Biases in human behavior

Massimo Egidi
megidi@gelso.unitn.it

CEEL
Computable and Experimental Economics Laboratory
University of Trento
http://www-ceel.gelso.unitn.it/

ABSTRACT

The paper shows that biases in individual's decision-making may result from the process of mental editing by which subjects produce a "representation" of the decision problem. During this process, individuals make systematic use of *default classifications* in order to reduce the short-term memory load and the complexity of symbolic manipulation. The result is the construction of an imperfect *mental representation* of the problem that nevertheless has the advantage of being simple, and yielding "satisficing" decisions. The imperfection origins in a trade-off that exists between the *simplicity* of representation of a strategy and his *efficiency*. To obtain simplicity, the strategy's rules have to be memorized and represented with some degree of abstraction, that allow to drastically reduce their number. Raising the level of abstraction with which a strategy's rule is represented, means to extend the domain of validity of the rule beyond the field in which the rule has been experimented, and may therefore induce to include unintentionally domains in which the rule is inefficient. Therefore the rise of errors in the mental representation of a problem may be the "natural" effect of the categorization and the identification of the building blocks of a strategy.
The biases may be persistent and give rise to lock-in effect, in which individuals remain trapped in sub-optimal strategies, as it is proved by experimental results on stability of sub-optimal strategies in games like Target The Two.
To understand why sub-optimal strategies, that embody errors, are locally stable, i.e. cannot be improved by small changes in the rules, it is considered Kauffman' NK model, because, among other properties, it shows that if there are interdependencies among the rules of a system, than the system admits many sub-optimal solutions that are locally stable, i.e. cannot be improved by simple mutations. But the fitness function in NK model is a random one, while in our context it is more reasonable to define the fitness of a strategy as efficiency of the program. If we introduce this kind of fitness, then the stability properties of the NK model do not hold any longer: the paper shows that while the elementary statements of a strategy are interdependent, it is possible to achieve an optimal configuration of the strategy via mutations and in consequence the sub-optimal solutions are not locally stable under mutations.
The paper therefore provides a different explanation of the existence and stability of suboptimal strategies, based on the difficulty to redefine the sub-problems that constitute the building blocks of the problem's representation.

1.Introduction

A large body of experimental results shows that human decisions display systematic deviations from fully rational ones, and that in many cases 'errors' persist even when the rational solution has been explicitly shown to the subjects. This happens both in individual and team decision-making. In the case of teams and organizations, it may happen that systematically erroneous decisions are made by organizations because - given the bounded rationality of the individuals of which they consist - they may adopt routinized strategies, which are not, changed even when they are highly sub-optimal.

Levinthal and March single out a number of ‘traps’ into which an organization may fall during the process of organizational learning, and among them they analyze the ‘failure myopia’ and the ‘success trap’.

‘Failure myopia’ consists in the tendency of an organization to ignore or underestimate failures in its activities. The consequence of this form of myopia is the tendency of an organization to overestimate its possibilities of success. A similar phenomenon has been found in the psychology of the individual probabilistic judgement, where it is known as the ‘hindsight effect’. As a result of this effect, individuals underestimate past errors in probabilistic assessment and consequently overestimate their evaluative capacities (Fischhoff, 1975).

As regards the ‘success trap’, the tendency of organizations to focus on success may induce them to persist excessively in the use of procedures and actions that have been associated with successes in the past. Consequently, an organization that falls into this trap tends to anchor its activity in processes of organizational exploitation, to the detriment of research and innovation. Moreover, this tendency may induce organizations not to adapt, or to adapt with difficulty, to changed environmental conditions.

The tendency to extrapolate to inappropriate situations procedures that in the past have proved efficacious, which is triggered by the emergence of new conditions in a context of routinized behavior, it has also been observed at the individual level. In solving repetitive tasks also individuals exhibit "routinized behaviors", i.e. they use in an automatic and sometimes not deliberate way the procedures the have learnt, even in conditions in which they are not efficient.

A key problem is therefore to see whether routinization is created only by the interaction effects among individuals of a team, or it is rooted on some features of individual mental activity. There are clues that the second alternative is the right one: the tendency to routinize exists also at the individual level and has been implicitly proved in the studies by Luchins (1942) and Luchins and Luchins (1950). They suggest in fact that in contexts of repetitive tasks, individuals may “automatically” use the rules they have learnt in a given context even beyond their original domain. This process is called “mechanization of thought”. To better clarify this point I have compared the biases emerging in the team activity with those emerging in the activity of individuals facing the same problem. Starting from a previous experiment with Narduzzo, (1996), in which, using the game *Target the Two*, we have shown that in cooperative contexts team biases may be originated by a process of routinization, I have run a new experiment in which single individuals were performing the same activities of the teams in the previous experiment. The results of this new experiment suggest that biases emerging when individuals discover a satisficing solution of a problem in a given context, are originated in the individual's construction of mental models and reinforced and stabilized by the teams interaction.[2]

Having realized that, at least in some experimental context, we can consider biases in individual behaviors as the “seed” of the process of team routinization, in the rest of the paper I will focus the attention on the first argument, i.e. the emergence of biases in *individual* problem solving and reasoning.

As the experiments with *Target the Two* show, the routinization of the individual's behavior may be a source of biases, because individuals tend to extend the domain of the routine beyond the domain of optimality. The extrapolation of a procedure beyond its domain of validity is the starting point I will use to explain how some biases are created by individuals. I will show that biases in individual’s decision-making may result from the process of mental editing by which subjects produce a “representation” of the decision problem. During this process, individuals make systematic use of *default classifications* in order to reduce the short-term memory load and the complexity of symbolic manipulation. The result is the construction of an imperfect *mental representation* of the problem that nevertheless has the advantage of being simple, and yielding "satisficing" decisions.

I will propose later in detail a definition of "mental representation". In short, a representation is a way to codify and compress the information on which individuals base their activity of problem solving and construct their strategies. The discovery and the construction of a strategy to solve a problem, especially in the context of repeated games, it is a slow and difficult process in which individuals, learning from the experience, try to build-up as simple and efficient as possible rules of action. But there exists a trade-off between the *simplicity* of representation of a strategy's rules and their *efficiency*. To obtain simplicity, the strategy's rules have to be memorized and represented with some degree of abstraction, to allow one drastically to reduce their number. Raising the level of abstraction with which a strategy's rule is represented, means to extend the domain of validity of the rule beyond the field in which the rule has been experimented, and may therefore induce one to include inadvertently domains in which the rule is inefficient. Therefore the rise of errors in the mental representation of a problem may be the "natural" effect of the categorization and the identification of the building blocks of a strategy. This argument will be discussed in section 2.

To describe how hidden errors are generated during the categorization of the building block of a strategy, we need to understand how individuals build-up sub-optimal solutions, and to compare these solutions with the optimal ones.

One well-known method of problem solving it to decompose a problem recursively into sub-problems that are simpler to be solved, until are identified elementary solvable sub-problems. It is useful to realize that a given problem may be decomposed in a large variety of different ways that imply also different levels of abstraction in the categorization of sub-problems. Every decomposition pattern is the result of a different way of codifying information at different levels of abstraction, and gives rise to a different strategy. In Section 2.3-26 we will compare the optimal decompositions of a problem with other decompositions, which are simpler and easier to be learnt, but sub-optimal. By analysing in full details a specific puzzle we will see that biases emerge from the activity of problem decomposition and extrapolation, in which individuals identify sub-problems as building blocks, that does not coincide perfectly with the domain of optimal responses, but embody also sub-optimal rules. The formal context in which we will examine the nature of the "representations" and the rise of biases during the categorization of sub-problems is that of puzzles and finite state machines (automata) (sections 2.1,2.2).

In Section 3.4 we will turn the attention to the stability of the representations. The underlying task is to explain the experimental results on stability of sub-optimal strategies. We have to understand why sub-optimal strategies, that embody errors, are locally stable, i.e. cannot be improved by small changes in the rules.

A very attractive schema to explore the properties of the optimal and sub-optimal strategies and their stability, is Kaufman's NK model, because, among other properties, it shows that if there are interdependencies among the rules of a system, then the system admits many sub-optimal solutions that are locally stable, i.e. cannot be improved by simple mutations.

However, the fitness function in the NK model is a random one. In our context, having identified a strategy with a program (in a world of finite state machines), it is more reasonable to define the fitness of a strategy as efficiency of the program.

The simplest way to define the efficiency of a program is to refer to the number of steps that it requires to achieve the goal. The shorter the path from the starting state to the goal state, the more efficient the strategy. Fitness therefore will be defined in relation to the length of the path connecting the starting and the goal state. This definition is simple and natural if we want to evaluate and compare the efficiency of different programs (strategies).

If we introduce this kind of fitness, then the stability properties of the NK model do not hold any longer: we will see, in fact, (section 3.2 and Appendix) that while the elementary statements of a strategy are interdependent, it is possible to achieve an optimal configuration of the strategy via mutations and in consequence the sub-optimal solutions are not locally stable under mutations.
Therefore a different explanation of the existence and stability of sub-optimal strategies have to be found.
Suppose that an individual has learnt a strategy during a sequence of repeated games. Suppose that in a new run he encounters a game configuration never previously faced. If the strategy that he has adopted is complete, one of the rules of the strategy is applicable *by default* to the configuration. Suppose that the rule of the strategy that matches with the new configuration is sub-optimal in this new context. Then if the individual applies the old rule, he does not increase the complexity of the strategy - he continues to use the list of known rules - but he increases the number of inefficient responses, because he applies once more a sub-optimal rule.
If on the contrary the individual discovers that a more efficient rule can be activated in response to the new specific condition, he must add the new rule to the list, considering it an “exception” to the rule that was previously used by default. In this case the individual improves the efficiency of the strategy but increases at the same time the complexity of the rule system.
Therefore the trade-off between the simplicity of the strategy's representation and his efficiency that we have discussed before emerges once again. I suggest that this trade-off is the driving force that causes the stability of a sub-optimal strategy, i.e., that prevents individuals from seeking a more efficient strategy when they must cope with only a single or a limited number of "exceptions" to the known rules.
The problem is that adding one exception to a system of abstract rules does not offer a new, compact representation of the strategy, because if individuals transform a specific exception into a new rule with some degree of generality, the new rule conflicts with many other pre-existing rules of the strategy. Therefore individuals may prefer to maintain the old system eventually adding few exceptions, that to rethink new abstract rules, because the mental effort to redefine a sub-problems system is higher that the effort to memorize an exception. Therefore few mutations, even improve the system efficiency, are not sufficient to push the system to a long jump toward a new stable configuration. (The rule system may be characterized by a central core of stable general rules, encircled by a cloud of more specific rules that can slightly vary in relation to the different individuals.)
If the number of “exceptions” becomes too large, and they happen systematically during the game, individuals cannot simply continue to memorize new exceptions, but have to restructure the space of the rules, recodifying information; In other words, they have to change the representation. This change may be highly discontinuous, because it generally requires one to destructure the division of problems and re-design the problem with new building blocks.
Concluding the paper, I will use the properties of the representations to suggest an explanation of the discontinuities of the organizational change: a brief comparison of the performances of distributed, parallel organizations, with the hierarchical ones in changing their internal rules and getting out from organizational traps will close the discussion.

