Decomposition patterns in problem solving

Massimo Egidi
massimo.egidi@unitn.it

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

ABSTRACT

The paper develops a theory of biases in decision making. Discovering a strategy for solving a game is a complex problem that may be solved by decomposition; a player decomposing a problem into many simple sub-problems may easily identify the optimal solution to each sub-problem: however it is shown that even though all partial solutions are optimal, the solution to the global problem may be largely sub-optimal. The conditions under which a decomposition process gives rise to a sub-optimal solution are explored, and it is shown that the sub-optimalities ultimately originate from the process of categorization that governs the creation of a decomposition pattern. Decisions based on a strategy discovered by decomposition are therefore frequently biased . The persistence of biased behaviours, observed in many experiments, is explained by showing the stability of different and non optimal representations of the same problem. An application to a simplified version of Rubik cube is finally developed.

1. Introduction

One of the most widely known heuristics to solve problems suggest to decompose it into parts (sub-problems) which can be solved separately. The decomposition process is repeatedly applied to each sub-problem until elementary sub-problems, easy to be solved, are identified. Applications to games and puzzles are very usual: Hanoi tower, Rubik cube, chess, are typical contexts in which decomposition is usefully applied and leads more easily to discover a solution strategy. During decomposition players identify progressively simpler sub-games, until they reach elementary sub-games that are no more decomposable. Given the simplicity of these elementary sub-games, players are supposed to be able to discover the optimal strategy for each of sub-game. This is apparently the key to reduce the complexity of research and obtain optimal global solutions.

However, as we will demonstrate in this work, when a problem is decomposed into parts, the pattern of decomposition that players – consciously or not – adopt, introduces hidden sub-optimalities. Hence, although the solution of each sub-problem is optimal, generally the global solution is largely suboptimal.

Decisions based on a suboptimal strategy discovered by decomposition are therefore biased . As we will see, biases and errors are ultimately originated by the process of categorization that governs the creation of a decomposition pattern.
Let me give a rough idea of the reasons and the effects of the imperfections in decomposition patterns, focusing on puzzles that provide a context in which the imperfections emerge clearly. In puzzles the optimal solution is given by the shortest path from the starting configuration(s) to the goal; given the enormous number of game configuration that should be analyzed to get an optimal solution, to obtain a simple representation of the solution players classify the states of the puzzle in a relatively low number of categories: large sets of game configurations are therefore aggregated in categories, and this categorization implicitly lead to a decomposition pattern.
Categories are the result of a process of abstraction and classification based on the salience of symbolic features of the configurations of the game. In Rubik for example, the disposition of colours of the tiles along one, two... corners are salient elements to categorize classes of configurations that are supposed to be progressively nearer to the final configuration. The same categories therefore allow identifying sub-goals with which the original puzzle is decomposed.
Players consider as equivalent large classes of states because of some similarities among them and try to *order* them in terms of distance to the target, i.e. to establish a *metric* among the categories. But frequently the order that players conjecture does not hold for all elements of the categories: for example in Rubik cube players may believe that a cube with three faces with the same colour is nearest to the goal than a cube with two faces with the same colour, which is not necessarily true for all elements of these two categories. Consequently the procedure he will adopt cannot be optimal for all configurations. Thus players achieve relatively simple and abstract representations of the strategy by categorization, but the evaluation of the order among the categories may be inaccurate; because of this distortion, the players do not always get close to the goal through the shortest path, but on the contrary at least for some configurations they achieve the goal following a tortuous path that in some steps gets farer from the goal.
The most intriguing aspect of this situation is that, while the paths to the goal generated by a unique procedure are optimal for some of the elements (the "right" ones) but not for all of them, when players solve the puzzle for the "right" elements they will be confirmed in the optimality of their decisions.
Therefore they cannot easily perceive that they have made a wrong classification and modify their decomposition pattern.
Hence biases in decision making are originated by the inconsistency between the categorization on which the solution of the game is based and the metric of the original problem. A player decomposing a problem into abstract and easily controllable sub-problems may easily identify the optimal solution to each sub-problem: however it frequently happens that, even though all partial solutions are optimal, the solution to the global problem is sub-optimal. I will call "metrically invariant" a decomposition pattern if by optimizing separately the sub-problems in which the problem is decomposed, we get the optimal solution.
It is intuitive to understand that there are different categorizations and decomposition patterns to each problem, some which keep the metrical features of the original problem, whilst the vast majority does not respect that feature and generates decision biases; this work will illustrate the formal conditions under which a decomposition pattern applied to a puzzle maintains (or not) its original metric and will explain the reasons why distortions of metric give rise to sub-optimal strategies.

The experimental data available to date in puzzle solving show that most individuals, once a strategy has been identified, remain stably anchored to it even though it is not optimal.
This tendency has been proved by Luchins (1942) and Luchins and Luchins (1950) who conducted experiments with subjects exposed to problems that had solutions and displayed different levels of

efficiency. The authors show that subjects, having identified the optimal solution of a task in a given context, may "automatically" and systematically use such solution applying it also to contexts where it proves to be sub-optimal. This process is called "mechanization of thought".
The experiments with *Target the Two* (Cohen and Bacdayan 1994, Egidi and Narduzzo 1997) confirm that a similar process is also implemented in the behaviour of a team, in an even more evident and persistent manner: groups of subjects who are to jointly solve a shared problem may remain even more stably "trapped" into sub-optimal solutions than single individuals. In fact, while difficulties encountered by a single subject when solving a problem in a new way depend on the possibility to discover a new solution to a problem and are influenced by cognitive limitations to individual learning, this is even more difficult in a group because there is a need to identify new ways to cooperate in problem solving to discover and adopt an alternative solution jointly.

The sub-optimality which is inbuilt in many decomposition patterns will enable us to later explain the phenomenon of the *mechanization of thinking* and some classes of error representation in problems, which have been observed in experiments conducted during strategy games. The explanation of the lock-in process into systematically biased decisions is based on the features of the non invariant decomposition patterns: when subjects discover a decomposition pattern, they do it in the context of a specific run or repeated sequences of runs. This means that they build up their categorisation and the consequent decomposition pattern upon a limited domain of game configurations. In the specific context where a pattern is discovered, the decomposition, though not invariant, may generate a locally optimal solution, and the subject may not be able to perceive that the validity of his discovery is limited.
Finally, it will be suggested that the imperfections in a decomposition pattern are related to the fact that the description of a puzzle and of its rules gives naturally rise to a system of categories ; for example in Rubik cube, the disposition of the colours along a corner, an edge or a face may be the basis for natural categories. Frequently this natural categorization is not invariant and therefore to discover an invariant decomposition pattern players have to re-codify the game and modify the categories and their level of abstraction. But re-codification requires to abandon the simplicity of the natural classification. Therefore, as we will see later, there is a *trade-off* between the simplicity of representation of a strategy and its *efficiency*.
After studying the features of decompositions and related categorizations, we will later apply them to a game which is a simplified version of Rubik cube, while comparing different decomposition patterns, both invariant and non invariant; we will then analyse some experimental data, showing that a large part of deviations from rational behaviour are the result of the adoption of weakly sub-optimal strategies, due to non invariant decompositions patterns.

2 Setting definitions and basic questions in puzzles solving

In puzzles, a *specific problem* is defined by a pair of states, i.e. starting and final state of the game, and a solution consists of a strategy which, starting from the initial state, allows the player to reach the final state. The strategy can be represented by a list of conditions-actions, where each state of the game (condition) is associated with an action (the adequate move); The path from the initial state to the goal is generated as follow: while comparing the starting state of the problem with the condition-action list, the move to be applied is identified and a second state is obtained (successor); by repeating this process, the final state is reached through a "path" composed of a sequence of connected states.

I refer to a *specific* problem because its reference elements - the starting state and the target state - are single states. In general however, a problem is defined with reference to classes of states, the class of the starting states P and the class of goal states G. A solution can be therefore defined as a strategy which - starting from <u>each</u> state pεP, allows the player to reach <u>a</u> final state g ε G.

