situations where for very focused sectors, one or two components dominate their strategy (low cost flights, low cost retailing, … both with logistics and cost being the dominant strategy players) producing dominant strategies that are close together because they share the main dominant factors.

Table 3 – Distance between Local Maxima

	N = 4	***N = 8***	***N =12***	***N =16***
K = 0	**0** (0) 0 (0)	**0 (0)** 0 (0)	**0 (0)** 0 (0)	**0 (0)** 0 (0)
K = 1	**1.49** (0.14) 1.28 (0)	**2.66 (0.47)** 1.56 (0)	**3.58 (0.17)** 1.88 (0)	**5.54 (0.08)** 1.88 (0)
K = 2	**2.49 (0.23)** 2.05 (0.29)	**3.78 (0.51)** 2.09 (0.14)	**4.01 (0.29)** 2.00 (0)	**5.89 (0.07)** 2.00 (0)
K = 3	**2.20 (0.19)** 1.95 (0.04)	**4.05 (0.26)** 2.04 (0.06)	**5.65 (0.30)** 2.17 (0.27)	**5.96 (0.11)** 2.00 (0)
K = 4		**3.95 (0.23)** 2.03 (0.08)	**5.74 (0.27)** 2.09 (0.18)	**6.16 (0.25)** 2.00 (0)
K = 5		**3.98 (0.21)** 2.04 (0.13)	**5.78 (0.19)** 2.08 (0.22)	**6.82 (0.47)** 2.00 (0)
K = 6		**3.91 (0.17)** 2.00 (0.01)	**5.77 (0.14)** 2.02 (0.05)	**7.74 (0.14)** 2.11 (0.26)
K = 7		**3.88 (0.16)** 2.00 (0)	**5.88 (0.12)** 2.05 (0.17)	**7.70 (0.14)** 2.01 (0.11)
K = 8			**5.83 (0.10)** 2.00 (0)	**7.83 (0.08)** 2.04 (0.14)
K = 9			**5.86 (0.08)** 2.0 (0.03)	**7.85 (0.06)** 2.04 (0.15)
K= 10			**5.83 (0.08)** 2.00 (0.02)	**7.89 (0.05)** 2.01 (0.05)
K =11			**5.88 (0.07)** 2.00 (0)	**7.92 (0.04)** 2.01 (0.05)
K =12				**7.89 (0.03)** 2.00 (0.04)
K =13				**7.92 (0.03)** 2.00 (0)
K = 14				**7.93 (0.02)** 2.00 (0.01)
K = 15				**7.96 (0.03)** 2.00 (0.02)

For each level of N and K fifty landscapes were created and local peaks – within 1 mutation distance – were counted exhaustively. Once found the average (above in bold) and the minimum (below in normal) distance between each local maxima and all the others was obtained using the Hamming distance as a measure (number of bits – strategy components – different from one strategy with respect another one. Result corresponds to the average and standard deviation (between parentheses) of the 50 landscapes tested.

We can see how as K increases, mean distance approaches N/2, beginning for K=1 with approximately N/3. A very interesting phenomena is that the minimum distance is always close to 2, but for K=1 that is even closer. That means that given any peak we can find another one really close, or that any peak carries information relevant not only to himself but to another one.

What is also interesting on the data of Table 3 is that the minimum distance between two peaks – the distance to the closest peak – is always around 2 (except for K=1 that is even lower). That means that given any peak we can find another one at a distance of two or lower in any case (note that standard deviation is always 0.2 or lower). This is also consistent with the real world experience that not all factors weight the same and some are dominant, around these dominant factors we can find several strategies resulting of the interchange of equally weighted, let us call them, secondary factors.

But this fact has also an important consequence for the collective behavior of the agents and their collective learning process. That is that the fact of being in one strategy local maxima carries information valid to other points, information that can potentially lead to find a different peak.

Table 4 – Mean Walk Lengths to Local Optima

	N = 4	*N = 8*	*N =12*	*N =16*
K = 0	**1.996** (1.005)	**4.013** (1.418)	**5.999** (1.741)	**7.979** (2.003)
K = 1	**1.682** (0.977)	**3.467** (1.379)	**4.976** (1.624)	**6.803** (1.876)
K = 2	**1.354** (0.869)	**2.813** (1.305)	**4.292** (1.603)	**5.773** (1.892)
K = 3	**1.159** (0.806)	**2.448** (1.266)	**3.705** (1.572)	**5.060** (1.838)
K = 4		**2.077** (1.111)	**3.214** (1.439)	**4.383** (1.753)
K = 5		**1.821** (1.041)	**2.801** (1.342)	**3.797** (1.614)
K = 6		**1.589** (1.003)	**2.524** (1.329)	**3.401** (1.515)
K = 7		**1.418** (0.961)	**2.233** (1.181)	**3.077** (1.400)
K = 8			**1.987**	**2.795**

	(1.081)	(1.326)
K = 9	**1.832**	**2.559**
	(1.047)	(1.302)
K= 10	**1.643**	**2.340**
	(1.029)	(1.224)
K =11	**1.510**	**2.154**
	(1.010)	(1.130)
K =12		**1.973**
		(1.064)
K =13		**1.804**
		(1.025)
K = 14		**1.684**
		(1.010)
K = 15		**1.560**
		(1.005)

For each level of N and K one hundred landscapes were created and 100 agents released in each landscape. Incremental walks lengths performing a greedy search were recorded. The average walk length is shown in bold and the average of the standard deviations of walk lengths for each landscape is shown between parentheses.

Mean walk length begins with N/2 and as K increases approaches 1 reflecting an increasingly rugged landscape. Mean walks lengths have also implications in how long a system will settled down and how much time will be available for diffusion between agents

Correspondence to the real world is in that case quite direct. Complex, very interrelated strategies make the economic agents very prone to get trapped into local maxima, many times forgetting key factors that are suddenly rediscovered by a new competitor. We have many examples of that where in mature sectors like flight companies or banks, complex strategies were carried out and companies craved niches around them until a new competitor simplified the strategic approach and was able to attain a higher maximum. But the shortening of walks due to the increasing complexity of the environment has also two important implications in a dynamic system where economic agents are interrelated and interdependent.

The first one is *diffusion*. A shorter path results inevitably in less time available to diffuse discoveries and to assimilate them. The second one is *equilibrium*, with a shorter path length equilibrium can potentially be reached earlier given the fact that the system will settled down when all agents have attained their maxima with regard of the positions of other agents. We can see, and we will present in the next sections several models of collective discovery where a group of agents guided only by local information and their greedy interest will in fact collaborate in the discovery of the highest peaks in the NK space attaining a situation that will be globally better than the one achievable without interrelation.

This is a process of collaborative pattern learning, where the agents will try to discover patterns in the NK space construction that result in a higher fitness, these are nothing else than the ones that have the highest value in the `K+1` gene table used originally to obtain the fitness of each point.

6.1 Greedy Algorithms for the NK Model

Given the configuration of an NK landscape, it is possible to walk through it by changing one or more components at a time. If the choice of components follows a greedy rule, then the approach can be classified as a greedy heuristic for NK landscapes.

Greedy heuristics are widely used in NK landscapes, mostly as a baseline upon which to compare other strategies because they easily get trapped into local maxima, especially in the case of 1-opt where only 1 component is investigated. Having an existing solution as a starting point, a new solution is built by maximizing a gain function $g(i,v):\{1,\dots,N\}\times\{0,1\}\to\mathbb{R}$ with $g(i,v)$ denoting the fitness of the position obtained by setting the component *i-th* component of the position in the landscape to the value v.

We define the gain function as the difference between the fitness of a partial position y with the *i-th* component set to v and the fitness of a partial position x with the component unspecified,

$$g(i,v) = f^p(y) - f^p(x) \text{ with}$$

$$y_j = \begin{cases} v\ ,if\ i=j \\ x_j\ ,otherwise \end{cases}$$

The fitness of a partial solution $f^p$is defined as the average fitness of all positions matching a template, e.g. assume a partial position x, where $x = (0,1,*,1,*\ 1)$ where * denotes indifference. Then, the fitness f^p of x will be the average fitness of four positions, namely (0,1,0,1,0,1), (0,1,0,1,1,1), (0,1,1,1,0,1) and (0,1,1,1,1,1).

However, when performing local search neighbouring positions can be reached by flipping a single component of the actual one. Therefore,

$$g(i,v) = f(y) - f(x) \text{ with}$$

$$y_j = \begin{cases} 1-x_j\ ,if\ i=j \\ x_j\quad\ \ ,otherwise \end{cases}$$

In terms of implementation, local search can be facilitated by using dynamic programming techniques (in many cases more than one agent will walk through a point previously calculated, especially when climbing basins, providing an opportunity for reuse) or by maintaining a gain vector with the partial values of the components because many will be unaffected by the change (especially when *K* low).

The implementation of 1-opt local search is straightforward and is presented in Algorithm 1. There an agent flips its position components maximizing its fitness by using the gain function previously described.

Algorithm 1 Local Search 1-opt

Input: x - *position in an NK landscape*
Input: L - a vector representing the NK landscape
Output: x - *resulting position*

function *LocalSearch1opt :* (x, L) → x

calculate gains of x, $g_i \; \forall \; i \; in \; \{1,...n\}$ *-gains changing 1-opt*

repeat
find *k* with g_k max
if $g_k > 0$
then
$x_k = 1 - x_k$ *-flip the component*
find *k* with g_k max *- update gains*
end
until $g_k \leq 0$
return x *- position after performing a local search 1-opt*
end

Similarly we can extend the last 1-opt algorithm to a k-opt local search, allowing the simultaneous change of 2 or more components instead of only one. The extension is completely straightforward if we enumerate all possible positions changing k components. However, its number grows exponentially, concretely following 2^k, making difficult its complete exploration as k grows. Therefore it will be useful to devise an algorithm that could allow even a partial exploration. Fortunately we can easily think of many of them and for the sake of the example we will discuss one of them.

Possible the simplest way to approach the problem is by limiting the search space of solutions to explore and the easiest way to do it is following a greedy approach as we are doing in the previous one. Therefore we will pick up a subset of components based on their partial contribution to the overall fitness. As the lector might guess, this is not by far optimal and this approach will increasingly fail in reaching the global optima as the number of interactions between the components grows, or put it in another ways as K grows.