2. Puzzles, Automata and Problem Solving

To explore problem solving activity in simplified environments described by systems of rules, it is convenient to choose systems like puzzles and planned activities. [3]
A puzzle is a game in which a single individual must achieve a given goal, usually by making changes to an initial configuration of the game according to a system of rules and constraints. The rules state what actions may be made for each state of the game, and what their effects will be. A puzzle, therefore, is defined by a system of action rules that operate on configurations or states of the game. Although this type of game has a very simple formal description, it still displays most of the typical properties of general problem-solving models. In fact planning with Artificial Intelligence systems, which is one of the most important areas of problem solving, has the same basic definitions as puzzle solving. [4]
There is a second reason for choosing the context of puzzle solving and planning in A.I. Just because it is a simplified context to examine the properties of problems, a puzzle is a good platform to analyze the decomposition in sub-problem and therefore to study the solution of a problem as a cooperative process; in fact, if it is possible to decompose in a set of independent sub-strategies the strategy to achieve the goal of the puzzle, then it is possible to attribute to different individuals the task of discovering and play the sub-strategies. A puzzle can therefore be transformed into a game played by numerous cooperating individuals; the game remains formally the same, but with the essential difference that in this case a strategy must be discovered by a team rather than by an individual, so that a process of organizational rather than individual learning takes place.
A puzzle can be represented very simply by defining the characteristics of the *states* of the game and the *rules* by which they are modified. A well-known puzzle is the Rubik's Cube – a cube with six faces consisting of nine tiles in six different colors. The rules of the puzzle essentially permit the player to rotate the sections of the cube and rearrange the positions of the tiles. Usually, the final configuration of the game is the one in which each face consists of tiles of the same color. This puzzle, like all others, can therefore be represented as a condition-action system, or in other words, a list of system-states coupled with the rules to be applied. The system can be depicted as follows:

Table 1

Conditions	Actions			
(States)	(Rules)			
S1	Ai	As	….	Ap
S2	Aj	Ar	….	Aq
S3	Ak	Ah	….	Al
…..	….		….	
SN	Ag	An	…	As

where $S_1,\ldots S_n$ are (all) the states of the game and A_1, $A_2,\ldots A_m$ are the actions that can be performed. This representation describes the puzzle as an array in which the first column contains the states of the system, and the others contain the actions permitted. (in Rubik cube, $N \cong 3\ 000\ 000\ 000$).
Note that if the states of the game are observable, we can know the configuration that is achieved by the puzzle when we apply a given move to a given configuration. In other words, by applying to the state S_i the action (or operator) A_j we get the new state S_k, for all i, k ε N, jε m.[5]
Therefore we can construct a new array that shows the links among the puzzle's configurations, as it is shown in Table 2.

Table 2

	Actions				
	A1	A2	A3	…..	Am
States	NewStates				
1	n1	m1	r1	….	w1
2	n2	m2	r2	….	w2
3	n3	m3	r3	….	w3
…..				….	

In the Table 2 we have codified the configurations (conditions) of the puzzle with progressive numbers 1,2,3,…N. The same holds for the "successors" or "next configurations" of a given configuration, i.e. the configurations that are reached by applying the permissible rules (A_1, A_2,…A_m) to a given configuration. In this form the Table exhibits the links between every state and its successors but showing not the configurations but only the numbers that denote them. This new table is called the *links array.* Each row of the links array therefore consists of the following elements: the first is the number that denotes the state of the system, i.e. the 'state label'; the other elements in the row are the numbers that denote the successors. Thus, if we apply rules A_1, A_2,…A_m to the state k, we obtain the "successors", i.e. the states nk,mk,…wk.
In this context a problem is defined by a pair of states, defined as initial state and goal state, and to solve a problem means to achieve the goal state from the initial state through a sequence of rules applications. This sequence is also called "plan" in A.I. planning literature, and together with the definitions I have provided, it fully applies to that field (Yang, 1997). Of course a problem for which does not exist any sequence connecting the starting configuration with the goal, it is a problem that does not allow solution.
As I have said, in order to devise a strategy to achieve the goal, it is necessary to find a sequence of actions A_{s1} A_{s2},…A_{sn} that , if applied orderly to the starting configuration, lead the system to the goal configuration. This means that a strategy can be described as a subset of the condition-action array of the puzzle (Table 1), in which to every condition it corresponds one and only one action. A strategy can therefore be considered as a selection of the condition-action array (Table 1), i.e. a list of "instructions" of the form condition-action; of course *effective* strategies must be composed of instructions that allow players to achieve the goal (more or less quickly), but the definition comprises also strategies that do not achieve a goal.
It is interesting to remark that the definition of *strategy* is equivalent to that of *program* in computation theory. In fact, a strategy as just defined is nothing but a list of instructions of condition-action type, selected in such a way to achieve the goal starting from an initial configuration: which is precisely the definition given to 'program' in computation theory.[6] The most important difference with the general definition of a program like a computing machine (Turing Machine, Post System, or equivalent mathematical devices) is that here the machine is defined as a list of instructions of the type condition-action and therefore is an automaton, which is computationally less powerful than a Post System. Holland's classifiers, that are computationally equivalent to Turing machines, are based on more powerful condition-action systems. (Holland, J. H. *et al.* , (1988))

2.1 Abstraction and Categorization

So far, we have considered puzzles in which one starting state and one goal state are defined. But it is useful to drop this restriction, especially if we consider that in the most known puzzles and games the definitions of the starting and goal states are given with some degree of abstraction, and therefore there are in general many starting states and many goal positions. In Chess, for example, the goal configuration is defined in abstract terms, as any configuration in which the King is under attack whatever move he makes; in Rubik puzzle a state is given by the positions of the colored tiles along the faces, and the moves are rotations that can be applied whatever the game configuration of the puzzle will be. And so on and so forth.

The simplest way to classify a set of states in a category is by using the symbol # (don’t' care): a state is in general represented by a set of atomic elements or variables (in Rubik the distribution and the colors of the tiles of the faces, in Chess the board positions of the pieces, and so on). Therefore while a specific state Sk may be represented by attributing a value to each of the atomic elements (Sk = {v1 v2…vn}), at a higher level of abstraction we can define states in which some of the atomic elements may assume any of the legal values. For example, by writing {v1 # v3… vn} we "do not care" of the value of the variable v2, and define a class of states, the states that identical for all variables but v2. The level of description of a problem, in which all states and rules are specified, is called *ground level*.

In many games there is a large variety of starting configurations, sometimes given at random, an abstract definition of the goal configurations, and the rules of the game are given with some degree of generality. If states, rules, starting and goal configurations are given with some degree of abstraction, problems and sub-problems are defined as pairs of sets of states, and to solve a problem means to discover a strategy for achieving one of the goals, for any given starting configuration.

It is easy to recognize that this way to define states and rules holds in all areas of problem solving activities: problems, states and rules are defined at some abstraction level. A problem may therefore be defined by a pair of *sets*, the set of starting state and the set of goal states, given the rules that apply to the states.

2.2 Sub-problems identification

A classic procedure to decompose a problem in sub-problems is based on the construction of an and-or tree of sub-problems (Newell and Simon (1972), Nilsson, (1971)).

Here it is sufficient to outline briefly the procedure's principles. To decompose a problem P we have to identify sub-problems and connect them each other in such a way to reduce the original problem to a combination of sub-problems. Suppose that we split the original problem in N sub problems, P_1, P_2, …P_N; there are two alternative ways in which a problem can be splitted.

First, it may happen that to solve the original problem P we have to solve P_1, *and* P_2, …*and* P_N .This is called an <u>*and*</u> decomposition.

Second, it may happen that to solve P we have to solve *either* P_1, *or* P_2, … *or* P_N. This is called an <u>*or*</u> decomposition.

An elementary problem is a problem that is immediately solvable by applying one of the rules (actions) of the game.

The decomposition procedure works as follows: suppose that we split the original problem in N sub problems, P_1, P_2, …P_M (<u>*and/or*</u> decomposition). It may happen that P_1 is an elementary problem, otherwise it has to be in turn decomposed in sub-problems, P_{11}, P_{12}, …$P_{1M'}$ and so on recursively. The same holds for P_2, …P_M; therefore the process of decomposition continues until all sub-problems are reduced to elementary sub-problems or some sub-problem proves unsolvable. The elementary sub-problems at which the recursion process stops are also called *terminals*.

Any *solvable* problem can be therefore decomposed in a set of *and /or* sub-problems. With the usual conventions of graph theory we can represent the problem as an and/or tree of sub-problems. An example of decomposition tree is provided in section 2.6
Once we have successfully decomposed a problem, at the same time we have identified a strategy to solve it. In fact call P_{A1} ,P_{A2} ……,P_{AN} the elementary sub problems (terminals); by definition each of them is represented by a pair $\{S_{Ak}\ S_{Ah}\}$ of states that are connected by an action A: $P = S_{Ak} \rightarrow A \rightarrow S_{Ah}$; therefore instead to use the pair $\{S_k\ S_h\}$, we can represent a terminal state as a pair state-action $S_{Ak} \rightarrow A$.
Here again the states are normally specified at some level of abstraction, i.e. can be described by strings of the type $S_{Ak} = \{v1\ v2\ \#\ \dots\ \#\ vn\}$, and in consequence the whole set of terminal problem can be represented by an array of the type illustrated in Table 3.

Table 3

A Strategy as set of terminal problems									
(Terminal States)									Actions
S1	=	#	vi	….	#	vj	vh		Ai
S2	=	vk	#	….	vr	#	vt		Aj
S3	=	vs	#	….	vj	vx	#		As
…..		….		….					
SN	=	vb	vx	…	#	#	#		Ar

The array contains all terminal states, and therefore represents in a very natural way the strategy that the players have discovered. The terminal sub problems in fact are all sub-problems identified during the problem decomposition that can be solved with an elementary action, and therefore starting from one of the initial configurations, the player can apply sequentially the actions prescripted by the terminals until the goal is achieved.
Note that a large part of the terminal sub-problems are in some abstract form: should all terminal sub problems be defined at ground level, the list of the statements would be enormous, and the advantage of decomposing the problem would vanish.
A decomposition is therefore a way to build up a hierarchical system of abstractions. The states of the system that are involved in the strategy are classified in categories, according with the decomposition principles, and to each category it is attached the action to be done. Of course the lower is the number of categories, the simpler the representation of the problem.

Remember that if the starting and goal configurations are unique, individual problem solving activity consists in a search for a sequence of actions that, applied to the starting configuration, permit to achieve the goal. In general the starting configurations and goal configurations are numerous, and the problem is to determine if there are classes of starting configurations that lead to classes of goals configurations *with the same sequence of actions*.
But of course this property, which is highly desirable, does not hold in general: it can easily happen that it is impossible to connect all the configurations of a starting class with the configurations of a goal class by using the same sequence of actions. More interestingly, it may happen that it is possible to connect the two sets with the same sequence, but the sequence is optimal (the minimal) only for some configurations, not for all of them. Here is the origin of the biases.
To better understand the question, it is useful to compare a "natural" decomposition with an optimal one.

2.3 In search of an optimal solution

At the ground level, an optimal solution of a puzzle is a minimal path connecting a starting configuration with the goal. A well known procedure to find a minimal path between two states, it is based on the idea of moving back from the goal to the initial configuration. Starting from a state that has been defined as goal, we can move back to its predecessors, and so on recursively until a starting configuration has been achieved. In this way we discover the minimal path from a state and the goal, if it exists. A bit more in detail, the backward procedure to identify the minimal paths, i.e. paths with the lower number of steps to connect a starting state with the goal, works as follows: starting from a goal configuration G, we can find all his "predecessors”, i.e. all the states (Table 2) that have G as successor. This can be easily done (Table 2) by selecting the rows of Table 2 in which the G state appears. Call X the set of predessors of G. Next we build up the set X’ of configurations that are predecessors of the elements of X (and do not belong to G). Next we continue back including in X" the predecessors of X' (provided that do not belong neither to X' nor to G) and so on until all states of the puzzle are achieved. It is thus possible to reconstruct all links between the final configuration, his predecessors, the predecessors of the predecessors and so on, and to classify the predecessors for the distance (number of steps) from the goal configuration. So doing we build up a tree directed to G. The set B=XUX'UX"… is the basin of attraction of the goal G.
If the problem or the puzzle has been defined at some level of abstraction, we may expect that the goal configuration is described at an abstract level too and therefore there are many configurations at ground level that match the goal requirements as it happens for example in Chess. In this case we can easily extend the backward procedure to all goal configurations, and discover the global basin of attraction. Of course if there are states in the set of starting states that do not belong to the global basin of attraction this means that for these states the problem cannot be solved. For obvious reasons we limit the discussion to solvable puzzles, or to puzzles defined on domains on which are solvable.