To formally represent a *puzzle*, the following shall be defined

1 A set of *states* $X=\{x^1, x^2, x^3, \ldots. x^n\}$ of the game and two subsets, P and G which are the set of starting and goal (final) states respectively. If a problem is a specific one, P and G are composed of a single element each.

2 A set of *moves* $M=\{m_1, m_2, \ldots m_m\}$

3 A set of *rules* indicating to which states a move is applied and the configuration of the puzzle after that move is applied. This set of rules can be written in the form of a matrix of state transition, A, where its element a_{ij} it represents the move m_k that must be applied to the state x^i to reach the state x^j. This state is called successor of state x^i and vice versa x^i is called predecessor of state x^j; $a_{ij}=0$ means that there is no move connecting state i with state j . (1) The transition array A can be used to represent the puzzle as a (directed) graph where nodes represent the states of the game whereas the moves are represented as directed arches connecting two nodes. The graph is supposed to be strongly connected, i.e. it is assumed that all goal states are reachable by any starting state. A is related to the adjacency matrix of a graph $C=\{ c_{ij} \}$ is defined as follows: $c_{ij}=1$ if arc (x_i,x_j) exists in graph Γ ;$c_{ij}=0$ if arc (x_i,x_j) does not exist in Γ . Thus the adjacency matrix C can be obtained by transition matrix A, by simply setting $c_{ij}=1$ if $a_{ij}=m_k$ and $c_{ij}=a_{ij}=0$ otherwise. (1)

<u>2.1 Decomposition patterns and mental models</u>

As stated in the literature (Nillson 1986), a problem can be decomposed into a tree of *and/or* sub-problems. In the first instance, to solve a problem one must solve <u>all</u> sub-problems the problem is composed of, while in the second case one should solve just one of the problems it is made up of. The identification of sub-problems is achieved through processes of *abstraction* (each <u>*and*</u> structure is an abstraction based on salient elements) and *specification* (given a problem, any <u>*or*</u> subset is a specification of its special conditions).
Decomposition is a recursive process where sub-problems, sub-problems of sub-problems, and so on…are identified until problems having a "minimal" size, i.e. one which can be solved in just one move, are finally achieved. *Terminal sub-problems*, i.e. those which can be solved in just one move, are the finest "grid" where a problem can be defined and solved : a terminal sub-problem is composed of two subsets, Ψ_h and R_h (starting and final states) and solved in just one move m_w: $\Psi_h \rightarrow m_w \rightarrow R_h$.
The list of all terminal problems (each provided with the relevant move) can be considered as a *program,* [1] in the form of condition-action equations $\Psi_h \rightarrow m_w$, which solves the problem. Ψ_h is the set of starting states of the k^{th} sub-problem, which will be called *k^{th} building block of the decomposition pattern.*
The instruction $\Psi_h \rightarrow m_w$, means that *the same move* m_w can be applied to all configurations belonging to the same building block Ψ_h . In "natural" human problem decompositions, the sub-problems are

[1] Each strategy can be considered as a program as it is made of a condition-action sequence, taken from a transition matrix. As such a programme is equivalent to an automaton and not to a Turing machine.

normally identified by abstractions and categorizations; consequently the sets of starting and target states in a terminal sub-problem are normally categories, and is natural to search a move that applies to every element of the first category (starting states) to reach the second (goal states).
More generally the solution of a terminal sub-problem $\{\Psi_h , R_h\}$ may consist in a different move for every pair $\{x, x'\}$, $(x \in \Psi_h \quad x' \in R_h)$, and therefore is composed by A_h, a sub-matrix of A that connects the state $x \in \Psi_h$ with the states $x' \in R_h$.
A *full decomposition pattern* of a given problem is obtained when the decomposition into sub-problems is developed until the terminal problems $\{\Psi_h R_h\}$ are achieved.

When decomposing a problem according to *and/or* trees, the building blocks can be usually described through abstractions made on symbols describing the configurations of the game. For example, in Rubik cube, each configuration is described by the position of the coloured tiles on the six faces of the cube, and if we want to operate on the configuration with a corner, we may generally think that every configuration is described by a combination of k symbols $\alpha_1, \alpha_2, \alpha_3, \ldots \alpha_k$. A configuration is described by a string of n such symbols , in the form of $x^i = [\alpha_r, \alpha_s, \alpha_w, \ldots \alpha_x]$. A string where the symbol # (don't care) appears in position h is used to indicate a category obtained by abstraction on the h^{th} element.
The transition matrix describes the whole game in *extended form*. It must be noted that generally a game is *not* described in extended form: it is generally expressed in a much more compact form where moves are applied to very broad categories of states. In the Rubik cube, for example, the legal moves (vertical and horizontal rotations) are applied to the cube regardless of the configuration of the game. The same happens for final configurations. For example, the definition of "winning configuration" in chess implies a large (and unknown!) number of different configurations, which all share the same property: the king must be under checkmate.
When determining the strategy of a game, individuals try to identify rules which are valid not for a single state but for broad classes of game configurations. The compact representation of a strategy, implying the grouping of many configurations into classes, is "naturally" performed by the human mind which simplifies and focuses its attention on special configurations, while abstracting some properties. The strategy is thus described as a more limited list of instructions which can be more easily handled than the extended representation.
When terminal problems and building blocks are identified by decomposition, they are generally identified in the form of *categories*, i.e. sets of configurations that share some common, salient, feature and therefore can be represented without taking into consideration the non-salient features. In this way for every different decomposition pattern vast sets of configurations, to which the same move is applied, are grouped and described as a single strings. The list of a strategy described extensively, i.e. indicating the action to select for each configuration, may then be represented synthetically, provided building blocks can be represented into categories.
The strategy (or program) can be described in a compact form as set of building blocks, to each of which and an action is connected. A complete programme can therefore be composed of a relatively short list of instructions defined by building blocks of the following kind:.

$C_1 = \# \alpha_r, \alpha_s, \#, \ldots \# \alpha_x \rightarrow m_w$
$C_2 = \# \# \alpha_r, \alpha_s, , \ldots \alpha_x \rightarrow m_s$
$C_3 = \alpha_r, \alpha_s, \#, \ldots \alpha_x \rightarrow m_t$
$C_4 = \alpha_r, \alpha_s, \#, \ldots \alpha_x \# \# \rightarrow m_u$
. . . . . .

$C_n = \alpha_r, \alpha_s, \#, \ldots \alpha_x \ \# \ \rightarrow m_z$

every row of the program is composed by a category C_s that represent a set of configurations and by a move m_u. The category is an aggregation of configurations. This aggregation can also be considered a *classifier*, because every category allows the player to classify a set of different elements on the basis of some common and salient properties. We can therefore consider C_k as classifiers or equivalently as mental categories, created by abstractions leaded by salience. Following Holland (1988) we can consider a *mental model* as the system of classifiers $C_1..C_n$. For the strategy to be complete, each configuration should belong (at least) to one of the C_i components i.e. that the property $C_1 \cup C_2 \cup C_3 \ldots\ldots \cup C_n \cup G = X$ is valid, where X is the set of all possible configurations and G is the set of goal-configurations. The components may be independent or not. In the first case $C_h \cap C_k = \emptyset$ for every $h \neq k$. In the second case for some $h \neq k$, $C_h \cap C_k \neq \emptyset$, and therefore some state x^s, belongs to both components: this implies a *conflict* in the decision to be made. In Holland (1986, cap. 2) the conflict is solved by attributing a higher weight to the most specific component.

3 Theorems and general properties

Strange properties of distances

The properties of a decomposition pattern are strictly related to the features of the shortest paths between starting states and goals of every sub-problem. Therefore we start the discussion by stating some properties of shortest paths.