Algorithm 2 Local Search k-opt

Input: x - *position in an NK landscape*
Input: L - a vector representing the NK landscape
Output: x - *resulting position*

function *LocalSearchkopt :* (x, L) → x

calculate gains of x, $g_i \; \forall \, i \; in \; \{1,\ldots n\}$ -*gains changing 1-opt*

repeat
G=0, G_{max}=0, x_{old}=x, S={1, ...n}
find *j* with g_j max
while $S \neq \emptyset \; and \; j \neq \; null$
G = G + g_j
$x_j \; = \; 1 - \; x_j$
if $G \; > \; G_{max}$
then
$G_{max} \; = \; G$
$x_{max} \; = \; x$
end
$S \; = \; S \; - \{j\}$
calculate gains of x, $g_i \; \forall \, i \; in \; \{1,\ldots n\}$
end

if $G_{max} \; > \; 0$
then
$x = x_{max}$
else
$x \; = \; x_{old}$
end
until $G_{max} \; \leq \; 0$
return x - *position after performing a local search k-opt*
end

7 Model-Based Learning

In their evolution, Cultural Algorithms have suffered a process of sophistication in almost any of representative characteristic, from the knowledge representation in the shared space to the selection and acceptance functions or the evolutionary process that takes place among the population of agents.

This sophistication allowed tackling types of problems that, at first sight, could appear to be far away from their reach. This is not only the case of the family of applications devoted to optimization, but also the one who addresses elaborated simulations of emergence. However, it could certainly be interesting to revisit the initial concepts where Cultural Algorithms have their roots. This exercise could provide not only a basis for more realistic social models when addressing the evolutions of human collectives but possibly insights that could open new understandings and new possibilities in the use of Cultural Algorithms.

A set of these initial concepts that lay behind Cultural Algorithms are related to the selection function. How do we select the individual that is going to serve as the basis for inferring theory? Could it have large implications in terms of the amount of information that should be processed, sample size and the complexity of this process?. All these, could have significant implications in terms of applicability and the realism of simulations.

Evolutionary psychology provides some clues on *how* the selection function could work. In fact, once individuals begin to learn from others, it is obviously wise to be selective, preferentially pay attention to, and learn from those who are highly successful or particularly skilled subjects (Henrich and Gil-White, 2001), being the probability of imitation correlated with the observed difference in payoffs (Apesteguia, Huck and Oechssler, 2005; Schlag 1998, 1999).

However, in this form of model-based cultural learning assessing from whom to learn is not a straightforward task. This is why we rely on cues such as competence, success and prestige and we devise and build social mechanisms that promote their emergence.

Therefore, in a social setting that makes use of competence and success signals, highly skilled individuals will be in high demand and a selection pressure leading to a deference mechanism will appear (Gurven, 2004). Naive entrants can therefore look at the existing pattern of deference in order to solve the costly information problem. Consequently, solving the problem of from whom to learn is reduced to aggregating information from the distribution of deference.

Therefore, the mechanism of selection, from the point of view of evolutionary psychology, is rooted on the selection of a few best and its use as a model.

Similarly to the selection function, the adoption function drives to a large extend the workings of Cultural Algorithms. Again, we hope to find some clues revisiting what we know of the way social communities incorporate and adopt knowledge. A well-known social mechanism to perform this aggregation is *conformist transmission* (Boyd and Richerson, 2005; Henrich and Boyd, 1998), that can be described as copy the majority or copy what the majority understands as best practices.

Moreover, conformist transmission is known to be the best route to adaptation in information poor environments (Henrich and Boyd 1998; Kameda and Nakanishi, 2002). Hence, when individual ambiguity due to low accuracy of the information obtained through individual learning increases, so does the reliance on conformist transmission (McElreath, Lubell, et al. 2005).

The last element that we should revisit is the knowledge representation that has evolved in many ways, some of them highly sophisticated. Examining the roots of the mechanism for abstraction and representation of knowledge, could again be valuable for elaborating models more closely related to our current knowledge.

In this case, probably the strand of research more relevant comes from Cognitive Science that sees pattern matching as central to reasoning and learning. The two predominant cognitive models, the connectionist that sees the brain as a computer (Hebb, 1949) and the evolutionist (Dosi, Marengo and Fagiolo, 2003) that postulates a process based in somatic selections, view the brain as employing pattern-based reasoning, rather than abstract logical reasoning, which is essential for explaining how we make choices in a world of uncertainty. "*Thinking occurs in terms of synthesized patterns, not logic and for this reason it may always exceed in its reach syntactical or mechanical relations*" (Edelman and Tonini, 2001, p. 152).

This pattern based reasoning mechanism also influenced Institutional Economics. In fact, when dealing with the process of economic change, Douglas C. North, indicates:

> "Much of learning comes from absorbing and adjusting to subtle events that have an impact on our lives, incrementally modifying our behavior ever so slightly. Implicit knowledge evolves without ever being reasoned out. In fact we are relatively poor at reasoning compared to our ability to understand problems and see solutions. We are good at understanding and comprehending if the issue is sufficiently similar to other events that have happened in our experience. Ideas to far from the norms embodied in our culture cannot easily be incorporate in our culture. Ideas are adopted if and when the share a kind of cohesion that does not take them too far from the norms we possess. Pattern matching is the way we perceive, remember and

> comprehend. This is the key to our ability to generalize and use analogy. This ability makes us good not only at modeling "reality", but also at constructing theories in the face of real uncertainty." (North, 2005, p. 26-27).

These insights provide some help and some clues on ways to characterize an alternative model for Cultural Algorithms.

Such a model will be based on selecting the best performing individuals form a community on the basis of a public signal. From them, patterns of action supposedly producing these high results will be extracted and based on their popularity incorporated and adopted by the individuals in the social community still not endowed with them.

It is a model, in many aspects, more simple than many of the ones reviewed, but because of this simplicity, we will probably be more able to infer conclusions on its workings and the relative importance of its different elements.

Several research questions arise naturally from the simple observation of the model. Among them, we can mention:

1) What is the optimal size of the sample to be used in the selection process?
2) How comprehensive should be the patterns to be inferred?
3) How extensive needs to be the process of adoption in order to obtain an optimal or near optimal result?

In addition to these questions that relate to a particular aspect of Cultural Algorithms, a general one arises: the dependency of these variables on the type of information being treated.

Certainly we can hypothesize that very simple types of knowledge could need smaller sample sizes, while more complex ones will require larger samples. Therefore, we should frame the previous research questions with the next one:

4) Do sample size, comprehensiveness of patterns and the adoption process depend on the complexity of the informational landscape to be traversed?

7.1 Modeling Pattern-Based Strategy Search

The first two elements that should be first elucidated in order set the basis for our model are the type of knowledge representation that is going to be used and its mapping to a fitness function.

In the second case, the choice looks pretty obvious. In fact if we need to relate the rest of components to complexity, as we mention in the research question number four, our almost sole choice is the use of NK spaces for the mapping function. NK spaces allow the easy control of two key variables of the landscape: its size, using the parameter N, and its complexity, using the parameter K.

The election of NK spaces as mapping function already determines, at least to some extent, the selection of the first one: knowledge representation. NK spaces use a simple binary where each component represents a knowledge component. Therefore, in this case there is a lack of representation for knowledge hierarchy on in general any type of knowledge structure. We certainly could have incorporated some type of structure when representing the knowledge of the agents, however, in order to answer the proposed research question this structure would probably not help and we will therefore complicate the model with unnecessary elements for its purpose.

In addition to that, if we aim to use this model for comparison, a baseline is needed. Because agents move in an NK landscape, the optimization function is naturally mapped as search. Incremental search that captures the notion that agents have a bounded visibility by limiting the scope to a limited number of components, one in our case, is clearly a baseline against whom to compare the performance or Cultural Algorithms.

Therefore, in our model an agent is endowed with a vector of binary choices representing unique components, $a_s = \{s_1, s_2, \ldots, s_n\}$ which can take a value of either 0 or 1. This vector is directly mapped to a position in the NK space, and through this mapping, a fitness value, corresponding to the fitness of this position in the NK space, for each agent is obtained. Agents move through the NK space performing individual search strategies either incremental or a Cultural Algorithm based on patterns.

Incremental search is implemented by performing a hill-climbing algorithm where agents are allowed to freely move between 1-component ranges seeking the *best* fitness. A pattern is a group of binary choices composed by one or more components. Therefore all possible patterns of length l can be described as combinations of $C_N^l \, 2^l$.

For example given N=4, $s=\{s_1, s_2, s_3, s_4\}$, the possible patterns that could be derived from all possible combinations of length two of the two components of the strategy s are,

s_1, s_2 **s_2, s_3** **s_3, s_4**
s_1, s_3 **s_2, s_4**
s_1, s_4

each of them can take any of the possible 2^l combinations of s_i, $s_i \in \{0,1\}$, therefore

s_1	s_2
0	0
0	1
1	0
1	1

Agents aim to discover the best patterns in the population of agents in order to incorporate them into their strategy and improve their fitness.

The belief space is built by performing the following functions:

1. Monitoring the position of the agents in the landscape at each iteration
2. Ranking the agents according to their fitness and select a percentage of best performers (governed by a parameter ρ, $\rho \in (0..1]$) as best cases
3. Collect the patterns among these best cases and rank these patterns according to one of the following strategies

    a. the average fitness of the pattern. Representing a rational agent with complete and exact knowledge of the fitness of the rest.
    b. their popularity. Mapping the mechanism of conformist transmission.

For the sake of the example let's assume that there are only 4 agents with N=4 and ρ=1, thus we consider the entire population as best cases.

		0 0	***0 1***	***1 0***	***1 1***	
	s_1, s_2	0	1	2	1	
a_1 = { 0 1	**s_1, s_3**	0	1	0	3	1 0 }
a_2 = { 1 0	**s_1, s_4**	1	0	2	1	1 0 }
a_3 = { 1 1	**s_2, s_3**	0	2	0	2	1 0 }
a_4 = { 1 0	**s_2, s_4**	1	1	2	0	1 1 }
	s_3, s_4	0	0	3	1	

following the previous table, the ranking of patterns according to their popularity as best cases will be established as

Strategy components	*Pattern*	*Number of Cases*
s_1, s_3	1 1	3

s_3, s_4	1 0	3
s_1, s_2	1 0	2
s_1, s_4	1 0	2
s_2, s_3	0 1	2
s_2, s_3	1 1	2
s_2, s_4	1 0	2
s_1, s_2	0 1	1
s_1, s_2	1 1	1
s_1, s_4	0 0	1
s_1, s_4	1 1	1
s_2, s_4	0 0	1
s_2, s_4	0 1	1
s_3, s_4	1 1	1

building that way an structure of patterns p as a set of tuples $<c_i, v_i, o_i, a_i, f_i>$ where c_i stands for the set of components that the pattern considers, v_i for their binary value, o_i for the number of occurrences in the population of best cases considered (in case that we use the popularity classification), a_i for the average fitness of each pattern (in case that we use the average fitness classification) and f_i for the maximum fitness of the pattern, taken as the one of the best performing agent endowed with that pattern.