Suppose that a goal configuration is given at some abstract level {# # s_x # s_h # s_w} where s_h are as usual atomic values. We can find the predecessors of this configuration at the ground level, and then try to compact them in few classes. In general we will identify many different classes each of which is connected with the goal by a different rule of action; we therefore compact the predecessors in sub-classes X_1, X_2,...X_w in relation to the action’s rules that must be applied to achieve the goal. So, if we call A_1, A_2,..A_w the action’s rules that connect the first level predecessors to the goal G, we have $X_i \rightarrow A_i \rightarrow G$ (i=1,..w)). So doing, we can identify the classes of all first level predecessors, and iterating back the same procedure we identify the tree of the optimal strategy.
Of course there are no reasons why a class X_i could be *described* by one only string at abstract level: each set X_i is *defined* on the basis of the action rule that connect it to the goal but is *described* succinctly on the basis of some abstraction (individuals cannot memorize huge sets of elements, and must save on representation and memorization). A good description reduces the number of ground configurations to be considered to a small number of abstract configurations [7]. If we are not lucky, it may happen that does not exists any level of abstraction for describing the set of predecessors, and we have to describe it at ground level. If we are lucky, we will find one abstract string $S_1$to describe X_i. In general we may expect to need of many strings, S_1, S_2,...S_r to describe X_i.
The central point is therefore that in general if we identify a set X_j to which an optimal rule is applicable to obtain a set Y_j, the description of the set X_j cannot be reduced to a single abstract string, but it may require many strings. Therefore we cannot describe the solution of an elementary sub-problem with a simple abstract condition-action statement, within the given representation of the problem, but we need a more complex description. (In Johnson-Laird language, a “mental model”, in Simon approach, a “chunk”).

Vice versa, suppose that an individual identifies a specific sub-problem, and discovers an optimal action A: if he tries to apply the action A to a generalization of the sub-problem, obtained by extending by default the original domain of applicability, he may introduce a systematic error. In fact the domain of applicability of the rule A - which was optimal if applied to the original configuration - does not necessarily coincide with the domain of optimality. A may be sub-optimal if applied to some configurations that belongs to the extended domain. As a consequence, when individuals try to extend by default generalization a rule that was optimal in a given context, they may extend the rule beyond the domain of optimality and therefore introduce hidden errors.
In the next session we describe in detail full example of this phenomenon, in the context of a hyper simplified Rubik cube game.

After a full process of decomposition, by a backward procedure, we create a hierarchy of connected sub problems, at different levels of abstraction. The decomposition is optimal, because it identifies the minimal tree, and therefore the minimal number of actions to be made to achieve the goal. The key point we have focalized is that we do not have, in general, a “perfect representation” of the decomposition (terminal) units X_i i.e. a representation in which every X_i is described by one only abstract string.
When we decompose a problem, we conjecture or propose the atomic sub-problems, as "basic dimensions" of the problem, and implicitly we define the degrees of abstraction of these dimensions. This explains why a systematic mismatch exists between the *optimal* sub-problems and the *conjectured* sub-problems. If we do not create new codifications, for example taking into account the relations among the atomic components, the “basic dimensions”, i.e. the terminal sub-problems that individuals discover do not identify perfectly the optimal sub-problems, and induce individuals in imperfect extrapolations and abstractions. [8]
Concluding this section, note that the level of abstraction with which the strategy’s statements are represented, depends upon the way players analyze the problem, which in turn depends upon the way the problem is codified. Individuals may try to find out a satisficing or an optimal decomposition, on the ground of a given representation, or may try to recodify the problem and change the representation himself.
Within a given codification, it is clear that there are many different decomposition patterns, and that humans do not and cannot easily discover the optimal one: since an optimal decomposition can only be designed by someone who has a perfect knowledge of the problem, bounded rational individuals can only discover satisficing decompositions and try to improve them progressively.
One of the typical ways to identify a decomposition pattern, mainly for repeated games in which players have to learn a strategy while playing, is the attempt to induce general rules from examples. Induction in this sense (Holland *et al.*, 1988) is the fundamental force that drives, via abstraction, classification and extrapolation, the categorization and the creation of a representation of a strategy.[9]This process is very powerful, and has some similarities with the process with which kids learn the language: it is astonishing that a three year old baby is a perfect speaker of his native language, without knowing anything of the syntactical and grammatical rules. His ability to extrapolate general rules from examples is the key element of his growing linguistic skill.
During the identification of the basic sub-problems, hidden errors are created and included in the representation, because individuals identify sub-problems on the ground of a given representation, that induces to extrapolate the rules they have learnt beyond the domain of optimality. Biases emerge therefore as the outcome of the process of classifying and categorizing the rules of action, accordingly with a decomposition pattern.

2.4 Minirubik

In this section I examine in detail a simple puzzle, by computing the optimal strategy and comparing them with a simple strategy obtained by decomposition of the problem. I show how "errors" are created during the decomposition analysis.
Let us consider a simple puzzle based on an extreme simplification of Rubik’s cube. The player has a square consisting of four differently colored tiles denoted with the letters A,B,C,D.

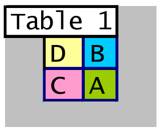


Table 1

The tiles can be exchanged horizontally or vertically in the ways shown in the following figure.

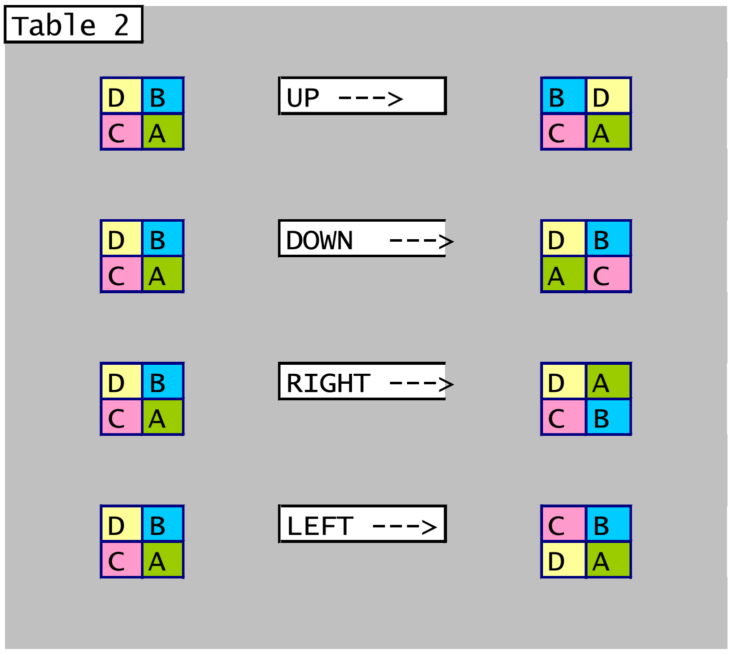


Table 2

Let us suppose that a tournament is held in which the initial configuration for every round is selected at random, and for the sake of simplicity let us assume that the final configuration is fixed as the following sequence:

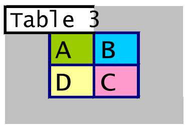


Table 3

The players must exchange the tiles until they have produced this final configuration, and they are rewarded according to the number of moves that they make to achieve it, although a maximum time limit is set for each game. In this context a problem at ground level is defined by a pair of game configurations, the starting one and the goal one, as in the Table 4.

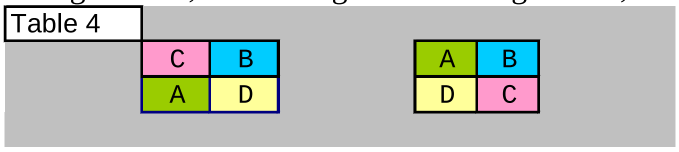
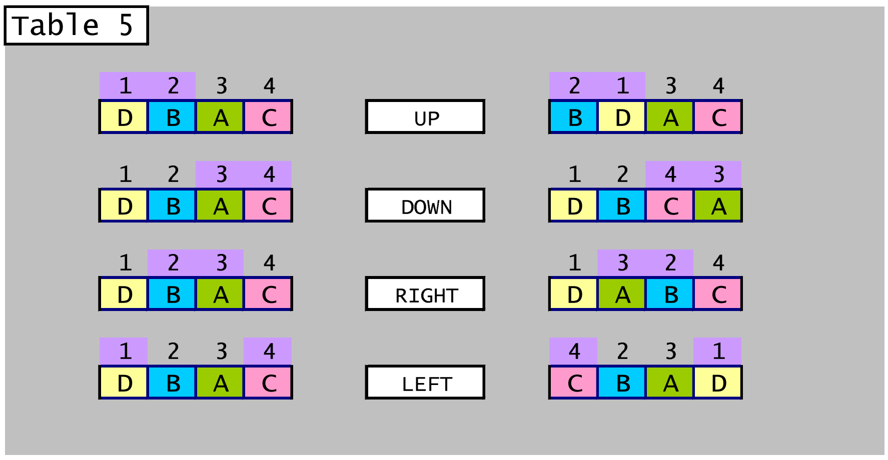


Table 4

A less natural but more concise representation of the puzzle is as follows:

Table 5



1	2	3	4		2	1	3	4
D	B	A	C	UP	B	D	A	C
1	2	3	4		1	2	4	3
D	B	A	C	DOWN	D	B	C	A
1	2	3	4		1	3	2	4
D	B	A	C	RIGHT	D	A	B	C
1	2	3	4		4	2	3	1
D	B	A	C	LEFT	C	B	A	D

In this formulation, the rules can be recast as exchanges among tiles in horizontal strings and stated as follows:

Up: exchange between adjacent tiles in positions 1 and 2;
Down: exchange between adjacent tiles in positions 3 and 4;
Right: exchange between adjacent tiles in positions 2 and 3;
Left: exchange between the tiles on the extreme left and extreme right (positions 1 and 4).

Let us now suppose that we want to expand an initial configuration of the game to create the successors, i.e. the new configurations produced by all possible moves applied to the initial configuration. By applying orderly moves U, D, R, L to all 24 different configurations of the game we obtain a list of states, called "successors". In Table 6, listed to the right of every configuration are the four successors; of course any successor is again a configuration of the game. This table is the *links array* and contains all the links between each state and its successors. We can use the links array to discover some of the paths that begin with a given configuration and finish with the target configuration. (There is obviously a countable infinity of such paths).

Table 6

	1	2	3	4
1	A	B	C	D
2	A	B	D	C
3	A	C	B	D
4	A	C	D	B
5	A	D	B	C
6	A	D	C	B
7	B	A	C	D
8	B	A	D	C
9	B	C	A	D
10	B	C	D	A
11	B	D	A	C
12	B	D	C	A
13	C	A	B	D
14	C	A	D	B
15	C	B	A	D
16	C	B	D	A
17	C	D	A	B
18	C	D	B	A
19	D	A	B	C
20	D	A	C	B
21	D	B	A	C
22	D	B	C	A
23	D	C	A	B
24	D	C	B	A

Links' Array

	U	D	L	R
1	7	2	22	3
2	8	1	16	5
3	13	4	24	1
4	14	3	10	6
5	19	6	18	2
6	20	5	12	4
7	1	8	20	9
8	2	7	14	11
9	15	10	23	7
10	16	9	4	12
11	21	12	17	8
12	22	11	6	10
13	3	14	19	15
14	4	13	8	17
15	9	16	21	13
16	10	15	2	18
17	23	18	11	14
18	24	17	5	16
19	5	20	13	21
20	6	19	7	23
21	11	22	15	19
22	12	21	1	24
23	17	24	9	20
24	18	23	3	22

2.5 Identifying the optimal strategies

We have still described the backward procedure to identify the minimal paths. Here we only apply the procedure. We start by identifying on the array of links (Table 6) the set of configurations X(1) from which is possible to achieve the final configuration 1 in one step. To do this, we identify on the right side of the Table 6 the rows that contain the number 1: these are rows 2,3,7 and 22, and therefore we have the set is X(1)={2, 3, 7, 22}. We then identify the set X(2) of configurations from which is possible to achieve an element of X(1) in one step. The reader can verify that X(2)= {4, 5, 8, 9, 12, 13, 16, 20, 21, 24 }. Repeating this procedure recursively, we obtain X(3)= {6, 10, 11, 14, 15, 18, 19, 23 } and finally X(4)={17}.