The *distance* between two nodes in a graph can be defined as the number of arches within the shortest path, connecting the two nodes, i.e. the length of the shortest path connecting the two nodes. In puzzles and games, the goal might be composed of a set of nodes, and the same may happen in configurations characterising the starting states; it is therefore worth defining the meaning of "shortest path" or "distance" between sets of nodes.

Suppose that G is the set of target nodes and s is a node in the graph (not belonging to G). Consider the set M of shortest paths between s and all nodes of G. We define as "the shortest path between s and G" the shortest element of M. The distance between s and G is the length of the shortest path. This means that the distance between a node and a set G is defined as the shortest distance between that node and the nodes in G. Put in different terms, suppose that, among all elements of G, $g_k \in G$ is the "closest" element to s; the shortest path between s and G is by our definition the shortest path between g_k and s .

This definition may easily be generalized to sets of nodes: the shortest path between two sets A and G will be the shortest path between the two closest elements of A and G.

Distances exhibit some "strange" properties, and noticeably the non-reversibility; in fact the following property holds:

For directed graphs : While the transition matrix of a puzzle is not necessarily symmetric, the shortest path from node x^h to node x^k not necessarily coincides with the "inverse" shortest path from x^k to x^h. The pair (x^h, x^k) may have a different distance than the pair (x^k, x^h). Therefore we have

$$D(x^h, x^k) \neq D(x^k, x^h)$$

For nondirected graphs: for every pair of nodes, (x^h, x^k) we get

$$D(x^h, x^k) = D(x^k, x^h).$$

In fact, the distance between node x^h and node x^k is given by the length of the shortest path connecting node x^h to node x^k ; let (x^h , x^s, x^t …,x^v , x^k) be the nodes along the shortest path. These nodes are directly connected, and therefore the coefficients of the adjacency matrix are non zero : $a_{hs}=a_{st}…= a_{vk}$ =1. While the graph is nondirected, the adjacency matrix is symmetrical, i.e. $a_{ij} =a_{ji}$ ∀ i,j. It follows that the inverse shortest path from x^k to x^h exists and has the same length of the shortest path from x^h to x^k, because for the symmetry of the adjacency matrix the coefficients $a_{kv},…a_{ts}$ a_{sh} are nonzero.

3.1 Backward branching algorithm in the search for shortest paths

Suppose that a hyper-rational player exists, provided with unlimited computational and memory capacity. Assume for a moment that the goal of the puzzle is composed by only one state. To discover the shortest path to the final goal in a puzzle the hyper-rational player may implement a "backward branching" strategy as follows: starting from the final configuration of the puzzle, the player labels all "previous" configurations, i.e. those leading to the final goal in a move, and then those leading to the final goal in two moves, and so on, thereby assigning to each configuration the *distance* to the goal. The states of the game are thus classified and ordered in relation to the distance to the final goal and an *order* of configurations in established in relation to the distance to the objective.
The same algorithm can be improved if the goal is a set of states.
Call G the set of target nodes. S_1 is built as the set of nodes adjacent and directed at least to one of the nodes in G, which will be labelled at a distance 1.[4] Let us then build S_2 , the set of nodes directed at least to one of the nodes of S_1, excluding nodes belonging to G; they are labelled at a distance 2; similarly, S_3 is built, being the set of nodes directed to at least one of the nodes of S_2 , excluding those belonging to $S_1 \cup G$; in this way, after N iterations (N≤ number or nodes in Γ), all elements of the graph will have been reached and labelled. The sets S_1 , S_2 , S_3 … S_N that have been identified with this procedure will be called "layers" at distance of 1,2,3… steps to G. All nodes in layer k are at k distance to the final goal G.

We will show that *any decomposition pattern whose building blocks are layers, is invariant.* To get a clear description of this property, it is convenient to re-describe the *backward branching* procedure in terms fit to the matrix representation of the graphs.

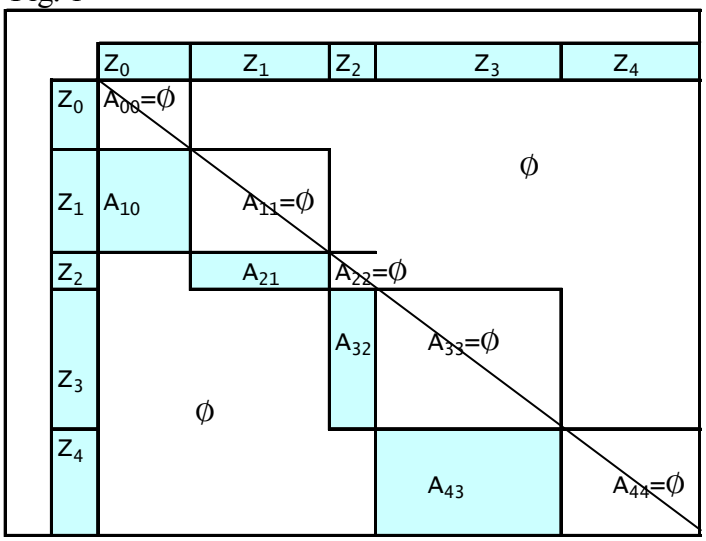
We will come up to the matrix representation by using an iterative method: first we show how to build up a matrix representation of a layer S_{k+1} if S_k still exists , for every k; finally we apply the method to S_0=G.
Assume that the by applying iteratively the *backward branching* procedure we have built up the layers S_1 , S_2 , S_3, …. S_k. Our goal is to build up S_{k+1}. Remember that S_1 is the set of the predecessors of G, S_2 the set of predecessors of S_1, and so on.
Call Z_h the set of indexes of the nodes (configurations) that belong to the layer S_h and call z_h the *number* of elements in Z_h. For continuity of the representation, call Z_0 the set of indexes of the nodes in G. For example if S_k={ x_r, x_s , x_t, x_u }then Z_k={r,s,t,u} and z_k=4.
The following properties hold: ∀ h, k (h≠k) $Z_h \cap Z_k = \emptyset$, (because two nodes belonging to different layers cannot have the same distance to the goal) and $Z_0 \cup Z_1 \cup Z_2 \cup Z_3 \cup …. Z_s$={1,2,3,…n}, where s is the number of layers.

To identify the predecessors of a given node $x_h \in S_k$ we have first to find out the columns of A whose indexes belong to Z_k. Suppose that j∈ Z_k . Check the coefficients a_{ij} along the column j, for i=1,..n. If

$a_{ij} \neq 0$ than the link between x_i and x_j exists. This means that the node x_i is adjacent (and directed) to node x_j and therefore is a predecessor of x_j. Therefore by repeating the same procedure for all elements j in Z_k , we collect all indexes i for which $a_{ij} \neq 0$, $j \in Z_k$. It is easy to recognise that the set of all such indexes i is Z_{k+1} , because Z_{k+1} is by definition the set of x_i that are predecessors of $x_j \in Z_k$.
This procedure can be applied first to the elements of the goal G, to obtain the first layer S_1 and its indexes Z_1 and then iteratively to all other layers.
Finally, call $A_{k+1,k}$ the submatrix of A composed by the coefficient a_{ij} with $i \in Z_{k+1}$ and $j \in Z_k$. $A_{k+1,k}$ is composed by the coefficients a_{ij} that link the layer S_k and its predecessor S_{k+1}.