Moreover, agents are endowed with a memory $m_a = \{s_{a,1}, \ldots . , s_{a,n}\}$ which contains all strategies known by the agent because they have been previously applied. Therefore agents will only apply strategies that are either unknown or being known lead to a fitness higher than the current one.

The model also needs an influence function, again we tried to keep close with what a realistic behaviour could be, introducing elements like the non adoption of patterns conducing to a lower fitness and endowing the agents with a certain size of memory in order not to repeat previous mistakes. This influence function is presented in Algorithm 3,

Algorithm 3 Influence function

Input: agent.{s,f} - *agent's strategy (agents.s) and fitness (agents.f)*
Input: P - set of patterns in the belief space
Output: agent.{s,f} - *agent with adopted pattern*

function *ApplyBestPattern :* (agent, P) → agent

P' = sort(P, v, 'descend') - *Sort patterns by # votes*
for each p ∈ P'

```
            if  p.f ≤ agent.fitness - pattern has lower fitness than the agent
             then
                    continue
            end

            if   p ⊂ agent.s   -pattern exist in the agent's strategy vector
             then
                    continue
            end

            xs= apply(p, agent.s)

            if xs in agent.mem and xs.fitness ≤ agent.fitness
             then
                    - strategy has been applied and has a lower fitness
                    continue
             else
                    agent.s = apply(p, agent.s)          - apply pattern
                    agent.mem = include(agent.mem, agent.s)
                    - include in memory
                    return agent
            end
    end
    return agent
end
```

Therefore, in every round of the simulation each agent considers the existing selection of patterns sorted by popularity (or alternatively by the average fitness of the agents endowed with them) until one pattern if found and applied or all patterns have been discarded by any of the following conditions:

1. The knowledge vector of the agent already includes the pattern.
2. The *best* performer agent with this pattern has fitness lower or equal than the one of the agent.
3. The position resulting from applying the pattern to the strategy of the agent is already in the memory of the agent with a lower or equal fitness (the appliance of the pattern does not result in a gain for the agent).

As we can observe, in this case, agents do not adopt the patterns blindly, on the contrary they are endowed of cognitive capacities, mimicking human cognitive capacities that make them actively participate in the exploratory process. We can portray this participation in three distinctive ways.

First, they contribute to pattern discovery with their knowledge vectors that constitute the raw materials for obtaining the best patterns.

Secondly, they perform a selection process. Patterns, in fact, are not applied blindly by the agents. Agents select patterns by comparing the fitness of the best performer agent endowed with the pattern with their own fitness and adopting it only in the case that the maximum fitness attainable by adopting the pattern is higher than the one that they already have.

Therefore, we assume that fitness estimation, representative of the pattern, is public. This aims to model the common scenario where best practices are exemplified and presented with the aid of a real stellar performer. We can find multiple examples of this widespread practice, for example in the economic world, stellar performance is mostly public and the characteristics of the best performers highly publicized.

This selection process that comes naturally, product of common sense, in the real world: a practice whose results seem to be worse than the ones obtained by practices in use, is not commonly applied, has important implications in the selection process, because that way, popular but non-performing practices are rapidly discarded. This is the case, for example, of the initial states of the simulation where the strategies of the agents are randomly picked and when their level of popularity does not correspond to a selection process socially evaluating their performance but to this initial random arrangement.

The second mechanism of selection with which the agents are endowed is their memory. As we will discuss later in this chapter, the size of the memory has important implications on the resulting performance of the system. However, the main effect of memory is to prevent the agents to fall into traps and to avoid circular loops.

This, of course, has the undesired effect of averting some jumps that have to find an alternative route. For illustration, let us assume positions `a`, `b` and `c` in the landscape, with fitness f_a, f_b and f_c respectively, such as $f_b < f_a < f_c$, and f_c optimal maxima. Given that a path exist between `a` and `c` through `b`, agents will have to experience a reduction in their fitness when reaching position `b`. If `b` is in the memory of the agent, this fact will prevent that movement and position `c` could not be attained through this path. Fortunately, an alternative route to `c` will be found in most of the cases (with lower probability as `K` increases because the diameter of the basins of attraction determining the different ways to access the maxima, decrease). In addition to that, as we will discuss later, the process of discovery progresses through successive generations of alleged best practices, rendering important parts of the memory obsolete.

Memory has therefore this dual effect of avoiding traps, but at the same time limiting discovery. In the business literature we can find many examples of this paradox, especially in areas related to change and change management.

Finally, there is a third mechanism of exploration driven by the agents: a variation of crossover. Agents generate potentially new strategies by adopting new patterns. Let us, for the sake of the example, consider an agent with knowledge vector 01110 (N=5) adopting pattern p=< component={2,4}, value={0,0}>. Once adopted the agent will hold vector 00100 as a result.

However, even if pattern adoption drives discovery, it is also true that the progressive adoption of the most popular patterns reduces the level of heterogeneity among the population of agents. As we will discuss later on, this effect can be counteracted by a proactive exploration on the side of the agents.

7.1.1 Aggregating Information: Average Fitness vs. Conformist Transmission

Conformist transmission is based on popularity and this can obviously raise some skepticism around its goodness as an indicator of the performance of a pattern. Therefore, in order to assess its level of performance as a measure we chose two additional measures for comparison.

First, and as a baseline, incremental search, where agents engage in a greedy search of 1-component range. And secondly, instead using *popularity* that is clearly an inefficient proxy for high fitness (especially in the early stages, where distribution is basically random), we can try to cover for this inefficiency by using a statistical measure of the performance of a pattern; the most obvious one is the average fitness of a pattern. We then compared the performance of a population of agents when using these two different functions for pattern ranking: popularity and its average fitness, against incremental search.

In figure 2 we can appreciate the results of this exercise. There the best 25% performers among the population of agents have been selected as best cases from where to infer patterns (we will see later on that these results are consistent for a lower set of best cases). We can observe that the use of the average fitness or popularity (number of agents endowed with a certain pattern) lead to results roughly equivalent. Results that are however, much better than the ones obtained by the use of incremental search, where agents easily fall and get trapped into local maxima.

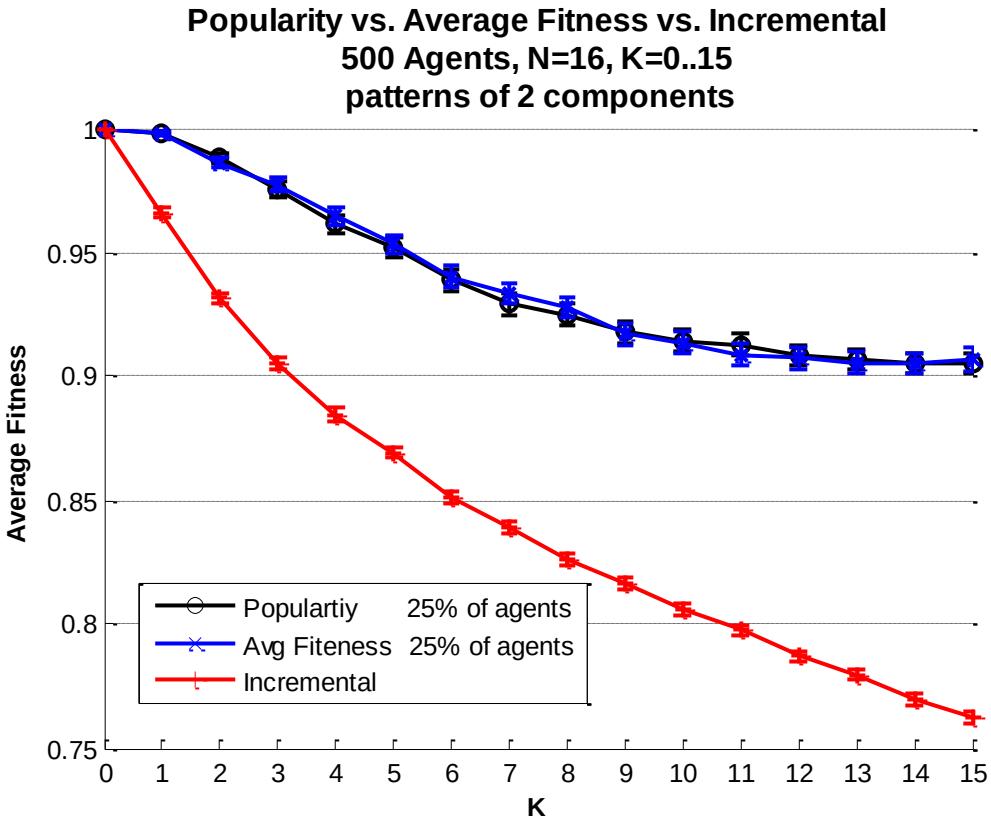


Fig 2. Comparing three strategies: Popularity, Average Fitness and Hill Climbing

500 agents performing patterned search are released in a NK landscape, 100 tries are performed and results are averaged. Three different approaches are used for selecting the pattern that agents will apply: a) the popularity of a pattern among a set of Best Performers, b) the average fitness of the agents endowed with that pattern and c) incremental search (hill climbing) where agents can change one component at a time . Error bars correspond to the standard error of the mean.

We can observe that popularity is a heuristic as good as average fitness, although less costly and more easily observable. Both of them produce better results than performing incremental search with their distance increasing as complexity does.

We also have to consider the operational efficiency of the mechanism. Compared to obtaining an accurate measure of the average fitness, popularity is easy observable and very cheap to assess because only involves counting. This is even truer in cases where the uncertainty of the economic or technological environment poses greater challenges to an accurate assessment of fitness, in these cases, popularity among best cases continues to be an easy, simple and fast measure.