We have therefore found all the links that connect all the configurations of the game to the final configuration 1 with a minimal distance; if we rewrite the connections we have identified on the array of links, and cancel (or cross) the links which have not been involved in the backward procedure, we obtain the section A 'minimal connections' of Table 7. The Table tells us what are the minimal paths between any configuration and the goal configuration, 1.

It is clear from the table of links that there are many minimal paths connecting a state with the goal; for example if we want to find the minimal paths from state 11 and 1, we find the paths 11→21→22→1, 11→12→22→1, 11→8→2→1, 11→8→7→1, all of which have the same minimal distance from the goal 1.

In the center of Table 7, I have reported the corresponding optimal actions; so for example the path 11→21→22→1 can be followed by applying the actions U,D,L : 11→U→21→D→22→L→1. On the right side of the Table 7, I have selected one optimal strategy, and computed the sequences of actions that lead to the goal applying to every configuration the optimal strategy.

Table 7

A - MINIMAL CONNECTIONS									B-OPTIMAL Action				C - MINIMAL PATHS		
Conditions					U	D	L	R	Actions				Action	Path	Length
A	B	C	D	1	~~7~~	~~2~~	~~22~~	~~3~~							
A	B	D	C	2	~~8~~	1	~~16~~	~~5~~		D			D	D	1
A	C	B	D	3	~~13~~	~~4~~	~~24~~	1				R	R	R	1
A	C	D	B	4	~~14~~	3	~~10~~	~~6~~		D			D	DR	2
A	D	B	C	5	~~19~~	~~6~~	~~18~~	2				R	R	RD	2
A	D	C	B	6	20	5	12	4	U	D	L	R	D	DRD	3
B	A	C	D	7	1	~~8~~	~~20~~	~~9~~	U				U	U	1
B	A	D	C	8	2	7	~~14~~	~~11~~	U	D			U	UD	2
B	C	A	D	9	~~15~~	~~10~~	~~23~~	7				R	R	RU	2
B	C	D	A	10	16	9	4	12	U	D	L	R	L	LDR	3
B	D	A	C	11	21	12	~~17~~	8	U	D		R	R	RUD	3
B	D	C	A	12	22	~~11~~	~~6~~	~~10~~	U				U	UL	2
C	A	B	D	13	3	~~14~~	~~19~~	~~15~~	U				U	UR	2
C	A	D	B	14	4	13	8	~~17~~	U	D	L		U	UDR	3
C	B	A	D	15	9	16	21	13	U	D	L	R	D	DLD	3
C	B	D	A	16	~~10~~	~~15~~	2	~~18~~			L		L	LD	2
C	D	A	B	17	23	18	11	14	U	D	L	R	L	LRUD	4
C	D	B	A	18	24	~~17~~	5	16	U		L	R	L	LRD	3
D	A	B	C	19	5	20	13	21	U	D	L	R	U	URD	3
D	A	C	B	20	~~6~~	~~19~~	7	~~23~~			L		L	LU	2
D	B	A	C	21	~~11~~	22	~~15~~	~~19~~		D			D	DL	2
D	B	C	A	22	~~12~~	~~21~~	1	~~24~~			L		L	L	1
D	C	A	B	23	~~17~~	24	9	20		D	L	R	D	DLR	3
D	C	B	A	24	~~18~~	~~23~~	3	22			L	R	L	LR	2
															52

So far we have formatted the set of optimal (and equivalent) strategies as a condition-action table in which for every condition we have identified one or more optimal actions.
We can therefore measure the (minimal) path length for every configuration; the sum of all minimal distances from any configuration and the goal is a measure of the global performance of the strategy. It can be straightforwardly checked that the minimal length is 52 steps. Of course, any strategy obtained selecting one action for the second column (Actions) of the Table is allowed. We can therefore build up a repertoire of many different sets of rules, each of which is a complete optimal solution and requires 52 moves to achieve the goal from all starting configurations. The possible combinations of the optimal actions are 331776, which means that the game admit 331776 different optimal strategies with the same fitness (f=52).

2.6 Representations and biases: an example: the "First A" strategy

It is obvious that players do not discover the solutions by backward search, and the important point is to understand how they reconstruct a good or optimal strategy, and why they make errors. In some experiments with MiniRubik, a number of players carried out a very simple, “ingenuous”, strategy, based on the attempt to put every square of the board in his right place, disregarding the interferences among the different actions. This strategy can be easily described and formalized via decomposition of the problem in sub-problems.

Assume as before that the goal of the game is to achieve the configuration ABCD, whatever the starting configuration will be. The strategy proposed by players, applied to a randomly chosen configuration, prescribes to start moving A to the first position, then moving B to the second position, etc. until the goal configuration (Table 8) has been achieved.

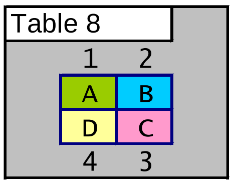


Table 8

This is a “conjectural” approach to the problem, in the sense that “assumes”, without any evidence, that it will be possible to move orderly A, B, C and D in their final positions without any interference among the goals. In other words, it is assumed (ingenuously) that it is possible to shift a piece in his right position without moving away other pieces from their right positions, i.e. that the goals can be achieved independently.

To write this strategy in form of instructions (or condition-actions) we have to decompose the problem into three sub-problems:

(P1) " Move A from the initial position in the string (whatever it is) to position 1" :

(####) → (A###) .

(P2) "Move B from his present position (eventually scrambled after step 1) to position 2 *leaving* A in the first position":

(A###) → (AB##).

(P3) "If C and D are not in the required positions 3 and 4, exchange them":

(AB##) → (ABCD)

The three statements P1, P2 and P3 are a "conjectural strategy", i.e. a proposal for a solution of the problem that embodies a decomposition of the problem without proving that the sub-problems can be solved independently;
Therefore we have to reduce the Prescriptions P1,..P3 to specific elementary actions, by decomposing every prescription into more elementary prescriptions recursively until we reduce to elementary actions.
Starting from statement P1, we may note that the movement of A toward position 1 is a “circular” movement. A reasonable elementary rule is to move A following the “closest” distance to position 1; this means that if A is in position 2 it is natural to move it counterclockwise while it is natural to move it clockwise if it is in position 4. This rule does not give any suggestion if A is in position 3, because of course the “distance” between positions 3 and 1 are the same both if we move clockwise or counterclockwise.

Therefore we can reduce (arbitrarily) P1 to the following sentence: "Move A clockwise if it is in position 4, counterclockwise if it is in position 2 and 3 ". Call P11 this specification: it is now possible to rewrite this statement as an organized sequence of actions, to be performed for every different position of A in the board.

P11 is therefore rewritten as the sequence of OR actions

```
if   (#A##)    then → U → (A###)
if   (###A)    then → L → (A###)
if   (##A#)    then → R → (#A##)→U→ (A###)
```

I we apply the same method to P2 and P3 , we obtain the corresponding sequences of conditions-actions. The global problem can therefore be de-structured and reduced to a sequence of elementary actions as clarified in the following table.

Table 9

```
                                              | (#A##) → U →(A###)
                                              | (###A) → L →(A###)
                        | (####) → (A###)→    |
                        |                     |                       | (##A#)→ R →(#A##)
                        |                     | (##A#)→(A###)→        |
                        |                     |                       | (#A##)→ U →(A###)
                        |
(####) →(ABCD)          |                     | (A#B#) → R →(AB##)
                        |                     |
                        | (A###) →(AB##)→     |                        | (A##B) → D →(A#B#)
                        |                     | (A##B) → (AB##)→       |
                        |                                              | (A#B#) → R → (AB##)
                        |
                        | (AB##)→ D → (ABCD)
```

The strategy is now fully described, via a tree that represents the original problem decomposed in a structure, i.e. as a set of interconnected elementary sub-problems that are either elementary ones or can be reduced to elementary ones.

A different way to represent the strategy, is the array of conditions – actions sketched below. To build up the array I have simply written a list with the terminal nodes of the previous tree, nodes that are the elementary problems solved.

Table 10

FirstA Strategy

Conditions				Number	Action
A	#	#	B	1	D
#	#	#	A	2	L
A	#	B	#	3	R
#	#	A	#	4	R
#	A	#	#	5	U
A	B	D	C	6	D

Having transformed the three statements P1,..P3 in the format of a decision Tree, and equivalently in an array of Condition-actions, we can now compare the *conjectural* strategy with the optimal strategies we have previously computed with the backward algorithm.
It is now clear what is the consequence of designing a decomposition of the problem accordingly with a conjecture on independence among the sub-problem: the sub-problems we have identified are not perfectly independent.
The optimal solution of Rubik exists, and differs from the ones discovered via decomposition of the problem.

2.7 Optimality and sub optimality: hidden errors

By comparing the conjectural solution (Table 10), with the optimal one (Table 7), we can realize that the domains of applicability of the sub-problems are not perfectly independent:

Table 11

Optimal Strategy						Conditions involved (Table 5)
Conditions						
A	#	#	B	1	D	4,6
#	#	#	A	2	L	10, (12),16,18,22,24
B	D	C	A	2*	U	
A	#	B	#	3	R	3 ,5
#	#	A	#	4	R	9,11, 15,17, (21),23
D	B	A	C	4*	D	
#	A	#	#	5	U	7,8 13, 14, 19, (20)
D	A	C	B	5*	L	
A	B	D	C	6	D	

In fact the general rules of Table 10, are not optimal if applied to their entire domains; more precisely, we have that the rule n. 2 "if A is in position 4 then move it Left" is valid for all conditions in which A is in position 4 *excluding* one special condition, i.e. the string B D C A, for which we have to add a new rule (a specification of rule 2), i.e. the rule 2*. (The same holds for rules 4 and 5). It follows that the goal " move A from position 4 to position 1 " cannot in every case be separated from the goal "move B to his position (position 2)", because the action to be realized to accomplish the two goals interferes in the specific condition BDCA: in fact in these conditions if we try to get A in his final (goal) position, at the same time we get B farther from his goal position; so in this particular case the conjecture " I can first move A to his final position then move B independently" fails, if we require to accomplish the goal in a optimal way. Table 11 is therefore an optimal condition-action table, that differs from the FirstA strategy (Table 10) for the three statements 2*, 4* and 5*, which are "specifications" that conflict with the general rules 2,4 and 5.
In MiniRubik the major part of the combinations allow the player to move each piece in the direction of his right position without regards of what happens of the other pieces, simply because the interaction exist and is positive, i.e. getting a piece closer to his right position, the player gets at the same time closer to his right position some other piece. This positive interaction is violated in two particular situations in which goals are negatively interrelated.