So far we have identified both the sequence of layers Z_0 , Z_1 , Z_2 , $\cup \ldots . Z_s$ and the sequence of sub-matrices A_{10}, A_{21},…As,s-1 that link the adjacent layers. Using this procedure we can now build up the *matrix of optimal links*, B. This matrix is composed by the coefficients of A that represent links among layers, and therefore that give rise to the shortest paths from every initial state x to the goal G. The matrix is represented in fig.1

Fig. 1

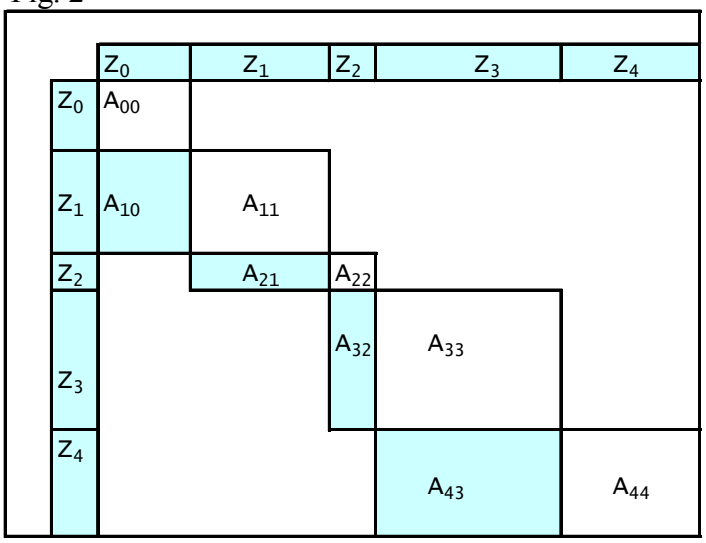


Rewriting the transition matrix

It is interesting to compare the original transition matrix A with the matrix B of shortest paths.
Let us restart the process of discovery and identification of the layers; at the first step, permute the columns of A in such a way to put the in the first z_0 positions the columns of indexes $j \in Z_0$ of matrix A. Then by identifying - within the first z_0 columns - the indexes i for which the coefficients a_{ij} are nonzero, we get Z_1 and A_{10} . Permuting the rows of A to put in the first z_1 positions the rows of indexes $i \in Z_1$ we now have the sub-matrix A_{10} in the first z_0 rows and z_1 columns of A. By continuing this process we identify Z_2 and A_{21} , and by permutation of rows and columns we put A_{21} in the columns from z_0 to z_0+z_1 and in the rows from z_1 to z_1+z_2. By continuing this procedure, we get finally a new matrix which is perfectly equivalent to A: the original array A in fact has been simply modified by permutations of rows and columns.
Call P a permutation matrix composed in the following way: the first z_1 column of P are composed by the unity column-vectors e^i with $i \in Z_1$; the next z_2 columns are composed by the unity column-vectors

e^k with $k \in Z_2$, and so on. Our final permuted matrix is given by $A^* = P^{-1}AP$. The permuted matrix A* is reported in fig. 2.

Fig. 2



Note that A*, while is perfectly equivalent to A, makes more evident some interesting properties that relate the layers with the matrix. First of all, the relation between the matrix of shortest paths B and A* can be written now in a very clear and simple form, as follows:

Property 1

Consider the sub-matrices A_{hk}, for $h \in Z_h$, $k \in Z_k$ that are component of A*; the matrix B of shortest paths (or optimal links) can be obtained by putting $A_{hk} = \emptyset$ for $k \neq h-1$ within matrix A*.

Property 2

If the graph is not directed, $A_{hk} = \emptyset$ for $h-1 > k > h+1$ because layers k and $h \pm s$, $s>1$ are not adjacent.
The only nonzero sub-matrices of A* are therefore A_{hh-1}, A_{hh+1}, that put in relation pairs of adjacent layers, and A_{hh}, that describes the connections within the same layer.

In general, the shortest path between two nodes is generally not unique and hence many equivalent paths can take origin from one state x^k ; in this case several optimal links will connect state x^k with some of its successors. This means that in the matrix B, for a given state x^k (and row k) there is more than one link to the *successor states*, and consequently more than one optimal action. In this case there will obviously be several shortest paths, which obviously will have the same length.

3.2 Properties of invariant decomposition patterns

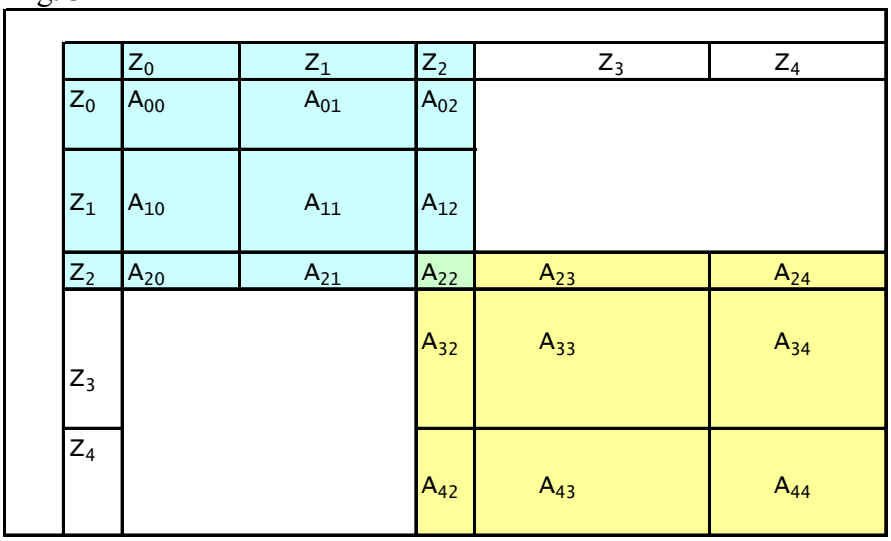
Let us now divide the graph into two parts while using a layer S_h or, which is the same, decompose a problem $S \rightarrow G$ into two sub-problems $S \rightarrow S_h$ and $S_h \rightarrow G$.

Call $S_{\leq h} = \{S_h \cup S_{h-1} \cup S_{h-2} \ldots \cup S_2 \cup S_1 \cup G\}$ the set of layers that have a distance not greater than h to G . Call $S_{\geq h} = \{ S_h \cup S_{h+1} \cup S_{h+2} \ldots \cup S_N \}$ the set of layers that have a distance greater or equal than h to G . The following properties hold: $S_{\leq h} \cup S_{\geq h} = \Gamma$, $S_{\leq h} \cap S_{\geq h} = S_h$). This decomposition of the graph is equivalent to decompose the problem $S \rightarrow G$ into two sub-problems $S \rightarrow S_h$ and $S_h \rightarrow G$.

Property 3. If S_h is a layer, the matrix of shortest paths of problem $S \rightarrow G$ is perfectly decomposable into the matrix of shortest paths of sub-problem $S \rightarrow S_h$ and the matrix of shortest paths of $S_h \rightarrow G$.

To get a simple proof, let consider the permuted matrix A* and decompose it into two sub-matrices $A(S, S_h)$ an $A(S_h, G)$ respectively defined by the layers $\{ Z_0, Z_1, Z_2, \ldots Z_h\}$ and $\{Z_h, Z_{h+1}, \ldots Z_N\}$ (see fig.3)

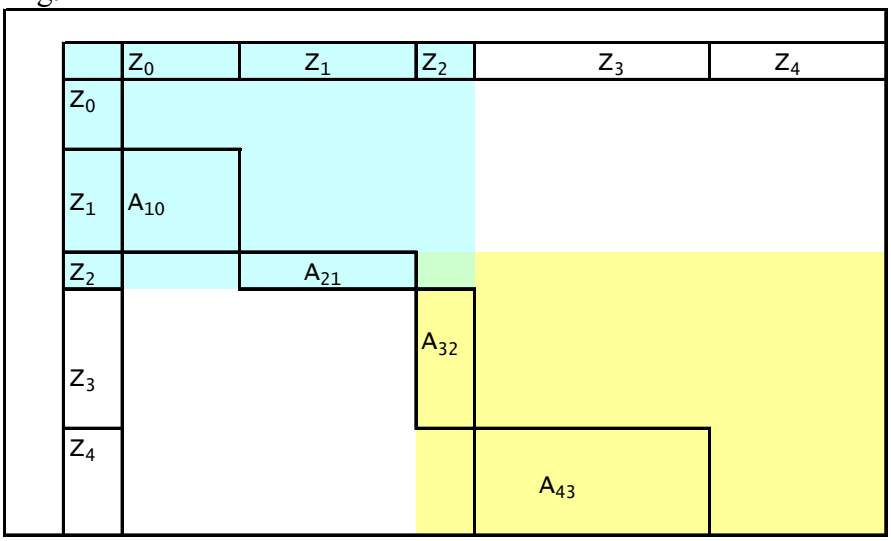
Fig. 3



According to our definition, the sub-problem $S_h \rightarrow G$ is represented by a sub-graph containing the nodes of $S_{\leq h}$ and having layer S_h as "starting" nodes and G as final nodes. $A(S_h, G)$ is the related matrix. Analogously, the sub-problem $S \rightarrow S_h$ is represented by a sub-graph containing nodes $S_{\geq h}$ and having the layers $S_{\geq h}$ as "starting" nodes and S_h as final nodes. $A(S, S_h)$ is the related matrix.