As we have seen, results of popularity as a heuristic are similar to using average fitness for a set of cases involving a 25% of the population of agents and both clearly superior to incremental innovation (hill-climbing).

Finally, let us discuss the mechanisms with some more detail.

Proposition 1. Accuracy of ordering depends on the size of the set of best cases (in the case of average fitness ordering) and on the selection process.

Let us consider first the case of ordering by average fitness. Following (1) we can consider that the fitness f_i of point i, as divided in two parts, $f_i = f_{i \in p} + f_{i \notin p}$, the fitness corresponding to components included in the pattern p, $f_{i \in p}$ and the fitness of the components not belonging to p $f_{i \notin p}$. As the size of the set of best cases considered increases, this second part will tend to reflect the mean of these components because the fitness of the components are drawn from a uniform random distribution. Equally, the first part will tend to reflect the quality of the pattern. However, if the sample selected does not correspond to a random sample of the agents (as in the initial state), ordering will be driven by the selection process.

This is also the case for popularity, which initially is completely random and later on is solely driven by the selection process.

Proposition 2. Pattern selection is a social process - between the agents and the belief space - affected by the quality of the patterns and the quality of the ordering, which is more relevant as the average fitness of the set of best cases considered decreases (larger sets of best cases).

Adoption of a pattern by an agent is governed by the ranking of the patterns, the offering of patterns and the agent's fitness. In fact the condition for adoption is $P_{adopted} = P \mid P.f > a.f \; \forall \; a \in Agent$, therefore only patterns with higher representative fitness will be adopted. However, ranking can affect pattern adoption, *e.g.* given two patterns p_i and p_j with $p_i.f > p_j.f$, a certain ordering can effectively mask p_i preventing its adoption and ultimately disappearing of the set of best cases being replace by the inferior pattern p_j.

In order to find out the relative importance of ranking, let us consider de case where only the best patterns are selected. Obviously ranking in that case will be completely superfluous. On the contrary, if heterogeneity is very high, the number of different patterns will also be higher (sets with more noise and smaller differences in ranking), in that case ranking quality will be very relevant in order to prevent masking. Therefore the importance of the ranking is directly correlated with the level of quality of the set of best cases, so their fitness. Nevertheless, the offering and ranking of patterns in the next iteration will depend on the ones selected by the agents in the previous one together with crossovers. In that sense pattern adoption is a social process where agents play a fundamental role by choosing to adopt and selecting patterns on the base of their own fitness.

8 Less is More

In the model being presented, the knowledge of the agents is being generalized in the belief space by inferring patterns from the knowledge vectors of the agents. The question that immediately arises and corresponds to our first research question is about the size of the sample that is implemented in the present model by the ρ parameter.

What is the most appropriate magnitude for ρ in order to achieve the optimal results? Will larger sample sizes conduct to more exact results or, on the contrary, small ones – therefore less costly to observe – will produce better or roughly equivalent results? As we stayed in our research question four, these questions are situated in the framework of complexity that naturally comes from the use of NK spaces as the landscape that agents use to map their fitness function.

Figure 3 shows how different magnitudes of ρ perform. Agents are ranked following their fitness and a sample of best performers is used in order to distil patterns. We use sample sets that range from the best 2% of the agents ranked by their fitness to the 100% of them, the whole population.

There, we can observe how average fitness as a heuristic performs better for small sample sizes and how its performance decreases as the sample set of agents considered, enlarges. Obviously, the differences are significant for low and medium levels of complexity, however when complexity is too high, patterns encounter difficulties deciphering the landscape and its performance decreases.

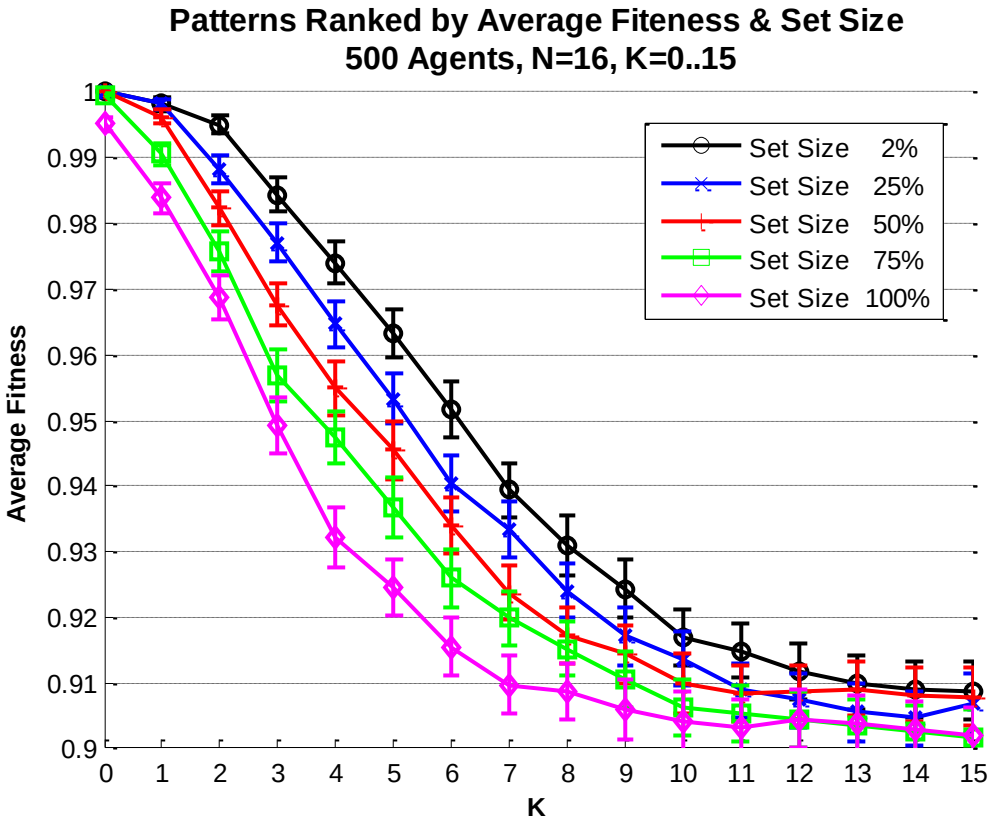


Fig 3. Variation in performance for different set sizes using average fitness for pattern ranking

A set of 500 agents performing patterned search with patterns ranked on the average fitness of each pattern, are released in a NK landscape, 100 tries are performed and the results averaged. The experiment is repeated for different sizes of the set of best cases, ranging from 2% (10 best cases) to 100% (500 best cases). The mean and its standard error are presented.

We can observe how selecting a smaller sample set produces better results than a larger one(to an extent, as we will see later the number of cases has to be sufficient to contain enough information to allow the deciphering of the landscape).

Will this result maintain when using popularity as heuristic? In Figure 4 we have the answer to this question. There, we can observe that popularity performs similarly to average fitness.

Definitely, *less is more*, a small and well-selected set of best performers produce better results than a larger one or the use of the whole population and this result is valid for any level of complexity.

Nevertheless, we can also observe some differences when comparing with the heuristic based on the average fitness. Even if for a small sample, results are roughly

similar, as the size of the sample increases we can observe how, in popularity based ranking, the degradation of results is more pronounced, compared to the average fitness that appears to be more robust respect to variations on the sample size of. Therefore, when using the popularity of patterns as the guide for pattern selection and adoption, its accurate selection, given the small size of the best performing set, appears to be of great importance.

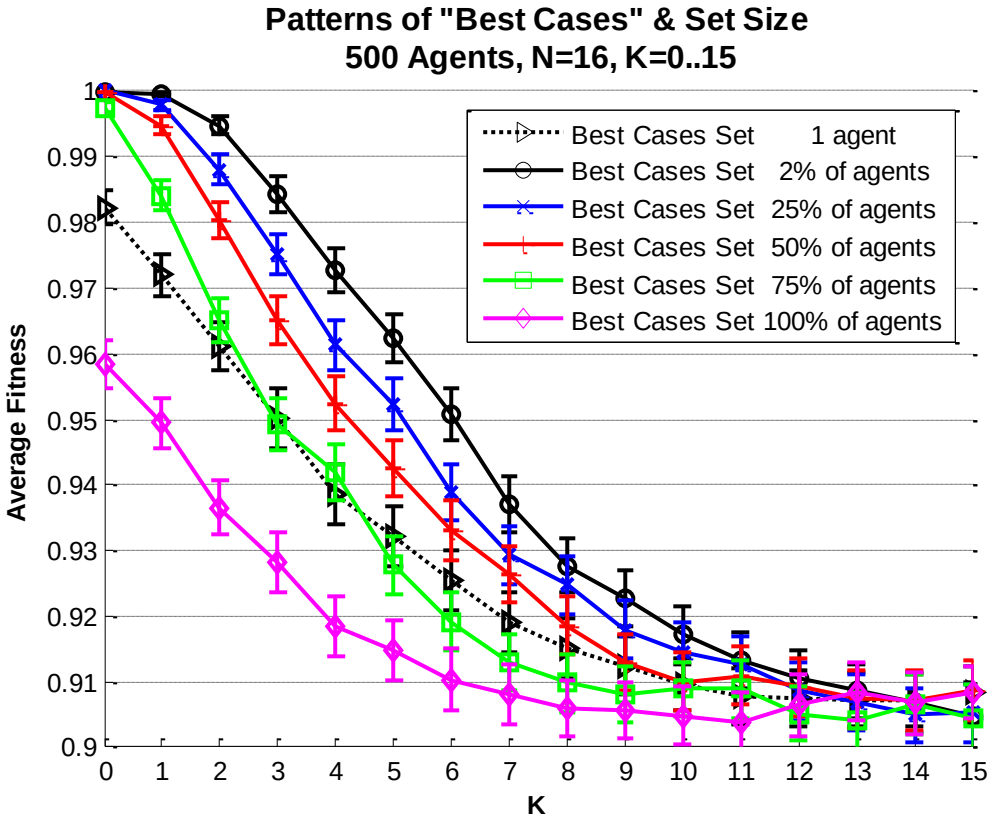


Fig 4. Variation in performance for different sample sizes pursuing a popularity heuristic

Again a set of 500 agents performing patterned search based, in this case, on the popularity of each pattern, are released in a NK landscape, 100 tries are performed and the results averaged. The experiment is repeated for different sizes of the set of best cases. The mean and its standard error are presented.