For example CADB is solved by UDR: CADB→ U→ ACDB→D→ACBD this action gets D in his right position and at the same time gets B closer to his right position.
If on the contrary we attempt to solve BDCA with the prescription of FirstA strategy, we get LDRD: BDCA→L→ADCB→D→ADBC→RABDC→DABCD, which is of course inefficient, while the game can be solved with two moves, UL, only: BDCA→U→DBCA→L→ABCD. The first move prescribed by FirstA strategy in fact gets at the same time A on his right place, but B farther from his right position.
The space of the problems generated by Minirubik is fully defined by table 7 in which we have reconstructed all minimal paths to achieve a solution. When an individual is at the beginning of his experience with the game first he have to learn from examples, and the attempt to generalize specific situations may be misleading and prevent players to identify optimal solutions. So some players have done: in a preliminary experiment, some of them have followed the FirstA solution, which is fully covering every situation, is perfectly consistent, but unfortunately does not take into account a fundamental property of the game: i.e. that when players move a piece, getting a piece closer to his right position, sometime he gets at the same time closer to his right position some other piece, but sometime he pushes away a piece from the right position. Therefore a few errors have inadvertently included in the action rules.

3. Applying to puzzles and problem solving the Nk model

We have seen that there are many different, both optimal and sub-optimal, ways to decompose and represent a problem. The related question is the stability of the representations. From *Target The Two* and other experimental contexts we have some evidence of the persistency of sub-optimal solutions of given problems. The question is to explain why sub-optimal strategies, that embody errors, are locally stable, i.e. cannot be improved by small changes, by "correcting" the specific errors, i.e. by mutations of specific strategy rules.
The most important candidate to tackle the problem is Kauffman's NK model, that allows us to represent the learning of a strategy as an evolutionary process of rules mutation and selection.
The NK model has the following main features: the evolution of an organism, or in general of a complex biological system, is guided by its 'fitness', that is, by its reproductive success in the environment; the characteristics that determine the fitness of an organism can be represented in a discrete space because they are a list of 'traits' that can assume different values.
In Kauffman's originary approach 'traits' can be proteins or genes, each of which can assume different "configurations" or "values" (alleles). An organism is characterized by N traits, each of which assumes a given value. A mutation is nothing else than a change in the value of a trait (allele), and therefore to explore the effect of single mutations on the organism fitness we have to change the values of the traits, one at the time. A crucial property of the traits is 'epistasis': when a mutation is introduced, it normally happens that the effect on the organism' fitness depends on the values of other traits.
"The assumption that each gene contributes to overall fitness independently of all other genes is clearly an idealization. In a genetic system with N genes, the fitness contribution of one or another allele of one gene may often depend upon the alleles of some of the remaining N-1 genes. Such dependencies are called *epistatic interactions* " (Kauffman, 1989, p. 539).
Call K the average number of genes that contribute to the fitness variation of the organism when a mutation occurs, i.e. the average number of epistatic interactions. K may vary from K=0 (total independence) to K=N-1 (total interdependence). In the first case, (K=0) the effect of a mutation on the fitness depends solely upon the single gene that is affected by the mutation; therefore by comparing the different effects of different mutations on the same gene, we can discover the allele

that produces the higher increase on the fitness. If we sequentially do the same comparison on all the genes, we can discover for every gene the alleles that make the best contribution to the fitness; therefore we can increase the fitness of the organism until his maximum value, acting on every gene independently. This means that an organism with zero epistatic interactions may achieve an optimal configuration in response to a sequence of random mutations.
Kauffman shows that as the epistatic interaction grows, the number of local optima increase, and an organism, which is affected by mutations, once it has reached a local optimum, may remain trapped forever.
It is natural to understand that the NK model can be applied to every complex system, and therefore it can be useful to describe the properties of complex organizations. The only *caveat* is to remember that in NK model it is assumed that epistatic interactions are so complex, that the statistical features of their consequences are modeled by random fitness functions. With this assumption, Kauffman shows, among other properties, that the basins of attraction of local optima of low K systems are larger on average, than the basins of high K systems. Moreover, the higher their fitness, the larger their basin of attraction. One of the consequences of these properties, as Kauffman suggests, is that systems with low K value are expected to outperform systems with high K and therefore are more likely to be selected during evolution.
The most important properties of NK model concern features deriving from complexity, and in particular from the decomposability of complex systems. Among the recent developments on this field, important contributions have been given by Page (1996) and Marengo (1998,1999). The latter author provides a definition of independence among the sub-systems of a complex entity, and explores the properties of near-decomposability of the systems. The idea is that to optimize the fitness of a complex system, we can try to optimize separately parts of them. This is possible if the contribution to the fitness of any part is independent from the contribution of the others. If the elements of the complex system are K "epistatically" interdependent, this means that K of them on average jointly contribute to the fitness, and a complex system is therefore decomposable in sub systems of the K average size, each of which can be optimized separately.
At the extreme situation, K=0 and then the single elements of the system contribute separately to the fitness: in consequence the discovery of the configuration with optimal fitness may be realized by modifying the alleles independently from each other.
On this basis, we can consider each element of a string of conditions-actions as a genetic trait, and therefore try to apply the NK model to the world of puzzles. Referring to Table 1, a pair condition-action can be considered as a gene, and the different possible actions triggered by a "condition" give rise to different alleles of the same gene.
We may therefore treat a puzzle as an Nk model and examine its properties – in particular those that concern 'ruggedness' of the fitness surface, i.e. the distribution of local and global optima in the space of the fitness function. (See also Kauffman (1989) pp. 680-690)

The Nk model is intended to analyze properties of complex systems and has been used for this purpose in many fields, beyond the specific field of biology in which has been created. (See for example Levinthal (1997)). If we drop the assumption, which is typical of the originary context, i.e. to consider the fitness of the system as a random function, and introduce a measure of efficiency of the system, some of the Nk model's properties do note hold anymore. In Appendix it is shown in fact that in puzzles K>0, i.e. that the elementary statements of a strategy are interdependent, because introducing a different rule of action for a given system configuration (a mutation) - the fitness of the system may get better or worse in relation to the other rules adopted for the strategy. But, despite the fact that the epistatic degree K is positive, it is shown that it is possible to achieve an optimal configuration of the strategy via mutations and in consequence the sub-optimal solutions are not locally stable under mutations.

These properties hold if one assumes the fitness as a variable that measures the "efficiency" of the system. The system here can be identified as a strategy or a program, and is very natural to measure the efficiency of a program in relation to the number of steps it requires to achieve the goal. Given a starting configuration Sj, the shorter is the number k of actions $A_{j1}, A_{j2}, .. A_{jk}$ with which the goal is achieved using P, the higher the efficiency of the program. When there are many starting configurations it is natural to extend the definition of efficiency of the strategy P in relation to the sum of the lengths of the chains connecting the different starting states with the goal, obtained applying P. It is obvious that the problem is to find the strategy P* that minimizes this length. Therefore we can choose as efficiency any function f(P) that is monotonically decreasing with the global length of the chains connecting the starting configurations with the goals. A simple example is given by f(P)=Ns-L, where L is the global length of the chains and Ns the global length of the network defined by the links' array. [10]

3.1 Cycles in random strategies [11]

Instead of considering a strategy as the result of a conscious process of decomposition of a problem into sub-problems, we could investigate whether a strategy can be the result of an unintentional adaptive process through which the system reaches an optimal configuration by ways of successive mutations.
In order to deal with this point, it is first necessary to answer the following question: is it possible to determine a strategy by randomly choosing one action for each condition of the matrix that describes the puzzle (Table 4)? How can we be certain that we will successfully reach the final goal of the game through a strategy based only on a random selection of the legal rules?
We have to check whether, once we have randomly chosen a sequence of actions from the game matrix, we get a list of conditions-actions that constitute a "consistent" strategy; i.e. a strategy that it will allow us to reach the goal regardless of the starting configuration. Here "consistent " refers to the player's conduct when he conjectures a decomposition of the problem in sub-problems. If a rule of the strategy conducts from the state A to state B and at the same time the strategy comprises a set of rules which conduct from the state B to the state A, this creates a "vicious circle", that denotes that the player has been inconsistent in decomposing the problem. Rules giving rise to cycles must therefore be excluded from a strategy, if strategies are to be effective, i.e. to allow players to reach the goal.[12]
It is convenient to remark that the set of condition-action rules that describe a game, (Table 4) can be equivalently represented by a graph, in which every condition is a node, and every action is an arc connecting two nodes. Having considered as *consistent* a strategy that allows us to reach the final stage regardless of the starting configuration, the selection of a consistent strategy means choosing the rules in such a way that from every node there exist only one path directed to the goal node; this set of paths directed from the nodes to the root forms a directed tree. In other words, a consistent strategy is a selection, within the graph, of a tree rooted in the final goal and connected with all configurations.
This kind of tree is usually called spanning tree of the graph: a *spanning tree* of a graph G is a tree containing all vertices of G. It is well known (Berge 1985, pp. 22, 24) that if a graph is connected, then it admits a spanning tree. There are several ways to build up from a connected graph a spanning tree rooted in the goal node. Many of these trees will be highly sub-optimal, and in fact the problem of finding an optimal strategy it is equivalent to the problem of finding a minimal tree.
By definition, a spanning tree is not directed. To describe an effective strategy, we require something more restrictive that a spanning tree, i.e. a spanning tree in which every arc is directed to the root.

Before discuss of tree optimization by mutations, we need to see what happens with a random strategy. It is likely that, if the action rules are selected at random, quite often the selected links do not give rise to a tree, but to a graph with cycles, and that as a result the strategy will not allow the final objective to be reached from all starting configurations. The graph that represents the random strategy in this case would be made of disjointed cycles: starting from a node and applying the rules of the strategy, we could fall in a cycle without the possibility of achieving the goal.
This is very likely to happen, especially in games where, given an action, it is allowed to make also the reverse one, as in MiniRubik: in this game if you can reach condition Y from condition X through action A, there is also the possibility to go from Y to X through a symmetric action. In a random selection, both actions could be selected and that would create a cycle (an inconsistent strategy).
It is possible to show that if the strategy randomly chosen does not allow cycles, it is formed by a spanning tree rooted in the goal, in which all nodes are directed to the goal. This means that by applying a consistent (without cycles) strategy to every starting configuration, after a finite number of steps we achieve the goal. In terms of finite state machines theory, this means that the finite machine represented by the strategy converges, after a finite number of steps to a stable point. In terms of graphs theory, this means that from every node there exists a path directed to the goal.
Summing up, to ensure that it is possible to reach the final stage regardless of the initial stage, the graph must be directed and connected. For every node of the graph there must exist a directed path connecting it to the goal node. Even though this assumption is restrictive, in relation to general graphs properties, it is the mathematical formulation of the idea that the graph represents a problem that admit a reachable solution. Chess is a good example of this situation, because it is a game in which players may be trapped in loops while a large set of reachable goal configurations exist.

3.2 Convergence to an optimal strategy via mutations

It is useful at this stage to examine the epistatic properties of puzzles, to understand to what degree strategies can be decomposed into independent components, and what are the properties of the solutions. We will establish some counterintuitive properties of the solutions: first, we will see that puzzles allow only stable optimal solutions, while stable sub-optimal solutions are not permitted. Second, we will also show that there is a positive epistatic degree, i.e. introducing a mutation on a gene, the effect on the fitness depends upon the alleles of other genes. In terms of problem solving this means that a strategy is not decomposable into independent elementary statements (the genes), but nevertheless, against the prediction of NK model, it is possible to achieve the optimal solution optimizing iteratively the elementary statements. In evolutionary terms, this means that a strategy may evolve through mutations starting from a random (consistent) configuration and reaching an optimal configuration. As it has been clarified before, we assume that the fitness function of a condition-action system is defined in relation to the length of the spanning tree of the strategy, in such a way to measure the efficiency of the strategy [13]
Assume that a strategy is given, either at random or as the outcome of a learning process, and that it is consistent, i.e. allow an individual to achieve the goal. As usual the strategy is described by a list of condition-actions rules, and equivalently by a tree directed to the goal node (or a forest directed to the set of goals). Note that a strategy is a tree extracted by the graph that represents all the possible runs of the game. The graph is equivalent to the links array of the game (Table 4), and “embodies” the strategy’s tree, exactly as the links array embodies the strategy. Therefore, if we introduce mutations in the strategy, i.e. we change some action rules, this implies that we change some nodes and the related links of the strategy, deleting some nodes and including new nodes in the tree of the strategy. More precisely, suppose to modify a strategy by introducing a mutation on one of his elements, say on the condition N: suppose that in the tree the condition (node) N was linked with the node K by the action A; now we change the action A, and therefore we delete the

successor K and link N with a new successor M by the new action A’. In sum changing a link means changing the successor of the node N. The problem is to verify if it is possible, starting from a randomly chosen tree, to modify it by mutations until it is transformed in a minimal tree (i.e. a tree whose paths from the starting configurations to the goal have minimal length).