Following Property 1 the matrices $B(S_h, G)$ and $B(S, S_h)$ of shortest paths can be obtained by putting $A_{hk} = \emptyset$ for $k \neq h-1$ respectively within matrix $A(S_h, G)$ and $A(S, S_h)$ .It is therefore evident that B is perfectly composed by the two matrices $B(S_h, G)$ and $B(S, S_h)$ because $B = B(S_h, G) \cup B(S, S_h)$ and $B(S_h, G) \cap B(S, S_h) = A_{hh} = \emptyset$.
Fig. 4 shows the nonzero components of the two matrices $B(S_h, G)$ and $B(S, S_h)$ and make even clear that by optimising separately the paths of the two matrices $A(S_h, G)$ and $A(S, S_h)$ we get exactly the same result than optimising the paths of the global matrix A .

Fig. 4



Hence the matrix of optimal links of the original game is composed by the two sub-matrices of optimal links of the two sub-problems. The decomposition pattern is *invariant*, in the sense that the matrices of the two sub-problems can be independently optimised to get a global optimisation.
From property 3 we may conclude that each decomposition of a problem into two sub-problems, having a layer S_k as separating set, is invariant; moreover, the following equality holds

$$D(S,G) = D(S, S_k) + D(S_k , G)$$

where D is the distance i.e. the length of the shortest path, between S,S' and G.
We may therefore establish the propriety of triangularity of distances, based on the previously seen properties; if C, C',…are the building blocks of a decomposition pattern, then

$$D(C,G) \leq D(C,C') + D(C'G)$$

This inequality will be discussed more carefully in the context of Property 6

Propriety 4 Any decomposition into sub-problems having the layers as separator elements is invariant and therefore by optimising separately every sub-problem defined by a pair of layers we get a global optimal solution.
This property is the extension of Property 3 to any other decomposition having - as building elements - layers ordered according to the distance to the goal. The permuted transition matrix A* can be decomposed perfectly into the sub-matrices $A_{h,h-1}$ (h=1,..N) and correspondingly the original problem is decomposed into the sequence of elementary sub-problems $S_N \rightarrow S_{N-1}$, $S_{N-1} \rightarrow S_{N-2}$, ... $S_1 \rightarrow G$.
A full decomposition pattern will be composed by pairs of adjacent layers, or pairs of subset of adjacent layers.

To understand why the construction of a full and invariant decomposition pattern is quite unusual in human problem solving , note that, once we have got as terminal sub-problems pairs of layers, not

necessarily the sub-problem $S_{k+1} \rightarrow S_k$ can b solved by a unique move, the same for all nodes in S_k. Hence layers are *not* usually categories in a decomposition pattern. However, a pair of layers S_{k+1} , S_k can be decomposed into homogeneous parts, i.e. into pairs of subsets each of which is connected by the same optimal move. We can therefore compact the strategy representation while grouping parts of a layer into *building blocks,* and a single (optimal) move will match each of them so that a compact and invariant representation is achieved. Of course any decomposition patterns where building block are subsets of layers S_k is invariant.
Unfortunately, this kind of decomposition, and the clustering of sub-sets of layers, normally cannot be described by abstract strings based on the natural code description of the game.
The subsets of layers to which it is applied the same, optimal move, in general are *not* classifiers C_s , i.e are aggregations that cannot be represented as categories, insofar are *not* defined on the basis of abstractions . This discrepancy between the sub-sets of layers and a compact representation based on classifiers created by abstractions, is an important origin of the distortions, because human reasoning is fundamentally based on categorization.

Property 5 Any decomposition pattern based on aggregation of adjacent layers maintains the metrical invariance.

If S_1, S_2, S_3,.. are the layers at distance 1,2,3,… to goal G and we create a new goal is G'= G ∪ S_1, the whole system of distances is not changed; node x - which used to be at a distance k to G will now be at a distance k-1 to G', but the same will happen to all the other nodes, and hence the metric will be unchanged. In other words, for each pair of nodes {x, y} not belonging to G∪S_1 we will have d(x, G)- d(x, G ∪ S_1)= d(y, G)- d(y, G ∪ S_1). Consequently, the matrix of minimal links related to goal G and the new matrix of minimal links related to goal G ∩ S_1 are identical in their common parts (the matrix of G∪ S_1 has obviously no links in the rows corresponding to the nodes of S_1). The same is true for goal G∪S_1∪S_2 and for any other combination of adjacent layers .

3.3 Aggregations distorting the metric

We have shown that layers are aggregations which can be made with no modifications of the system of distances to the final goal, i.e. that the decomposition patterns made by layers or combinations of adjacent layers are invariant; the opposite is not necessarily true: in fact, there are invariant decompositions patterns which violate the condition to be based on pure layers or combinations of adjacent layers. Small "deviations" from this rule are therefore allowed, as we will see in this section, and a more general condition for invariance will be discussed in section 3.4. The main properties of invariant decomposition patterns based on aggregations different from layers descend from a property of "triangularity" that we state now.

Propriety 6 - Triangularity in relation to the domain of shortest paths.

For any given state x, it may exist one or many shortest paths to the goal G. Consider the set of shortest paths between x and G (where G may be a set of target nodes).
Call Λ the set of the nodes of such paths. For every x' (x'≠ x, x'≠ G) we have

If x'ε Λ

$$d(x,G)=d(x,x')+d(x',G)$$

If $x' \notin \Lambda$

$$d(x, G)<d(x, x')+d(x',G)$$

Demonstration:
If $x' \varepsilon \Lambda$ the equivalence is valid for the definition of distance: x' is in fact the node of a shortest path between x and G and the distance is by definition the number of nodes in a shortest path.
Let us take $x' \notin \Lambda$. x' obviously belongs to one of the layers of G, let us suppose it is layer k at a distance S_k . If S_k is more distant from G than x, i.e. if $d(S_k, G)>d(x, G)$, then, as $d(x',G)= d(S_k, G)$ we will have $d(x',G)>d(x,G)$; all the more so

$$d(x',G)+d(x, x')>d(x, G)$$

If, on the contrary $d(S_k ,G) \leq d(x, G)$, there surely will be at least one $x^* \varepsilon \Lambda$ in layer S_k, and hence $d(x^*,G)=d(x',G)$. But $d(x, x^*)<d(x, x')$ because $x^* \varepsilon \Lambda$ while $x' \notin \Lambda$.
Hence, $d(x, G)=d(x, x^*)+d(x^*,G)< d(x, x')+d(x',G)$.
This concludes the demonstration.