We can observe a similar behavior than in the case of using the average fitness as a selection method. However, in that case the system is even more sensible to set sample size, resulting in a faster and deeper degradation as set size increases, compared to Figure 3.

Proposition 3. Performance of pattern driven search increases inversely to the size of best cases sampled by fitness in decreasing order (considering only set sizes with enough information to decipher the landscape).

Because of proposition 2 we know that the average quality of the patterns presented to the agents is more relevant than the ordering because of its inaccuracy (proposition 1). Obviously the better the cases selected the better the quality of the patterns.

Therefore, less noisy sets with a higher pattern accuracy will perform better. Smaller sets of best practices sampled by fitness fit better this characteristic than larger sets.

Proposition 4. The size of the set of best cases sampled by fitness is more relevant in the case of ordering by popularity than in the case of ordering by average fitness.

Because of proposition 1 we know that the accuracy of the ordering increases as the set of best cases grows in the case of ranking by average fitness. And because of proposition 2 we know that the accuracy of ranking by popularity decreases as the set enlarges. Therefore larger sets will benefit from a ranking by average fitness and suffer from a ranking by popularity.

Given the results just described, the reader may be tempted to think that if less is more, maybe one is enough. In fact, as Figure 5 shows, results with only one agent are clearly worse and as we will discuss later on, there is an optimal range for set size, that even if small, must be significant.

The next question that we can ask ourselves could be around the consistency of these results across a diversity of population sizes. Do larger populations of agents benefit for a larger collection of best cases? Figure 4 provides the answer to this question. There we can observe how smaller sets continue to produce better results than larger ones for diverse cardinalities of the agent set. Concretely we present two examples, one with 100 agents and another one with 30 times that size, 3000 agents, where we observe similar behaviours.

However, as the size of the population increases we can witness two additional effects. First patterns are more accurate, and we are able to obtain high levels of average fitness. And secondly, the gap between sample set size closes, the system is therefore less sensitive to the size of the sample. Obviously, larger sets of agents have more possibilities of finding the best positions in the landscape than a smaller one. This implies that sample selection may be less critical in larger populations, because the lack of accuracy can be traded by larger samples.

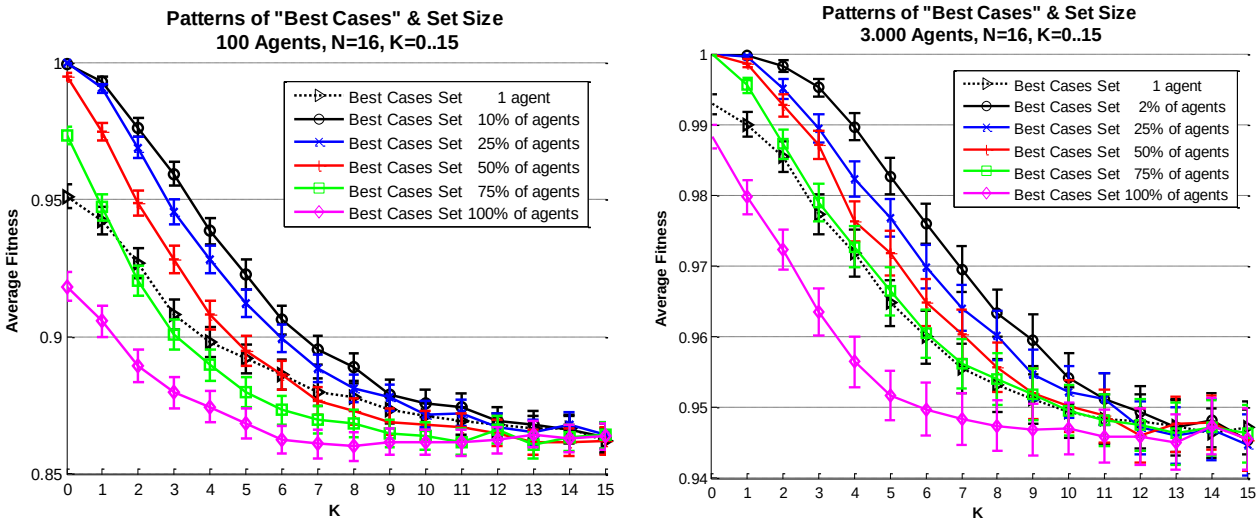


Fig 5. Comparison between two populations of 100 and 3,000 agents respectively

Results of the average fitness of two populations of agents, comprising 100 and 3,000 agents respectively are presented. In both cases 100 tries were performed with random generated landscapes in each try. Population average fitness and standard error are plotted in the graph.

We can observe a similar pattern of behavior, where smaller set sizes obtain better results than larger ones, together with two additional effects: a) lager populations allow the inference of better patterns leading to better results and b) the gap between sample set size decreases as population size increases.

Having established the fact that a smaller sample leads to better results than a larger one, being however, the sample significant enough to allow inferences that could be generalized in form of patterns, the next question should be about the possibility of exactly determining the optimal size of the sample, or at least providing some useful insights about it. This is the objective of the experiments presented in Figure 6.

There, we can observe how if it is not really possible to find an exact number of cases where the system performs at its peak, there is a range that clearly leads to better results. In our case and for 100 agents and a landscape of N=16 (2^{16} possible points) it can be situated between 5 and 15.

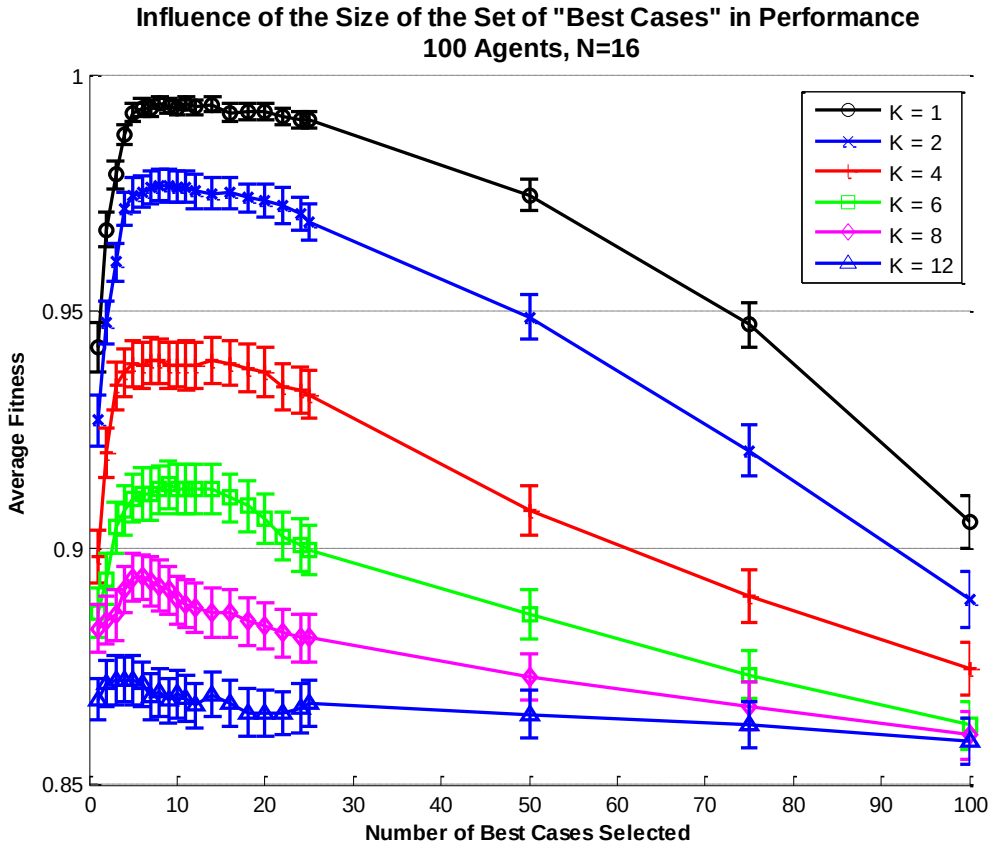


Fig 6. Determining the optimal sample size

A number of experiments with different sizes of the best case set, ranging from 1 agent to 100 agents, over a total population of 100 agents, are conducted. Results on population average fitness and standard error of 100 random experiments are plotted.

We can observe that even if it is not possible to point out an exact sample size, there is a range, between 5 and 15 where the system performs at its peak.

In fact, the optimal sample size to collect in order to be able to infer relevant information in the form of patterns is a balance between information and noise. A smaller sample ensures us low noise but at the cost of lower amounts of information. Following this reasoning, one may be tempted to believe that higher levels of complexity will require a larger sample in order to capture a level of information that is big enough to *solve* the landscape. Unfortunately, with more information comes more noise, reducing that way its effectiveness and leading to worse results. Later on, in this paper, we will discuss how the pro-active exploration by the agents, in addition to the use of the Cultural Algorithm, partially solves this problem.

8.1.1 Pattern Complexity

Until now, we have restricted our experiments to patterns of two bits, a simplification easily generalizable but explicatory enough for our objectives. However, we can also imagine patterns of 1 bit or in general, patterns of any length.

However, as pattern length increases, so does its number. In the case of binary patterns we have $C_2^{16}\ 2^2 = 480$ patterns, while for 3 bits $C_3^{16}\ 2^3 = 4{,}480$ patterns exist, for 4 bits $C_4^{16}\ 2^4 = 29{,}120$, and for 8 bits $C_8^{16}\ 2^8 = 3{,}294{,}720$ patterns.

This massive increase on the number of possible patterns implies that every agent will possess a larger number of them, potentially lessening the focus on the really good ones and promoting a dispersion of the popularity votes over a larger set. It seems therefore that we have to find a balance between two forces: larger patterns could be potentially more exact, while smaller ones will result in less noisy sets and focused population votes (due also to its smaller number).

In Figure 7 we can observe the results of the experiments carried out in order to get more insights about this balance. There, for two populations of 100 and 500 agents we can observe how patterns of size between 2 and 4 bits have similar levels of performance, especially at low complexity levels. Although smaller patterns (patterns of 2 bits) tend to marginally outperform larger ones, especially as complexity increases.

Again, patterns can be small, but to an extent, patterns of 1 bit although perform quite well at low complexity levels, abruptly fail as complexity increases. It can also be observed how larger populations of agents lead to better fitness results than smaller ones. Nevertheless, as the population increases so does the gap of performance between patterns of 1-bit and the rest, especially at high complexity levels.