Suppose that a certain stage of the process we improve the fitness of the strategy by removing a link between the node N and his successor K and connecting N to the new node M; this means that *at this stage of the process* M is nearer than K to the goal. But later on, after some mutations on the nodes included in the path between K and the goal, it may happen that K becomes nearer than M to the goal. Therefore we have to remove the link between N and M and link again N to K to increase the fitness. This shows that there is interdependence among the nodes, having defined the fitness in relation to the distance between the starting and the goal configurations: the epistatic degree of the system is positive, because the variation of the fitness when we change a rule depends upon the distances of many nodes from the goal.

A more detailed discussion on this point, with an example of epistasis, is reported in Appendix .

The previous observations on epistatic interdependence shows that to reduce a given tree to a minimal one, the order with which mutations are introduced *matters*. If, starting for a randomly chosen tree, we introduce mutations with a special order, i.e. the order defined by the backward algorithm, the effects of epistasis disappear. In fact, starting up the backward algorithm, the order with which mutations are introduced is the following: first, mutations are applied to the nodes (genes) that can be directly linked to the root; label these nodes as nodes of level 1; next, mutations are applied to the nodes of level 2, i.e. nodes that can be directly linked with the nodes of level 1, and so on. At every step, any new node is linked by the algorithm to some node already on the right position, i.e. at a minimal distance (level) from the root. For this reason, once a node has been linked by a mutation to some other node, it will never be removed again; the nodes are placed at the minimal distance from the root with a special order that allow avoid revisions of the links.

In general the order is different and epistasis is positive.

The revisions of the links among nodes may be needed because the path from a node to the goal has not been selected moving backward from the goal to the node, and choosing at any step the minimal distance, but moving with some other order that does not assures, when we select the adjacent node with higher fitness, that the candidates have been put at the minimal distance from the goal.

It is important to note that the backward procedure defines an order with which mutations are introduced: a special order by which it is ever possible to build up the minimal tree, provided that the graph is connected. This means that however it is chosen a strategy at the beginning of the process of mutation, the optimal strategy can be achieved by introducing mutations with the backward order. Consequently there are not locally stable sub-optimal solutions: in fact the local stability of a configuration means by definition that it is impossible to improve the fitness of the configuration by introducing single mutations.

But for every non-optimal configuration it is possible to apply the backward procedure, and in consequence to achieve an optimal configuration. This means that sub-optimal strategies are not stable, but converge to some optimal one.

Moreover, the optimal configuration can be achieved even if the order of mutations is randomly chosen, provided that all elements of the strategy are submitted to mutations until the fitness cannot be improved any more. In fact suppose that we decide to introduce mutations with a random order, say R={7,2,12,34,1,5,….} (The set R must be a random permutation of the set {1,2,3,…N}).

After a run, having applied mutations to all genes with the order defined by R, may be the fitness can be further improved; in this case, starting a second run and selecting again the favorable mutations with the same order R, we will improve the fitness again; we can continue with subsequent runs until there are no more mutations that increase the fitness. We have to show that the process converges.

To this end, suppose that the right order to intervene on genes, defined by the backward procedure, is given by {n_1, n_2 $n_{3,...}$ n_N } (n_1 is the first node to be linked to the root, n_2 is second node to be linked (either to the root or to n_1, and so on)).
Suppose now to introduce mutations with a different order. If the nodes that can be linked to the goal are {n_1, n_2 $n_{3,...}$ n_k } ($1 \geq k \geq N$), these nodes will be placed at their minimal distance from the goal after the first run. In the second run, the same reasoning applies to the nodes that can be linked with {n_1, n_2 $n_{3,...}$ n_k }, and so on. Therefore the process converges, and in the worst case after N runs all nodes will be placed at their minimal distance.
It follows that whatever order we follow to introduce mutations, the process converge to an optimal solution after no more than N runs.
The results of this section show that NK model's predictions are no longer valid if we assume a fitness function that measures the efficiency of the strategies: in fact in NK model, when K>0 and the fitness is random, stable sub-optimal solutions emerge. On the contrary, we have now shown that with our fitness, while K>0, only optimal stable solutions exist. These results have curious implications in relation to problem solving properties: the system (a puzzle) does not allow a perfect division of the problem in elementary sub-problems, because the elementary statements are not independent (K>0): but at the same time the system converges to an optimal solution via simple mutations, i.e. solving separately the elementary sub-problems.

3.4 An alternative explanation of stability of sub-optimal decompositions

We have seen that any division of a problem into sub-problems is based on a structure that is conjectured on the ground of some decomposition principles; the classifications and abstractions elicited by these principles may embody hidden errors, which cannot be perceived without close inspection. To find them, one would need to examine all the existing condition-action relations in detail. This would be an extremely time consuming task that nullifies all the advantages of a concise representation. Therefore individuals cannot actively search to detect exceptions to the rules they have established to solve the problem. We can see from the MiniRubik example that, in order to correct the errors hidden in abstract rules, it is necessary to be guided by emerging exceptions. Any attempt to discover the errors actively would pre-empt the effort to express the strategy in a simple manner.
Therefore, the locally optimal solutions in which individuals may remain trapped while analyzing a problem are created by the limits of their capacity to *falsify the rules* they have conjectured in all relevant domains, and to discover hidden errors. Just because they cannot check the effects of all mutations on the fitness, there is a very low probability that they discover mutations that improve fitness and therefore they remain trapped in the given representation.
Players do not represent in their minds all possible configurations but in general proceed by generalization on the basis of examples: this means that they try to induce general rules from specific experiences, as it happens for example in the process of learning a new language. Players therefore use whenever it is possible a generalization-extrapolation mental activity by extending as widely as possible the domain of validity of the rules they have discovered.
This procedure allow to create preliminary a complete - even highly suboptimal – strategy, in which for a large number of configurations the rules of action are defined “by extension” from similar examples. In general the procedure of extension creates systems of rules described with some degree of abstraction. Therefore the system of rules that form a strategy is complete, i.e. for any possible configuration of the game there is a rule that matches it, even for the most of the configurations the player does not have direct experience. Therefore, when a new configuration

happens that elicits a new rule as exception, in general the new configuration give rise to a new rule that is *more specific* that the default one. There is a problem in extending the new specific rule, i.e. in extrapolating to larger domains, because this extrapolation will *conflict* with the previous system of rules.
When they perceive a new example that can be solved more efficiently by activating a new rule compared with the standard rule, they have two possibilities: either they consider the new example as a exception to the previous general rule, or they try to re-define the building blocks of the rule's system, solving the conflicts among rules. This is of course a very heavy mental task, and therefore even though a single mutation can improve the system's performance, it is more likely that it is considered as exception and do not lead to a redefinition of abstract sub-problems.
Errors that are hidden within the sub-optimal representation can be uncovered and corrected mainly by rethinking the representation – that is, reformulating a new division into sub-problems – and changing it. Sub-optimal solutions are therefore stable under mutations because to change representation maintaining a limited number of general rules it takes radical modifications of the building blocks - the elementary sub-problems - of which a strategy is composed.[14]

4 Concluding remarks

Learning and discovering a strategy for solving a problem is a task that humans may tackle with an extraordinary speed, but necessarily producing sub-optimalities and biases, while an artificial algorithm based on the idea to discover an optimal solution by trials (mutations) should be successful but enormously slow. For example, a player wishing to tackle Rubik's cube and discover the optimal strategy through trial and errors will certainly achieve his goal, even though it may take him a number of years comparable with the age of the Earth. There are more than three billion configurations each of which allows 6 moves (3 vertical and 3 horizontal rotations). The player would not be locked-in in a sub-optimal strategy because he could improve by mutation the fitness of every sub-optimal configuration and the search would be successful, but in the long, long run.
Although counter-intuitive, this property holds true in the general setting of finite state machines, puzzles and automata problem solving. We have seen in fact that the problems that can be expressed by programs that are computationally equivalent to automata allow a set of optimal stable solutions, each of which may be reached through mutations.
These properties hold, as we have seen, provided we define properly a measure of efficiency - or fitness - of the strategy. The fitness for these problems has been defined in a very natural way in relation to the distance between a starting configuration and the final goal, following the principle that the lower the number of actions needed to achieve the goal, the better the fitness. When the fitness function is defined this way, the NK model predictions are not fulfilled. In fact, the degree of epistatic interdependence is positive because the effect of a mutation on the fitness is affected by the value (the "alleles") of some other elements of the system, but at the same time it is possible to modify the strategy via simple mutations, until the optimal configuration is achieved.
This strange property has a counterpart in terms of decomposition of problems in sub-problems. Here a "weak" decomposition principle works. In fact if we decompose a strategy in its elementary statements we realize that even though they are not independent, it is possible to optimize each statement separately and achieve an optimal solution (strategy). However this "possibility" is highly inefficient, because to achieve an optimal configuration by mutations a huge computation would be required.
Players involved in solving a puzzle should in fact remember a huge set of condition-actions, and could discover the optimal strategy by means of small incremental changes. Because of their limited capacity for calculation and memory, they proceed by constructing compact and simplified

representations of the strategies to adopt - representations based on the decomposition of the problem into sub-problems and abstractions at various levels. These representations introduce ‘errors’ into the choice of actions because the sub-problems identified are not fully independent, as the subjects believe. These errors are therefore concealed by the representation, and they can be uncovered and corrected only by dispensing with the representation - that is, the division into sub-problems - and radically changing it.

This gives stability to non optimal representations: in fact, to improve a representation it would be possible to modify single statements, so correcting the hidden errors but it could be incredibly costly, in terms of computation and mental effort, to enumerate and detect the wrong statements of the strategy. Any attempt in this direction would destroy the advantages of a simple and compact - although imperfect - representation.

Therefore it becomes essential to recodify the problem in such a way as to get closer to an optimal strategy without compromising the simplicity of the representation. As we have shown in the MiniRubik example, there is a trade-off between simplicity and efficiency of a decomposition. It follows that simple and compact representations may be locally stable sub-optimal solutions.

Let me conclude by suggesting some extensions of the previous analysis of individual problem solving, to the context of organizations, in full awareness that what follows, as all extensions and extrapolations, may embody hidden errors.

If we want to transfer these properties by analogy to organizations, we should imagine fully fragmented and distributed systems in which organizational learning may happen via random mutations, without any conscious design. Distributed organizations in which there is no conflict in reward allocation may therefore evolve until the optimal configuration is reached. It is difficult to map these types of abstract organizations onto some credible sketch of the real human organizations. But to be better illustrated, they could be compared to the organization of ants. The properties we have shown predict that an organization of insects – none of which is supposed to create complex mental models of the reality (Johnson-Laird, (1983)) - may very slowly adapt to an optimal division of tasks, by trial and error. The organization design emerges therefore spontaneously without a plan, as the result of a “social” computation. A huge, decentralized computation that can be carried out only by a very large society of unaware individuals.