Let us now examine what happens when a new goal is created by adding a new node G' to an existing goal composed by only one node G.
It is useful to compare the system of distances to the old goal G with the system of distances to the new goal $G \cup G'$. To this purpose, take randomly a node x different from G and G', and consider its distance respectively to G and to $G \cup G'$.
By definition $d(x, G \cup G')=\min(d(x,G),d(x,G'))$. Hence, if $d(x,G') \geq d(x,G)$ the distance label of x is unchanged and we will have $d(x,G)=d(x, G \cup G')$. On the contrary, if $d(x,G')<d(x, G)$ then, based on the definition of distance, $d(x, G \cup G')=d(x,G')$: here the distance label changes and is defined in relation to the new node G' which has been included.
Therefore we can say that the introduction of a new node G' distorts the metric for nodes x of the graph which are "closer" to G' than G. It is intuitive to realize that if G' is far from G large parts of the graph will exhibit relevant distortion to the original metric; in fact consider the nodes x along a shortest path between G and G': for any x that is nearer to G' than to G, the path to the goal is directed to G' and therefore a player starting to x and approaching to $G \cup G'$ along the shortest path, gets far from G. If on the contrary G and G' are adjacent, the distortion of the system of distances is irrelevant, as we will see.

Property 7 - If we add to a target node G some nodes of the first layer, the metric is unchanged

Assume that the goal G is composed by only one node, and call G_1, G_2 ,… G_s the nodes of the first layer S_1 of G.
For every x ($x \neq G$) of the graph, the shortest paths to G starting from x will include a node of the first layer. For a given G_k consider the set B_k of nodes x^i whose shortest paths to G pass through G_k . B_k can be considered a "basin of attraction" of G_k because all nodes $x^i \in B_k$ have a shortest path to G_k. The basins of attraction of different nodes of the first layer may have non empty intersection, i.e. it may happen that from a given x^j there is a shortest path to G passing through B_k and a shortest path to G passing through B_h.

Now let's build up a new target composed by the old one, G, plus a new one, G_h . If the distance of a node x* of the basin of attraction of G_h to G was d(x*, G) =d*, the distance to G∪G_h will be given by d(x*, G∪G_h) =d*-1. In fact, for property 8, d(x*, G) =d(x*, G_h) +d (G_h, G) because G_h belongs to the shortest path from x* to G. As consequence, d(x*,G∪G_h)=d(x*,G_h)=d(x*,G)- d(G_h,G) =d(x*,G)- 1.
If , besides the shortest path passing through G_h, x* admits a shortest path passing through a different node G_k of the first layer, the corresponding link in the matrix of shortest paths must be set to zero. In fact, call x' the successor of x* along the shortest path from x* to G through G_k. Since d(x',G)=d(x*,G)-1=d*-1= d(x*,G∪G_h), and d(x',G)=d(x', G∪G_h), x* and x' have the same distance to G∪G_h .Therefore in the matrix of shortest paths the coefficient that links x* with x' must be deleted.
Therefore along the boundary between two basins of attraction to G some links must be set to zero. This reduces the number of shortest paths from some nodes to the goal, but does not modify the residual links, because the shortest path from x* to G∪G_h coincides with the shortest path from x* to G for all steps (excluding of course the last one, from G_h to G which must be obviously deleted).
Beside to delete some links into the array of shortest paths, the creation of the new goal G∪G_h re-establishes new links; in fact if in the transition array there was a links a_{ij} between x^i, in the basin of attraction of G_h and x^j in the basin of attraction of G_k , and in the matrix of shortest paths to G this link was deleted because x^i and x^j were at the same distance to G, i.e. d(x^i, G)=d(x^j, G), now the link a_{ij} must be re-established because x^i is nearer than x^j to the new goal G∪G_h : d(x^i, G∪G_h)=d(x^i,G)-1=d(x^j,G)-1.
Consequently x^i is now the successor of x^j along a new shortest path from x^j to G∪G_h .This new path from x^j is added to the pre-existing shortest path from x^j to G through G_k, and therefore the new path does not alters substantially the matrix of shortest paths.
Summing up, the introduction of a new goal node, adjacent to the old one, does not modify substantially the structure of the matrix of shortest paths; while some coefficients are deleted, and other coefficients which was zero are re-established, the matrix M(G) of shortest paths to G and the matrix M(G∪ G_h)of the shortest paths to G∪ G_h have, for every row j at least one common coefficient a_{jn}. In consequence the matrices define at least one common shortest path from every starting node x^j.
The reasoning we have conducted so far can be extended to the case in which more than one node of the first layer is added to the original target-node G, and therefore we can conclude that the introduction of new adjacent nodes (belonging to the original first layer) , while requires cuts and seams to the graph of shortest paths, preserves at leas one shortest path to the goal from every starting node.

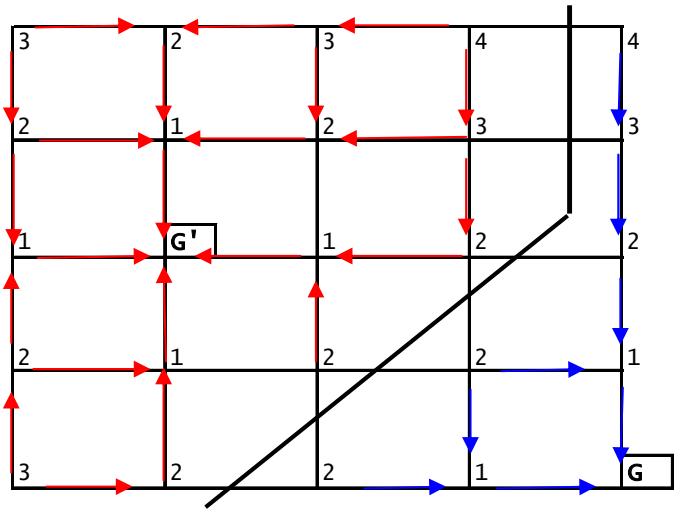
There is an intuitive explanation of property 9 that helps to understand why the distance among an old target node and new targets to be added matters.
If, in fact, we add new adjacent nodes to the former node G, the shortest paths to it are in the same direction as the shortest paths to G∪G_k∪….G_h. As we will see immediately, if the new nodes G_h are far from the former node G a part at least of the shortest paths change direction, because by approaching to some G_h the path get far from G.

Large distortions

Let us now include a new goal G' at a greater distance than 1 from the old goal G. In this case a separation is created between a sub-graph that contains the shortest paths that are directed towards G and the sub-graph that contains the shortest paths directed towards G'. Two sets are here generated, too: the set of points x' which are nearer to G' than to G, that is for which d(x',G')<d(x',G) and vice versa the set of points x closer to G, i.e. those for which d(x,G)<d(x,G') (see fig. 5). The frontier is composed either of nodes x^F for which d(x^F,G)=d(x^F,G') or is composed by pairs of nodes x^F ,$x^{F'}$ for

which $d(x^F,G)=d(x^{F'},G')$. The basins of attraction of the two nodes G and G' do not have any common shortest path. In this case, the shortest paths starting from points x' cannot pass through G (because G' is closer to x' than G) and hence the player, while following a minimal path towards G' moves away from G.

Fig. 5



Suppose that a player, in order to reach the final goal G, uses G ∪G' as an intermediate goal. For states x' the player, while moving along the optimal path to G', moves away from G; this means that the triangular inequality analysed above $d(x', G)< d(z',G') + d(G',G)$ holds.
Therefore the player, while moving along the optimal path to a sub-goal moves away from the final goal and does not optimise its global path.
The features we have seen so far can be easily generalised to the case where goal G is a set of nodes.