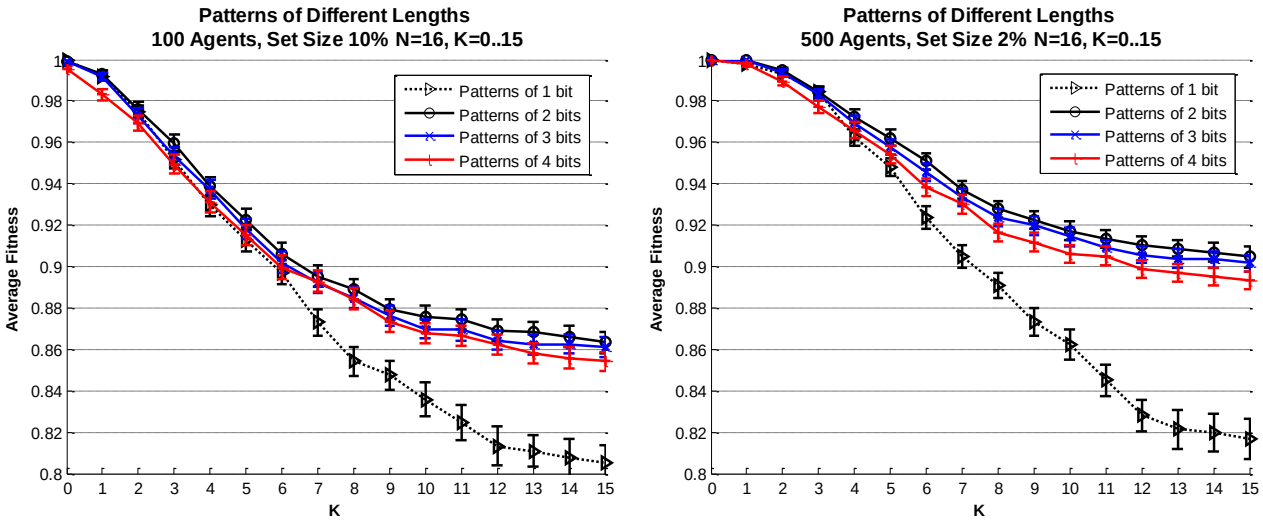


Fig 7. Performance of patterns of different lengths and different population sizes

Patterns of lengths from 1-bit to 4 components are used by two populations of 100 and 500 agents respectively. Results from 100 randomly created NK landscapes, average population fitness and its standard error are presented.

We can observe how patterns of 2-bits to 4-bits perform roughly similar, with a marginal advantage for the smaller 2-bit pattern. All beat the 1-bit pattern who abruptly fails as complexity increases.

9 The Importance of Diversity

In our implementation of cultural algorithms, agents rely solely on the knowledge coming from the belief space for their exploration. This mechanism served has well in the previous section in order to explore the best sample size and how comprehensive patterns should be.

However, this is not a realistic representation of *how* societies operate neither takes full advantage of the possibilities of software agents. In both cases, agents actively engage in search in addition to take advantage of the insight coming from the belief space. In this section we will discuss how our model behaves when facing this dual exploration, the one coming from the individual initiative and insights of the agents and the one coming from the distilled theories in the belief space.

Therefore, in figure 11 we compare the results of three different populations. One engaged solely on incremental search. A second one, engaged in popularity-based patterns. And a third one, where agents take advantage of their knowledge of the potential fitness if performing incremental improvements to decide if they use patterns (in the case that a pattern exist with better max fitness than the one attainable with an incremental improvement) or perform an incremental improvement. We have limited our exploration to incremental search; however, any other type of innovative engage on the side of the agents will produce similar results.

As we can observe, results increasingly favour the mix of patterns with incremental improvement as complexity (K) increases. What we can call active populations or populations actively engaged in exploration. The reason why this happens is because the increase in exploration those incremental improvements bring to the system, also results in an increase in its diversity. In fact, as we have discussed before, pattern adoption can be portrayed as a social construction. From the belief space comes a proposition of a concrete pattern that, given the information provided by the sample selected, looks promising. Is however, the population of agents who verifies or falsifies this hypothesis with its adoption, being the pattern discarded or reinforced.

If the exploration process performed by the agents, relies solely on the patterns provided by the belief space the only new information in the system is the one that comes from the crossover of the actual strategies of the agents with the proposed patterns, this new information is however limited, and the system converges fast to a very small number of strategies. This process of diversity creation is greatly enhanced by the active participation and engagement of the agents in an exploratory process.

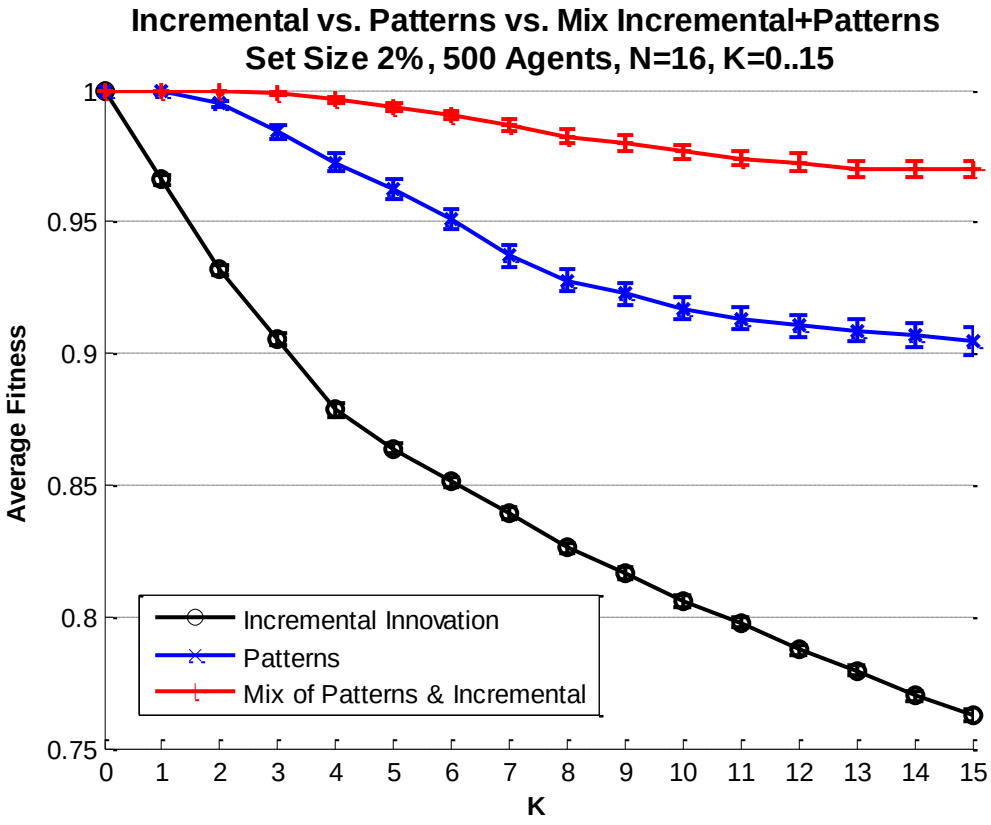


Fig 11. Comparison between three mechanisms: incremental, pattern-based and mix.

Three different experiments with populations of 500 agents performing three different strategies are presented: incremental, patterns based on popularity and mix. In the mix strategy, agents use their knowledge of the potential fitness of incremental search to decide which strategy to follow if incremental of pattern based.

We can observe how a mix strategy is clearly superior because it relies on a larger amount of relevant information upon which better patterns can be inferred.

The next question is if in this richer environment - information wise - still less is more, or we need larger sets of best cases in order to infer relevant patterns. Figure 12, attempts to answer this question by depicting the performance of several populations of agents with different set sizes. There, we can observe that if agents are actively engaged in incremental search and not only in taking advantage of the suggestions coming from the belief space, sample size is mostly irrelevant (if big enough).

The reason behind this change is the fact that by engaging in a mix of patterns and incremental search and by using the information provided by their knowledge of the immediate neighbourhood to direct pattern selection, the agents themselves manage

to reduce noise by avoiding irrelevant strategy configurations while increasing exploration and providing new opportunities for uncovering novel and better patterns.

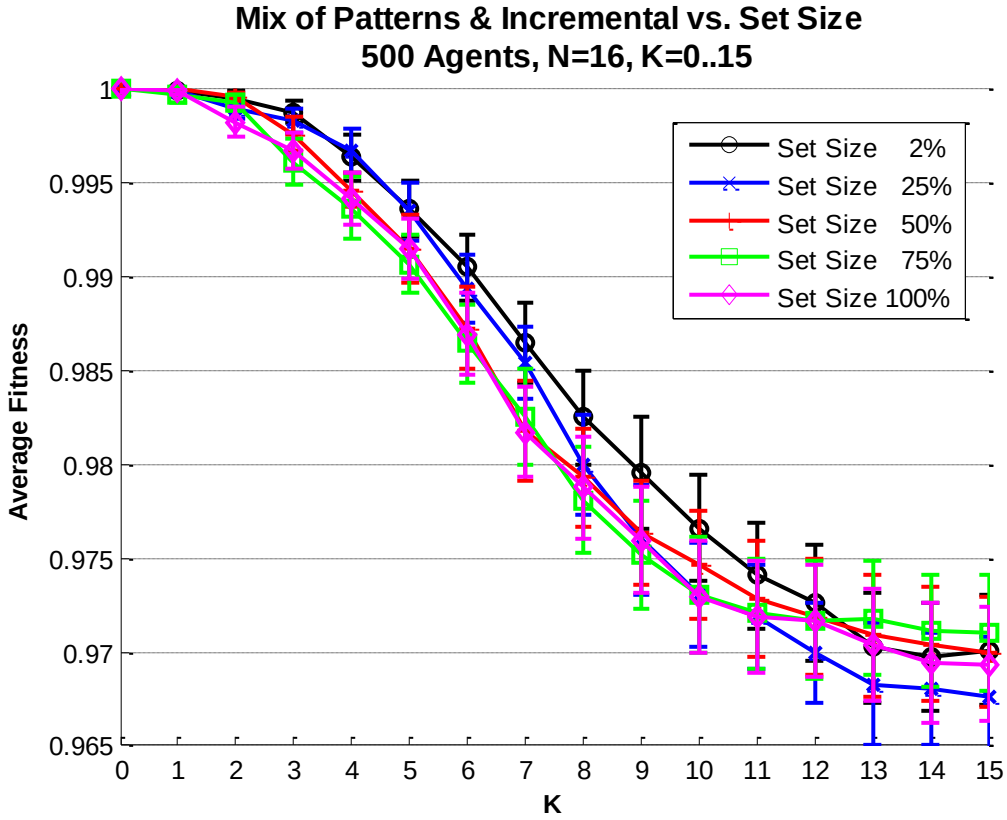


Fig 12. Relevance of set size when agents engage in a mix strategy.