It is reasonable to claim that, within human organizations, the discovery of a strategy is not based on these principles, or at least is not uniquely based on this mechanism. A more powerful mechanism to explain the organizational evolution is based on problem solving and problem representing. We have already discussed many properties related to these processes, and now it is convenient to add further elements to qualify the impact of the representations on organizational problem solving.

Human beings create simplified and incomplete representations of the problems facing them. Some representations may be locally stable for the reasons we have illustrated before. We could consider the locally stable representations in some sense as “ideologies” (Denzau and North (1994)) because they represent the reality in such a simplified way that it is valid only in certain context and cannot be extended to every condition. These simplified representations allow accelerating enormously the evolution of the organizational rules, but introduce strong discontinuities in the process. In fact in evolutionary conditions, in which an organization is learning a new division of knowledge and competences, the coordination rules should evolve together with the discovery of a new division of tasks. This process is a very complex one, which may require a huge coordination effort if it is displayed in a distributed way. As a consequence, the role of hierarchies becomes relevant as these are leaders of a top-down process of problem solving and planning. The difficulty to coordinate a distributed search for a new organizational form suggests that to get out from a lock-in, small local modifications of the team's strategies can be ineffective, while a dramatic recombination of action rules, planned by a centralized organization, may be successful.

The notion of *representation* may help to reconcile two apparently challenging "visions" of the evolution of the strategies and of the organizational change: on the one hand the idea of intelligent planning as conscious division of a problem in sub-problems, in which, despite the limits of individual rationality, a hierarchical organization may improve his efficiency.
On the other hand the idea of the organization's strategy as an evolving system that is transformed by small innovations, mutations, without a conscious plan shared by the individuals; this view considers institutions as the unplanned and unintentional result of the human actions, that, as Hayek claims, "can be understood as if it were made according to a single plan, although nobody has planned it".
We may try to go beyond this dichotomy, using the problem solving approach and the notion of representation. The nature of evolution in human artifacts, institutions and organizations, has different features compared to the evolutionary properties of biological systems: in human organizations evolution involves a process of collective learning that is driven by human conscious will, in which, during a rational, intelligent activity of planning, despite the effort of the individuals to be fully rational, nevertheless errors are unintentionally created.
Even though by mutations it is possible to introduce improvements into the organizations, and get closer to an optimal configuration, the evolution of an organization based *only* on mutations should require an enormous amount of time. The evolution of organizational structures is, on the contrary, relatively speed and discontinuous, because is based on the human ability to represent, design and revise their settings. Accordingly with the spirit of Cyert and March *Behavioral Theory of the Firm* [15], business organizations are characterized by complex hierarchies of rules that change with different speed and different applicability domains. Plans and strategies therefore are the intentional results of the human projectual activities but at the same time they conduct individuals and organizations to configurations that are largely different from the original design, because representations - both at the individual and teams level - systematically embody hidden errors.

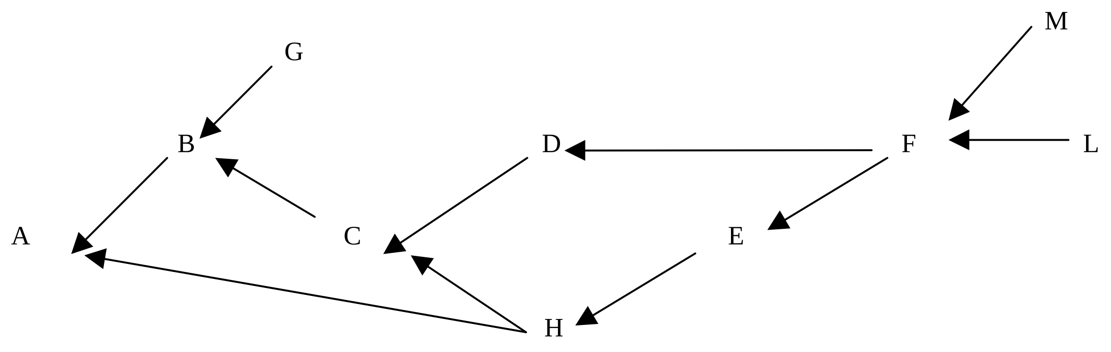
Appendix : Epistatic interactions

We have already noted that the link matrix of a puzzle can be represented by a graph, whose nodes represent the configurations of the system. (The node A is connected to node B by an oriented arc if and only if applying to A a legal rules of action we obtain B.). Just to give a simple example of the equivalence between Puzzles, condition-action systems and graphs, consider the table of the links below. Here, we have indicated in column 2 and 3 the states that can be reached by applying respectively actions X and Y to the conditions listed in the first column of the array.

Table 12

CONDITIONS	ACTIONS		
	X		Y
A	-		-
B	A		-
C	-		B
D	C		-
E	H		-
F	D		E
G	-		B
H	A		C
L	F		-
M	-		F

Table 13



From Table 12 we build up an equivalent graph, with the simple following convention: if in the matrix we have for example F→Y→E, i.e. if applying the rule Y to the configuration F we obtain the configuration E, then in the graph we draw a pair F, E connected by an edge oriented from E to F. The graph is called directed graph, because is composed by directed edges (arrows).
Table 13 shows the graph equivalent to the matrix of Table 12.
Whichever representation we choose for the game, a condition-action system or a graph, a strategy can be described as a selection: in the condition-action system, it is a selection of one among the possible actions corresponding to every condition. In the graph representation, the selection consists of the choosing of one and only one link between a node (condition) and the adjacent ones (successors). [1]

Now let us try to modify a strategy by introducing mutations. Given a strategy, by introducing a mutation, i.e. changing one action (from A to A') in the condition-action list, we change also one link: suppose that the game is given by the graph in Table 13 above. Let the strategy be defined by the list of Table 15, or equivalently by the spanning tree represented in Table 14, that differs from the graph of the game for the removal of the arches FD and HA .
We can now evaluate how the fitness changes when we apply to a given configurations the different available actions (the mutations of a gene): if, starting from a given strategy (tree), we substitute for example F→Y→E with F→X→D, this corresponds to remove the link F→E and attach the node F as parent of D.
It is obvious that this mutation is convenient if and only if D is nearer than E to the goal, because our fitness measures the distances between the nodes and the goal. In our example (table 14), D is nearest than E to the goal; then removing the link F→E and introducing the new link F→D, also all the nodes of the tree rooted in F (M and L in the example) become closer to the goal. It follows that this mutation increases the fitness value.

Table 14

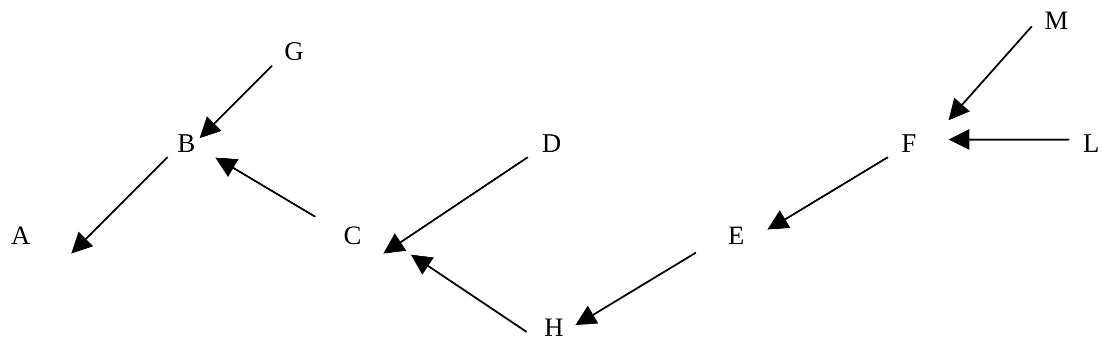
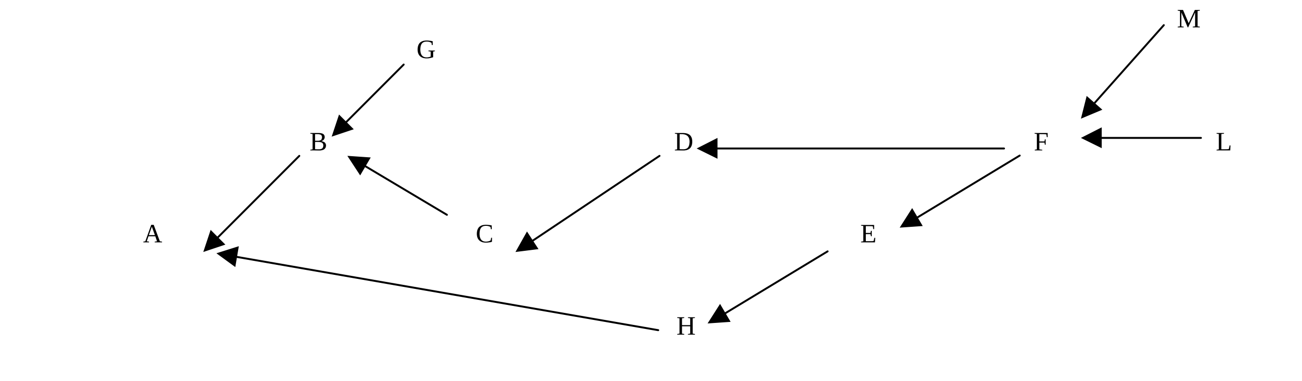


Table 15

CONDITIONS	ACTIONS		
	X		Y
A	-		-
B	A		-
C	-		B
D	C		-
E	H		-
F	-		E
G	-		B
H	-		C
L	F		-
M	-		F

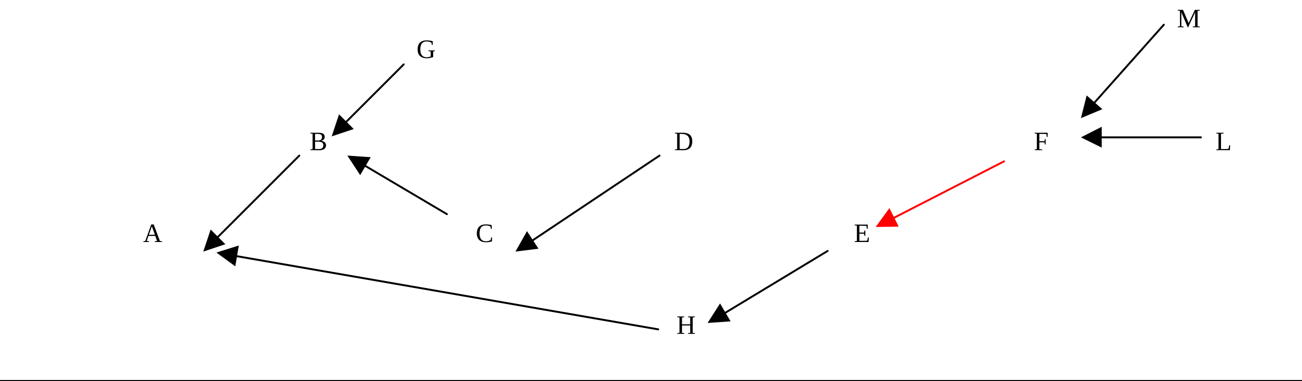
Nevertheless the effects of a mutation depend upon the "context": in fact by disconnecting the node F from E and linking it to D, the fitness function has been increased because the node H was connected to C. But if next we introduce a new mutation, by removing the link H→C and linking H to the root A, as a consequence the fitness associated with the node F changes. Now (Table 16) F is nearer to the goal if disconnected from D and reconnected again to E: therefore when we compare the value of the fitness function in relation to the different available links between F and his successors, we have that this value depends upon the length of the paths directed from the successors to the goal: therefore if we modify these paths, by changing some of the links of the nodes involved, also the values of the fitness function change.