3.4 Further properties of decompositions, beyond the layers

Remember that a decomposition has been defined *invariant* if the optimisation of the sub-goals leads to the optimisation of the global problem.
Property 8 A decomposition of a problem $\Psi \rightarrow G$ into two *and* sub-problems $\Psi \rightarrow R$ & $R \rightarrow G$ is *invariant* if and only if for all ($x \in \Psi$, $y \in R$) the equality of distances holds:

$$D(x,G)=D(x,y)+D(y,G)$$

Because of property 6, if every element of R belongs to a shortest path between Ψ and G, ($\forall x \in \Psi$, $y \in R \rightarrow y \in \Lambda(x)$) then the equality of the sum of the distances holds; this means that any shortest path from x to G pass through a node $y \in R$, and therefore the path can be split in two paths, from x to y and from y

to G, each of which is the shortest in his domain. Consequently by optimizing separately the two sub-problems we get two parts of the same shortest path, joined in some $y \in R$.
Vice versa, again because of property 6, if some element of R does not belong to any shortest path between Ψ and G the inequality

$$D(x,G)<D(x,y)+D(y,G)$$

holds for some ($x \in \Psi$,$y \in R$) and therefore the decomposition is not invariant.
But if all the shortest paths between Ψ and G pass through an element of R, each shortest path between Ψ and G can be split into a shortest path between Ψ and R and a shortest path between R and G. From this we may conclude that

Property 9 if and only if a decomposition of a problem $\Psi \rightarrow G$ into two sub-problems $\Psi \rightarrow R$ & $R \rightarrow G$ is invariant, R is a *barrier* with respect to the nodes of Ψ.
In fact if each shortest path from an element of Ψ to G passes through R i.e. R is a barrier, D(x,G)=D(x,y)+D(y,G) for Property 8.
Vice versa if the equality holds, i.e. if $\forall x \in \Psi$, $\exists y \in R$: D(x,G)=D(x,y)+D(y,G) , all nodes $y \in R$ are either starting states or intermediate states within a shortest path from Ψ to G. Therefore there are no shortest paths which do not pass through R.
Layers are a clear examples of barriers.

Consider now a decomposition developed iteratively until the terminal sub-problems are reached. If the decomposition of the game $\Psi \rightarrow G$ in $\cup$ & $R \rightarrow G$ is further developed, $R \rightarrow G$ will in its turn be decomposed into an *and* sub-problem or specified in *or* sub-problems, and the same applies to $\Psi \rightarrow R$.
We can apply the same considerations as previously: namely that if the $R \rightarrow G$ problem is decomposed into two *and* sub-problems, $R \rightarrow C$ & $C \rightarrow G$, the new decomposition will be invariant if the nodes of C, too, will act as a barrier between R and G.
If the $R \rightarrow G$ problem is instead decomposed into *or* sub-problems, the player will decompose R into sub-sets Ψ_1 Ψ_2 …. $\Psi_k \subseteq R$ so that $\Psi_1 \cup \Psi_2$ ….$\cup \Psi_k = R$, $\Psi_h \cap \Psi_k = \emptyset$ which constitute the points of departure for further sub-problems $\Psi_k \rightarrow G$. These new sub-problems will in their turn be decomposed into *and* nodes.
Proceeding in this manner, the decomposition of each sub-problem terminates when reaching the *terminal* sub-problems $\Psi_k \rightarrow R_k$, whose solutions consist of only one move. This obviously means that the nodes of Ψ_k are adjacent to the nodes of R_k. If the decomposition is invariant, all elements $x'^i \in R_k$ are nearer to the goal than their predecessors $x^i \in \Psi_k$ along the path to the goal. Otherwise, some x'^i will be nearer, other x'^i will be farther than his predecessor to the goal.
As we have previously remarked, the move that solves a terminal sub-problem, may be different for any different element $x^i \in \Psi_k$, or the same for all elements $x^i \in \Psi_k$. In the last case, if the elements of Ψ_k in turns can be represented as a unique category, the elementary problem is represented in one only condition-action statement.

Property 10 - In a decomposition developed until the terminal sub-problems $\Psi_k \rightarrow R_k$ have been reached, if every R_k acts as a barrier for its Ψ_k, the decomposition is invariant.

A complete decomposition pattern consists of an *and/or* set of elementary sub-problems $\Psi_h \rightarrow R_h$.
If the successor of each $x \in \Psi_k$ in R_k belongs to the shortest path from $x \in \Psi_k$ to G, all the successors of an $x \in \Psi_k$ are comprised in another building block Ψ_h (or in G). For every R_k, in fact, there is a

minimum cover U_k composed of the union of some Ψ_h: that is, there exists a $U_k = \Sigma \cup \Psi_j$ such that $R_k \subseteq U_k$, $U_k \cap R_k = R_k$ e $U_k \cup R_k = U_k$
Consequently, every $x \in \Psi_k$ has a successor in a $x' \in \Psi_h$ which is one step closer to G. Hence a minimal global path between a given x and G can be decomposed as follows: given x, the building block Ψ_k to which it belongs is identified and then the terminal sub-game $\Psi_k \rightarrow R_k$. This sub-game is resolved by a sub-matrix A_{kh} which associates with each $x \in \Psi_k$ a move m_w which puts x in a $x' \in \Psi_h$ which is one step closer to G. In its turn, x' belongs to a sub-problem $\Psi_h \rightarrow R_h$, so that there is a sub-matrix A_{hs} which associates with each $x' \in \Psi_h$ a move m_w that puts x' in a $x'' \in \Psi_s$ one step closer to G. All the concatenated sub-problems $\Psi_k \rightarrow \Psi_h$, $\Psi_h \rightarrow \Psi_s$ ….,$\Psi_v \rightarrow G$ and their optimal solutions are iteratively identified until objective G is reached along a minimal path. Thus, by resolving each sub-game with an optimal move, the minimal global path from x to G is constructed.

3.5 Non invariant decomposition patterns

Conjectures and biases

Of course in real contexts players may build up a decomposition pattern in a very chaotic way, sometime disregarding the requisites of logical coherence typical of the *and/or* trees creation. We do not pretend to model the decomposition *process*, i.e. the process of creation of a decomposition pattern, which is matter experimentally unexplored . We simply note that the requirements for a decomposition pattern may be relaxed; in particular we can in some situations relax the *or* decomposition feature, by assuming that for some $\Psi_k \rightarrow R_k$ we may have $R_k \subseteq \Psi_1 \cup \Psi_2$ ….$\cup \Psi_k$; moreover we can relax the *and* decomposition requirements, by admitting the property $\Psi_k \cap \Psi_k \neq \emptyset$ for some building block.
Whatever the features of a decomposition process are, if the process is developed until the terminal sub-games have been reached, the states of each pair $\{\Psi_k\ R_k\}$ are adjacent because they are connected by one single move. On identifying the relation $\Psi_k \rightarrow A_{ij} \rightarrow R_k$, the player implicitly assumes that the move takes him one step *closer* to the goal, and therefore that all the states of the set R_k are one step closer to the objective with respect to the states of the set Ψ_k.
This means that players performing the decomposition *conjecture an order* among pairs of sets (Ψ_k, R_k) that generally are categories. It must be emphasized that if Ψ_k are categories including very large sets of configurations, the attempts of players to establish an order among categories are not contrasting with bounded rationality assumptions. Players in fact try to decompose the problem in very few, general categories, easy to be compared, focusing the attention on few salient features of the game, simplifying as possible the game description and therefore conjecturing an order among a very limited number categories.
If player's conjecture is correct, i.e. if in every subproblem (Ψ_k, R_k), R_k is closest to the goal than Ψ_k , than the strategy yielded by the decomposition is optimal and the decomposition is invariant.
If the conjecture is not correct, errors occur which are often difficult to detect. In fact, categorization enables identification of terminal problems $\{\Psi_k\ R_k\}$ which are solved by applying the same move m_w to all the states of the elementary component Ψ_k to achieve the partial objective R_k. However, there is often no single action which, when applied to $\Psi_k \rightarrow m_w \rightarrow R_k$, yields a set of adjacent states all one step closer to G. On the contrary, some of these $x^i \in \Psi_k$ states are such that move m_w takes one step away from the objective, while other $x^j \in \Psi_k$ states take it one step closer to the objective. In this case the player has made a partly inaccurate conjecture in which the error is generated by the categorization.