A population of 500 agents engages in a mix strategy, using the information of incremental search to decide which strategy to follow: incremental or patterns. Experiments are performed 100 times and the average fitness and its standard error are plotted.

We can observe how if agents actively perform a mix strategy, set size is mostly irrelevant (although is still marginally better the smaller one). The reason behind is that agents themselves manage to clean unwanted patterns while increasing the process of exploration.

However individual exploration is costly, especially in human populations where structures prioritize effectiveness over a continuous involvement in exploration. Therefore, a relevant research question is whatever it will be possible to obtain good results by primary following the advices coming from the belief space and limiting the amount of exploration.

Answering this question could allow the agents to better focus their resources instead of diverting them with engaging in exploration activities. Figure 13 aims to provide an answer to this question. There we can observe how if agents rely 25% of their time on the advice coming from the belief spaces without even looking at their incremental opportunities and perform a mix of pattern-based and incremental search the remaining 75% of their time, they achieve similar results as if they were fully engaged in exploration 100% of their time.

What is more relevant is the fact that the level of engagement needed in incremental search, depends on complexity. In fact for lower levels of complexity, the engagement in even smaller amounts of exploration seems to be a viable alternative.

Even for higher levels, engaging only 50% of the time in a decision of whether to bet on incremental or patterns while relying the other 50% solely on patterns, seems to be a fairly good alternative that still reverts in important gains close to maximum.

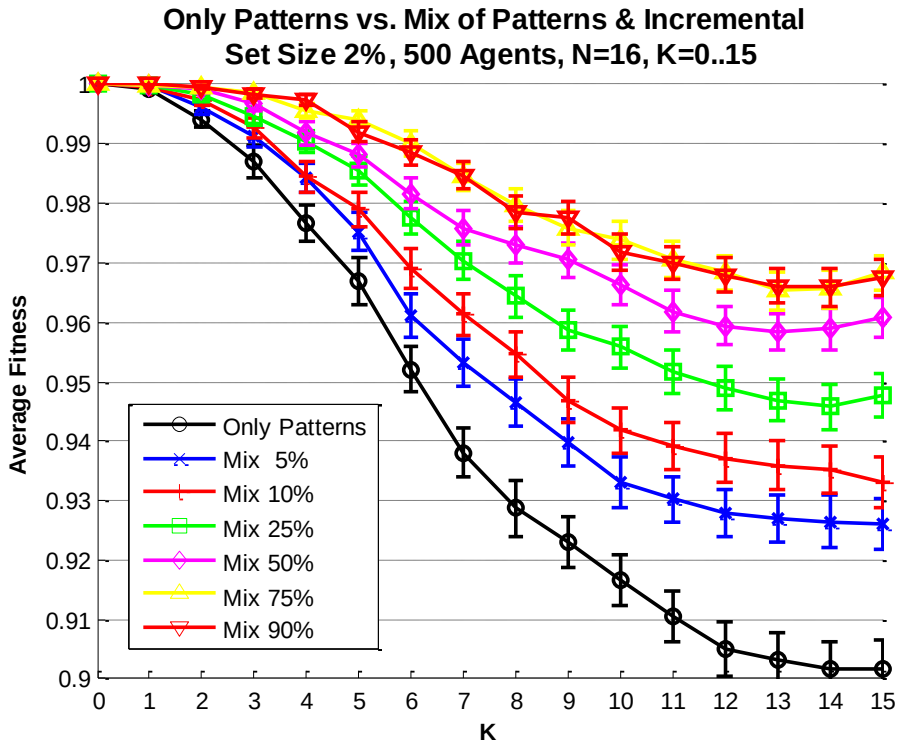


Fig 13. Relevance of the incremental engagement of the agents on patterns

Different percentages of a mix strategy (patterns and incremental) are included in a pattern driven strategy selection. Percentages range from 5% to 90% of forced mix strategy. Experiments are conducted using a set corresponding to the best performing 2% of the population of agents. Averaged results of 100 experiments per mix are presented. Average fitness and standard error of the population are plotted.

We can observe how for a low level of complexity the complete reliance on patterns produces results that are very close to mix strategies. However, as complexity increases becomes evident the need for a percentage of mix strategies in exploration. Nevertheless, we can appreciate how a reliance of 50% on patterns, leaving the other 50% to mix strategies produces results very close to maximum. We can also notice that beyond 75% of reliance in mix strategies there is no improvement.

10 Conclusions and Future Work

Cultural Algorithms can indeed achieve very good results in terms of performance. And they do so with a surprising frugality of resources and without the need of having access to specific performance figures; a public signal that can result in an ordered set is enough. Even more, as we have seen in this research, we only need to be able to discern the best group of agents from the rest.

Popularity is indeed a good heuristic, and as we have demonstrated not only a measure of popularity among the very best is the only signal needed, but Cultural Algorithms work better when this set comprising the best performers is rather small: less is more in Cultural Algorithms. We also have described that they are flexible enough to allow for substitutions, such as trading a larger population for a better selection.

During this work we also have explored other alternatives to discovery beyond the solely use of cultural algorithms. In fact, natural and artificial societies engage in autonomous discovery processes independently of the adoption of common beliefs. We have modeled this process by using a mix of incremental and patterned search. Again, we to a result pointing to frugality: a mere 25% of autonomous exploration was enough for the pattern mechanism to amplify and transform these results into ones equivalent to a complete involvement in exploration.

Both results point to probably the most interesting characteristic of Cultural Algorithms: the fact that they are social algorithms.

As such they exhibit an extraordinary capacity to amplify the autonomous discoveries of the agents through a process of abstraction, adoption and recombination through cross-over.

But, at the same time, they rely in this same process of adoption to ensure their success. In fact, the whole process can be stylized as one of hypothesis forming through social abstraction of best patterns. These hypotheses get corroborated or falsified by this same adoption process.

Societies can therefore be more innovative simply by being better adopters, making diffusion and discovery just two sides of the very same coin.

This is certainly an interesting result, because even do we really do not know much of how to spur discovery, we know much better how to foster diffusion.

References

Alami, J., Bernameur, L. El Imrani, A. (2007). Fuzzy Clustering Based Parallel Cultural Algorithm. *International Journal of Soft Computing*, 2(4): 562-571.

Apesteguia J., S. Huck and J. Oechssler. *Imitation - Theory and Experimental Evidence***.** Working Papers 0419, University of Heidelberg, Department of Economics, Apr 2005.

Al-Shehri, H. (1997). Evolution-based Decision Tree Opmitization Using Cultural Algorithms. PhD Dissertation, Wayne State University.

Bäck, T. (1996) *Evolutionary Algorithms in Theory and Practice*, Oxford, NY.

Bäck, T. (1996). Evolution strategies: An alternative evolutionary computation method. In Artificial Evolution, volume Lecture Notes in Computer Science 1063, pages 3-20. Springer-Verlag.

Becerra, R.L., Coello C.A. (2004).Culturizing Differential Evolution for Constrained Optimization. ENC 2004: 304-311.

Boyd, R., and Richerson, P. J.(1985). *Culture and the Evolutionary Process*. Chicago: University of Chicago Press.

Chung, C-J. and Reynolds, R.G. (2000). Knowledge-Based Self-Adaption in Evolutionary Search. International Journal of Pattern Recognition and Artificial Intelligence, vol. 14, n. 1.

Coello C. and Becerra, L. (2003). Evolutionary multi-objective optimization using a cultural algorithm. IEEE Swarm Intelligence Symposium.

Curran, D. (2006). An Empirical Analysis of Cultural Learning: Examining Fitness, Diversity and Changing Environments in Populations of Game-Playing Neural Network Agents. PhD Dissertation, National University of Ireland, Galway.

Dawkins, R. (1989). *The Selfish Gene* . Oxford University Press.

Dasgupta, D., Michalewicz, Z. (1997). *Evolutionary Algorithms in Engineering Applications.*Springer-Verlag.

Dosi, G. (1988). “Sources, procedures and microeconomic effects of innovation”. *Journal of Economic Literature* XXVI, 1120-1171.

Dosi, G., Fremann, C., Nelson, R.R., Silverberg, G., Soete, L. (Eds.) (1988). Technical Change and Economic Theory. Francis Printer. London.

Dosi, F., Fabiani, S., Aversi, R., Meacci, M. (1994). “The dynamics of international differentiation: a multicountry evolutionary model”. *Industrial and Corporate Change* 3, pp. 225-242.

Dosi, G., Ermoliev, Y.M., Kanivski, Y.M. (1994). Generalized Urn Schemes and Technological Dynamics. *Journal of Mathematical Economics*, 23:1-19, 1994.

Dosi, G., Marengo, L., and Fagiolo, G. (2003). Learning in Evolutionary Environments. Laboratory of Economics and Management Sant'Anna School of Advanced Studies 2003/20. Pisa 2003.

Dosi, G., Marsili, O., Orsenigo, L., Salvatore, R. (1995). "Learning market selection and the evolution of industrial structures". *Small Business Economics* 7, 411-436.

Dosi, G., Malerba, F., Marsili, O., Orsenigo, L. (1997). "Industrial structure and dynamics: evidence, interpretations and puzzles". *Industrial and Corporate Change* 6, 3-24.

Durham, W.H. (1991). *Coevolution: Genes, Culture and Human Diversity*. Stanford University Press.

Edelman,G., Tononi,F. (2001). *Consciousness: How Matter Becomes Imagination*. Penguin Books. London, 2001.

Flannery, K.V. (1968). Archaeological Systems Theory and Early Mesoamerica. In *Anthropological Archaeology in the Americas*, ed. by B. J. Meggers, pp. 67-87. Washington, Anthropological Society of Washington.

Fogel, D.B. (1994). An Introduction to Simulated Evolutionary Optimization. *IEEE Transactions on Neural Networks*, vol 5, number 1.