Table 16



In other words, the variation of the fitness that follows a change in a link between the node N and his successors, depends on how are linked *other nodes*: this means *by definition* that there is epistatic interaction, i.e. K is positive.
The final tree of minimal length (optimal fitness) is drawn on Table 17.

Table 17



Finally note that if we get a node N closer to the root, either this new connection does not have any effect on the positions of the other nodes (when N is a terminal one) or we have an effect on the same direction: we get closer to the root all the nodes rooted in N. The effect of a mutation can be therefore of different strength, in relation to the length of the sub-tree rooted in N (the greater is the sub-tree, the higher is the improvement of the fitness).

NOTES

1 I am grateful to A. Caranti , F. Giunchiglia, P. Garrouste, M. Ricottilli and M. Warglien, for useful discussions. The usual disclaimers hold. During the revision of this paper, Giunchiglia suggested to me that some of the ideas here exposed were closely related with "Intelligent Planning", a classic area of Artificial Intelligence. A key reference on this regard is Q.Yang (1997)

[2] The two experiments with *Target The Two* , in which individual biases and team traps are compared, are briefly outlined in Bonini N., Egidi M. (1999).

[3]See Yang Q. (1997)

[4] Of course, a condition-action system as a strategy for solving a puzzle is less powerful than a Post system. While the latter is equivalent to a Turing machine, the former is equivalent to an automaton. However, an automaton is perfectly suited to representing the strategy used to solve a puzzle.

[5] If there are covered position in the game, or more generally not all the conditions of a state are observable, as happens in TTT, Players have to consider a non-fully observable state as a class of states, and construct their strategies by considering all possible states with which the non observable state is compatible. The strategy construction becomes then computationally explosive, even though all considerations we are making on strategies and their properties still hold.

[6] In this simple form programs are equivalent to automata, not to Turing machines. They are therefore ‘weak’ in expressive capacity but nevertheless adequate for description of a strategy.

[7] It is in some sense a minimal representation of a set .For example, if the atomic components of the strings are binary units, 0-1 The set X={ {000000}, {000001}, {000010}, {000011}, {000100}, {000101}, {000110}, {000111}} can be described by a unique string: X={000###}, while the set X*={{000000}, {000001}, {000010}, {000011}, {000100}}, while is composed by a lower number of strings, can be described by a pair of strings. {0000##} U {000100}.

[8] The "representation" of a problem is therefore defined by a decomposition pattern based on a codification of the problem's configuration (See also Marengo (1999)).

[9] Holland (1988) re-defines the extrapolation process as Q-Morphism.

[10] A very interesting question is if it is possible to know a priori this length: in case of Turing machines this is a well known unsolvable problem. In our case, even though machines are automata, the existence of cycles makes impossible to provide an universal algorithm to verify if the machine stops.

[11] This paragraph contains a few expressions that require some elementary notions in graph theory and semigroup theory. The notions needed to conduct the proofs are briefly defined to allow the reader to follow the reasoning. Anyway the reader may skip the paragraph and go directly to the next one where the results obtained are discussed.
Let synthesize a small lexicon regarding graphs: A graph G(V,E) consists of a finite set V of <u>vertices (or nodes</u>) , and a set E of <u>edges</u> (or <u>arcs</u>) , joining pairs of distinct nodes. An arc may connect an unordered pair of nodes (A, B)=(B,A) or may connect an ordered pairs of nodes (A,B); in the latter case the is called <u>directed arc</u>. We will write A→B. If all edges are directed, the graph is called <u>directed graph</u>. We say that a vertex A is <u>adjacent</u> to vertex B if there is an arc from A to B. A <u>path</u> is a sequence of adjacent nodes. A <u>tree</u> is a graph with a vertex, called root, such that there is a unique path from the root to any other vertex of the tree. A tree with directed edges is commonly called <u>rooted tree</u>. Suppose to have a tree rooted in the vertex A, and all arcs are directed to the root. Choose two adjacent vertices C,D; if C→D then we call C parent (predecessor) of D and D child (successor) of C, and vice versa if D→C; vertices with no parents are called <u>leaves</u> of the tree or <u>terminal nodes</u>. In the present paper we use also rooted trees in which the orientation is inverted: all the paths starting from the root are directed to the terminal nodes.

[12] <u>A cycle</u> is a closed path, i.e. a path in which the last node coincides with the first. A graph is said to be cyclically connected if any two of its nodes are contained in a cycle.

[13] The average number of traits that contribute to the fitness, is related to the number of nodes that are linked to the goal by non-minimal paths.

[14] Interestingly, the fewer are the errors hidden into a representation, the smaller is the marginal advantage to restructure the representation. This property is similar to the quasi stability of dynamic biological systems. (Kauffman, 1989, pp. 593-597)

[15] (1992, pp.117-120)

[16] Or fixed points, that obviously are cycles of length 1.

References

Anderson, J.R. (1983). The architecture of cognition, Cambridge, MA: Harvard University Press.

Arthur W. B. (1988) 'Self reinforcing mechanisms in Economics', in Anderson W.P. and Arrow K.J. (editors) The economy as an evolving Complex System Redwood City, CA: Addison Wesley, pp. 9-31.

Berge C. (1985) Graphs, North Holland, Amsterdam.

Bonini N., Egidi M. (1999) 'Cognitive Traps in Individual and Organizational Behavior: some Empirical Evidence' Revue d'Economie Industrielle n. 88,2 pp.153-185

Cannon-Bowers J. A., Salas E. and Converse S. (1993). 'Shared mental models in expert team decision making'. In M. J. Castellan, Jr. (ed.), Individual and group decision making, Hillsdale: Erlbaum.

Chen W.K. (1971) Applied Graph Theory, North Holland, Amsterdam.

Clifford A.H., Preston G. B. (1961) The Algebraic Theory of Semigroups , vol. I , Providence, Rhode Island: American Mathematical Society.

Cohen M. D. (1991) 'Individual learning and organizational routine: Emerging connections', Organization Science, 2 (1) pp. 135-139.

Cohen M. D. and Bacdayan P. (1994) 'Organizational Routines Are Stored as Procedural Memory: Evidence Form a Laboratory Study', Organization Science, Vol.5, N.4, pp. 554-568.

Cohen M. D. Burkhart R., Dosi G. Egidi M., Marengo L., Warglien M., Winter S. (1996) 'Routines and Other Recurring Action Patterns of Organizations: Contemporary Research Issues' in Industrial and Corporate Change; 5(3), pp. 653-98.

Cyert R. M. and March (1963) A Behavioral Theory of the Firm, Prentice Hall, N.J., reprinted (1992) by Blackwell: Cambridge, Mass.

Cyert R. M. (1988) The Economic Theory of Organization and The Firm, Harvester Whestseaf, N.Y.

Denzau A.T. and North D. C. (1994) 'Shared Mental Models: Ideologies and Institutions', Kyklos; 47(1), pp. 3-31.

Egidi, M (1996) 'Routines, Hierarchies of Problems, Procedural Behavior: Some Evidence from Experiments' in: Arrow, K. J., et al., (eds.) The rational foundations of economic behavior. IEA Conference Volume, no. 114. New York: St. Martin's Press; London: Macmillan Press, pp. 303-33.

Egidi, M. Narduzzo, A. (1997) 'The Emergence of Path Dependent Behaviors in Cooperative Contexts' in: International Journal of Industrial Organization; 15(6), October 1997, pp. 677-709.

Fischhoff B. (1975). 'Hindsight vs. Foresight: The effect of Outcome Knowledge on Judgement under Uncertainty'. Journal of Experimental Psychology: Human Perception and Performance, Vol. 1, 3, 288-299.

Holland, J. H. (1975) Adaptation in natural and artificial systems, Ann Arbor: University of Michigan Press.

Holland, J. H., Holyoak, KJ., Nisbett, R.E., Thagard P.R., (1988) Induction - Processes of Inference, Learning, and Discovery , Cambridge (Mass) : MIT Press.

Johnson-Laird, P.N. (1983) Mental Models, Harvard University Press.

Kahneman, D., Tversky, A. (1986) 'Rational choice and the Framing of Decisions', in Hogart R. M. , Reder M. W. Rational choice - The Contrast between Economics and Psychology , Chicago: The University of Chicago Press.

Kauffman, S.A. (1993) The origins of order: self-organization and selection in evolution, Oxford: Oxford University Press.

Kauffman, S. A. (1989) 'Adaptation on Rugged Fitness Landscapes' in Lectures in the Sciences of Complexity, edited by Stein D. L. Santa Fe Institute Studies in the Sciences of Complexity, Vol. I, pp.527-712. Redwood City, CA: Addison Wesley.

Kauffman, S. A., Johnsen, S. (1992) 'Co-evolution to the Edge of Chaos: Coupled Fitness Landscapes, Poised States, and Co-Evolutionary Avalanches', in Langton C.G., Taylor L., Farmer J. D. , Rasmussen S. Artificial Life II, Redwood City, CA: Addison Wesley.

Laird, J.E., Newell, A., Rosembloom, P.S. (1987) 'Soar: An architecture for general intelligence', Artificial Intelligence 33, pp. 1-64.

Levinthal, D., (1997), 'Adaptation on Rugged Landscapes', Management Science Vol. 43, No7, July.

Levinthal, D. A. e March, J. G. (1993). 'The Myopia of Learning'. Strategic Management Journal, Vol. 14, 95-112.

Levitt, B. e March, J. G. (1988). 'Organizational learning', Annual Review of Sociology, 14, 319-340.

Loewenstein, G. e Elster, J. (eds.), (1992). Choice over time. New York: Russel Sage Foundation.

Loewenstein, G. e Thaler, R. H. (1989). 'Anomalies: intertemporal choice', Journal of Economic Perspectives, 3, 181-193.

Luchins, A.S (1942) 'Mechanization in Problem-Solving' , Psychological Monograph, 54, pp. 1-95.

Luchins, A.S., Luchins, E.H (1950) 'New experimental Attempts in Preventing Mechanization in Problem-Solving', The Journal of General Psychology, 42, pp. 279-291.

March J. G. (1988), Decisions and Organisations, Oxford, Basil Blackwell.

March, J. G. (1990) 'Exploration and exploitation in Organizational Learning, Organization Science, 2, pp. 71-87.

March, J.G. (1981) Footnotes to organizational change, Administrative Science Quarterly, 26, pp. 563-577.

March, J. G., Simon H. A. (1958) Organizations, (1993 2nd edition) New York: John Wiley.

Marengo L. (1999) “Decentralisation and market mechanisms in collective problem solving” Working Paper, University of Trento.

Frenken K., Marengo L. and Valente M. (1998) ”Interdependencies, nearly-decomposability and adaptation” , CEEL Working Papers, University of Trento.

Minsky, M. (1967) Computation. Finite and Infinite Machines, Englewood Cliffs,N.J: Prentice-Hall.

Newell A., Simon H.A. (1972) Human Problem Solving, Englewood Cliffs, N.J. : Prentice Hall.

Nilsson, N. J. (1971) Problem Solving Methods in Artificial Intelligence, New York: McGraw Hill.

North, D. C.(1991) 'Towards a Theory of Institutional Change' Quarterly Review of Economics and Business; 31(4), pp. 3-11.

Page S.E.(1996)” Two measures of difficulty”, Economic Theory, vol. 8 , pp.321-346.

Simon, H. A. and Ando A. (1961) Aggregation of variables in Dynamic Systems Econometrica, 29, pp.111-138

Simon, H. A. (1971) 'Theories of Bounded Rationality', in McGuire B. and Radner R. (editors.), Decision and Organisation , Amsterdam: North-Holland

Tucker A. (1984) Applied Combinatorics, New York :John Wiley & Sons

Yang Q. (1997) Intelligent Planning - A decomposition and Abstraction Based Approach Berlin: Springer.