Fogel L.J., Owens A.J., and Walsh M.J. (1966). *Artificial Intelligence Through Simulated Evolution*. Wiley & Sons, Inc., New York.

Geertz, C. (1973). Thick Description: Toward an Interpretive Theory of Culture. In *The Interpretation of Cultures: Selected Essays*. New York: Basic Books.

Gurven M.D. (2004). To give and to give not: the behavioral ecology of human food transfers. *Behavioral and Brain Sciences* 27(4):543-571.

Hebb, D.O. (1949). *Organization of Behavior.* New York : Wiley (1949).

Henrich J. and R. Boyd (1998). The evolution of conformist transmission and between-group differences*. Evolution and Human Behavior*, 19: 215-242.

Henrich J. and Gil-White, F. J (2001). The evolution of prestige: Freely conferred status as a mechanism for enhancing the benefits of cultural transmission. *Evolution and human behavior* 22:165-196.

Holland, J.H. (1975). *Adaptation in Natural and Artificial Systems*. Ann Arbor, University of Michigan Press.

Kameda T. and D. Nakanishi (2002). Cost-benefit analysis of social/cultural learning in a non-stationary uncertain environment: An evolutionary simulation and an experiment with human subjects. *Evolution and Human Behavior*, **23**, 373-393.

Kauffman and Levin (1987). Kauffman, S.A. and S. Levin. Towards a general theory of adaptive walks on rugged landscapes. *Journal or Theoretical Biology* 128:11-45, 1987.

Kauffman,S.A. (1989). ”Adaptation on rugged fitness landscapes”. *Lectures in the Sciences of Complexity* 527-618. Addison-Wesley, Redwood City. SFI Studies in the Sciences of Complexity, Lecture Volume I, 1989.

Kauffman, S. (1993). The Origins of Order. Oxford University Press. New York.

Kauffman,S.A., Levin,S. (1987). ”Towards a general theory of adaptive walks on rugged landscapes”. *Journal or Theoretical Biology* 128:11-45, 1987.

Kauffman, S.A., Lobo, J., Macready, W.G. (1998) “Optimal search on a technology landscape”, Santa Fe Working Paper 98-10-091.

Kinnaird-Heether, L. and Reynolds, R.G.(2009). Survival of the Fastest: Using Cultural Algorithms to Optimize the Design of a Controller for a 3D Racing Game. Presentation at Wayne State University, 1/27/2009.

Kohler, T. (2000). The Final 400 years of pre-Hispanic Agricultural Society in the Mesa Verde Region. Kiva, V.22.

Kobti, A., Kohler, T, Reynolds, R.G. (2003). Robustness in Coupled Human/Natural Systems in the Northern Prehispanic Southwest. Santa Fe, NM, Studies in the Sciences of Complexity.

Kobti, Z., Kohler, T., Reynolds, R.G. (2004). The effects of generalized reciprocal exchange on the resilience of social networks: An examples from the prehispanic Mesa Verde regions. J.Comput.Math.Org.Theory, pp 227-254.

Kobti, Z. and Reynolds, R.G. (2005). Modeling Protein Exchange across the Social Network in the Village – A Multi-Agent System Simulation. IEEE International Conference on Systems, Man and Cybernetics, vol. 4, pp 3179-3203.

Kobti, Z, Snowdon, A. W., Rahaman, S., Dunlop, T., Kent, R. D. (2006), A Cultural Algorithm to Guide Driver Learning in Applying Child Vehicle Safety Restraint, 2006 IEEE Congress on Evolutionary Computation, pp 1111- 1118.

Levintthal, D. (1997). “Adaptation in rugged landscapes”. *Management Science*, 43(7), pp. 934-950.

Levinthal,D. ,Warglien, M. (1999). “Landscape Design: Designing for Local Action in Complex Worlds”. *Organization Science* 10)(3): 342-357.

McElreath, R., M. Lubell, P. Richerson P*, et al. (2005)*. Applying evolutionary models to the laboratory study of social learning. *Evolution and Human Behavior*, **26** 483-508.

Michalewicz, Z., Logan, T.D., and Swaminathan , S. (1994). Evolutionary Operators for Continuous Convex Parameter Spaces. In Proceedings of the 3rd Annual Conference on Evolutionary Programming, A.V. Sebald and L.J. Fogel , Editors. World Scientific Publishing, River Edge, NJ, 1994.

Miller, B. L., and Goldberg, D. E. (1995). Genetic algorithms, tournament selection, and the effects of noise. *Complex Systems*, 9(3), 193–212. (Also IlliGAL Report No. 95006).

Mitchell, T.M. (1978). Version Spaces: An Approach to Concept Learning. PhD Dissertation, Stanford University.

Mitchell, T.M. (1997). *Machine Learning*. McGraw Hill.

Murdoch, W.W. (1975). *Environent: Resources, Pollutions and Society*. Sunderland, Massachusetts: Sinauer Associates, 2nd Edition. 1975.

North,C. (2005). Understanding the Process of Economic Change. Princeton University Press, 2005.

Ochoa, A. et al. (2007).Implementing Data Mining to improve a Game Board based on Cultural Algorithms, in proceedings of Hais'2007.

Ochoa, A., Padilla, A., Gonzalez, S. Castro, A., Hal, S. (2008). Improve a Game Board based on Cultural Algorithms. *The International Journal of Virtual Reality*, 7(2):41-46.

Ostrowski, D.A., Tassier, T., Everson, M., Reynolds, R.G. (2002). Using Cultural Algorithms to evolve strategies in agent-based models. Proceedings of the Congress on Evolutionary Computation, 2002 vol. 1, pp 741-746.

Ostrowski, D., Schleis, G.,Rychtyckyj, N., Reynolds, R.G. (2003). Using Cultural Algorithms in Industry. Proceedings of the Swarm Intelligence Symposium, IEEE, pp 187-192.

Rechenberg, I. (1973). *Evolutionstrategie: optimierung technisher systeme nach prinzipien der biologischen evolution*. Frommann-Hoolzboog Verlag.

Reynolds, R.G. (1978). On Modeling the Evolution of Hunter-Gatherer Decision-Making Systems. *Geographical Analysis*, vol X n. 1, January 1978.

Reynolds, R.G. (1979). An Adaptative Computer Model of the Evolution of Agriculture for Hunter-Gatherers in the Valley of Oaxaca. PhD Dissertation, Univeristy of Michigan, Ann Arbor.

Reynolds, R.G. and Maletic, J.I. (1994). The Evolution of Cooperate Using Cultural Algorithms. In Proceedings of the Third Annual Conference on Evolutionary Programming. Anthony V. Sebald and Lawrence J. Fogel Editors, World Scientific Press, Singapore.

Reynolds, R.G.(1994). An Introduction to Cultural Algorithms. Proceedings of the Third Annual Conference on Evolutionary Programming, San Diego, California, pp. 131-139.

Reynolds, R.G., Michalewicz, Z. and Cavaretta, M.J. (1995). Using Cultural Algorithms for Constraint Handling in Genocop. In Evolutionary Programming IV, J.R. MacDonnell, R.G. Reynolds and David B. Fogel, editors. Bradford Book, MIT Press, Cambridge, Massachusetts.

Reynolds, R.G. and Rolnick, S.R. (1995). Learning the Parameters for a Gradient-Based Approach to Image Segmentation from the Results of a Region Growing Approach Using Cultural Algorithms. IEEE International Conference on Evolutionary Computation, November 29-December 1, 1995 Perth, Australia, pp. 1135-1143.

Reynolds, R.G. and Sverdlick, W. (1995). An Evolution-Based Approach to Program Understanding Using Cultural Algorithms. *International Journal of Software Engineering and Knowledge Engineering*, Vol. 5, No. 2, June 1995.

Reynolds, R.G. and Chung, Chan-Jin (1997). Function Optimization using Evolutionary Programming with Self-Adaptative Cultural Algorithms. Lecture Notes on Artificial Intelligence. Springer-Verlag Press.

Reynolds, R.G. and Sternberg, M. (1997). Using Cultural Algorithms to Support the Re-Engineering of Rule-Based Expert Systems in Dynamic Performance Environments: A Fraud Detection Example. *IEEE Transactions on Evolutionary Computation*, Vol 1, No 4, November 1997.

Reynolds, R.G., Zannoni, E. (1997). Learning to Control the Program Evolution Process in Genetic Programming Systems Using Cultural Algortihms. *Journal of Evolutionary Computation*, vol . 5, n. 2, October 1997.

Reynolds, R.G., Chung, C.J. (1998). CAEP: An Evolution Based Tool for Real Valued Function Optimization Using Cultural Algorithms. International Journal on Artificial Intelligence Tools, vol. 7, no. 3, September 1998, pp. 239-293.

Reynolds, R.G., Jin, X. (2002). Regional Schemata for Real-Valued Constrained Function Optimization Using Cultural Algorithms. *Journal of Natural Computing*.

Reynolds, R.G., Jacoban, R., Brewster, J. (2003), Cultural swarms: assessing the impact of culture on social interaction and problem solving, Proceedings of the 2003 IEEE Swarm Intelligence Symposium, pp212-219.

Reynolds, R. G., Saleem, S. (2003), The Impact of Environmental Dynamics on Cultural Emergence, Oxford University Press, pp1-10.

Rivkin, J. (2000). "Imitation of Complex Strategies". *Management Science* (46), 2000, pp.824-844.

Rivkin,J.W., Siggelkow,N. (2002). "Organizational sticking points on NK Landscapes". *Complexity* 7(5): 31 – 43.

Rivkin J., Siggelkow N. (2003). "Balancing Search and Stability: Interdependencies Among Elements of Organizational Design", *Management Science*, v.49 n.3, 2003, pp.290-311.

Schlag K.H. (1998). Why Imitate, and If So, How?, : A Boundedly Rational Approach to Multi-armed Bandits, *Journal of Economic Theory*, Elsevier, vol. 78(1), pages 130-156, January 1998.

Schlag, Karl H. (1999). Which one should I imitate? *Journal of Mathematical Economics*, Elsevier, vol. 31(4), pages 493-522, May 1999.

Schwefel, H.P. (1975). *Evolution Strategy and Numerical Optimization (in German)*. Dissertation, Technical University of Berlin.

Tylor, E.B. (1871). *Primitive Culture*. (reissued by Cambridge University Press, 2010. ISBN 9781108017527).