Moreover, having identified the terminal sub-problems, and having assumed that the solution consists in applying the same move to all the components of the sub-problem, even if the player optimizes each single sub-prolem (with the constraint just stated) he will not be able to achieve the global optimum.
In fact, if $x^i \in \Psi_k$ ed $x^j \in \Psi_k$ and the player has identified the solution $\Psi_k \rightarrow m_w$, and if move m_w applied to x^i leads to a (closer) successor whereas if it applied to x^j it leads to a node further away, the player cannot improve his strategy. Even if he attempts to apply moves different from m_w to the $x^j \in \Psi_k$, its solution cannot achieve the optimum because there is no single move that makes this possible. The player must therefore modify the composition of the building blocks Ψ_k, assuming a decomposition pattern.
We have already pointed out that when a player modifies a condition-action instruction, if the condition is given by a category, he modifies the action to be carried out in relation to all the configurations that belong to the category. Therefore by introducing a representation of a strategy based on categories the representation is simplified but this limits the set of possible elementary modifications that can be applied to achieve the optimal solution. Consequently, as we have seen, some decomposition patterns of the problems necessarily produce sub-optimal strategies. *This implies that the larger is the domain of the categories, the lower the number possible modifications to achieve the optimum.*

Fitness

The sum of the distances between every starting state and the goal is a benchmark to measure the efficiency or the *fitness of the strategy* and compare the "deviation" of different decompositions patterns to optimality.
We can compare different types of decompositions patterns by associating to each of them, as a measure of *fitness*, the length of the paths from every starting state x^i to the goal, defined as follows: given the list of instructions of the strategy derived from a given decomposition pattern, we can calculate the length of path $L(x^i, G)$ between every starting configuration x^i and the goal G *based on the building blocks of the decomposition pattern*. The length of $L(x^i, G)$ is the sum of the lengths of the paths between every pairs of sub-problems that compose the problem. If a problem is fully decomposed, the length of the paths of the terminal sub-problems is 1, and therefore $L(x^i, G)$ is simply equal to the *number* of terminal sub-problems that are involved in the decomposition.
The sum of lengths $L_{TOT} = \sum_i L(x^i, G)$ is called "global path" .
Note that

$$L(x^i, G) \geq D(x^i, G)$$

Where $D(x^i, G)$ is the *distance* between x^i and G, that is the number of steps of the shortest path, or, equivalently, the label of distance associated through the *backward branching* algorithm;
L_{TOT} will vary with the different decomposition patterns and, within the same decomposition pattern, with the moves associated to every building block . For all invariant decomposition patternsL_{TOT} will have the same value, the minimum, that is $\sum_i L(x^i, G) = \sum_i D(x^i, G)$ and $L_{TOT} = \sum_i L(x^i, G) = \sum_i D(x^i, G)$. For the not invariant decomposition patterns we will have $L(x^i, G) \geq D(x^i, G)$ and

$$\sum_i L(x^i, G) > \sum_i D(x^i, G).$$

Hence $\sum_i L(x^i, G)$ can be naturally assumed as a measure of the fitness of a strategy (and of the associated decomposition pattern)

Suboptimalities

The non invariant decomposition patterns give rise to sub-optimal strategies for the reason we have just seen: $\sum_i L(x^i,G) > \sum_i D(x^i,G)$. But they display a more subtle property: they are - in general - optimal in a sub-domain of the configurations of the game. In fact, once a strategy is constructed according to a decomposition pattern, if the corresponding move of a given building block Ψ_i is m_h, and if x^r and x^s are two configurations of Ψ_i it may happen that for configuration $x^r \varepsilon \Psi_i$ we have $L(x^r,G) > D(x^i,G)$ while for $x^s \varepsilon \Psi_i$ we have $L(x^s,G) = D(x^i,G)$., because, as we have shown, the same move m_h, applied to two different states of Ψ_i, can get the player nearer or farther to the goal.

This means that in a non invariant decomposition pattern there can be numerous configurations, for every building block, for which the moves of the strategy are optimal. If the number of configurations for which the moves of the strategy are sub-optimal is low, it becomes extremely difficult to discover them, and a player can persist in using a strategy, without perceiving that there are sub-optimal characteristics.

Landscape

We have assumed the "total path" $L_{TOT} = \sum_i L(x^i,G)$, as a measure of the *fitness* of a strategy. Every strategy is based on a decomposition pattern, and therefore the fitness varies with the decomposition pattern discovered by the player. For a given decomposition pattern, a player can try to modify the fitness of his strategy by modifying the moves, within the constraint imposed by the building blocks.

It is possible to demonstrate that, if there are not restriction on the moves to be selected within the transition matrix, the fitness function admits absolute minimums (all equal) corresponding to the sum of the shortest paths. Moreover the minimal value is reachable by modifying the moves by trial and errors, despite a positive epistatic degree. (Egidi 2000, 2002). An hypothetical player that builds up his strategy in the extended form can therefore improve it while modifying by trials and errors the actions matching every condition of the program, until the optimal value of L_{TOT} is reached.

This property seems to be in contrast to the behaviours that have been observed experimentally, which indicate that individuals insist in using sub-optimal strategies, and do not modify them incrementally for improvement, as it would be possible. The notion of building block allows us to explain the apparent contrast: the strategies adopted by players are not in extended form - they are in fact compact-, and based on building blocks allowing much less mental effort thanks to categorization. However, the building blocks limit the possibilities to modify the moves of a strategy, because they constitute a constraint: all configurations of a given block must have the same move, and therefore if the move is modified to be applied to configuration $x^i \varepsilon \Psi_i$ one must also modify the moves of all other configurations of Ψ_i in the same way. Therefore, in the search of a minimal value of L_{TOT}, the imposed constraints only allow to reach a local minimum.

Consequently what makes a relative minimal value stable is the difficulty that players encounter in modifying the set of selected blocks, i.e. the decomposition pattern, and to adopt a different one. This difficulty depends on the fact that building blocks are represented abstractly and synthetically, they are in fact *mental categories*, that are obviously difficult to modify.

A strategy generated from a non invariant decomposition pattern therefore admits a *stable* local optimum in the sense that, for a given decomposition pattern, once one relative minimum has been identified, if the actions matching the building blocks are modified, the programme turns out to be less efficient, that is the number of moves necessary in order to reach the goal increases. Therefore, the solution based on the non invariant decomposition pattern though being sub-optimal cannot be improved - it is "locally" optimal and stable. In order to improve the strategy it is therefore necessary to modify the building blocks, i.e. introduce a new decomposition pattern.

Extrapolation and case based reasoning

It is a matter of understanding how players create the "wrong" building blocks, by discovering sub-optimal strategies where they remain entrapped.
If individuals are involved in a game or puzzle where they have to find a resolutive strategy, instead of having to solve a problem theoretically, with the necessary time, and if the game is repeated, the learning process cannot be implemented as it happens when the solution is looked for theoretically, a different learning system is implemented, based on extrapolation from specific cases.
As we will see from the experiment in the next section the players identify a series of building blocks based on experience acquired during the different runs of the game while *trying to derive general rules from individual experiences.* When possible, the players apply a mental exercise of generalization-extrapolation while widening as much as possible the domain of validity of the rules that they have discovered.
In general, the extension procedure creates systems of rules that are described with a certain degree of abstraction (the building blocks). The system of rules that constitute a strategy is complete, that is any configuration of the game has its corresponding rule, even if the player does not have direct experience in most configurations. [2] Also here the crucial mechanism is the symbolic representation and abstraction and categorization based on the focalisation of the salient features of the game.
An attempt is made by extrapolation to identify a decomposition pattern that is valid in the series of repeated experiments; but the classifications and abstractions can comprise non perceivable hidden errors, as we noted in the paragraph on *sub-optimalities*.
To identify them we should examine all existing "condition-action" relationship in detail in all possible cases of the game. This is an extremely time-consuming practice that would nullify all the advantages of a compact representation. Therefore subjects cannot actively search for the exceptions to the rules that they have established in order to resolve a problem. The example of MiniRubik (section 4) shows that one should be guided by the exceptions that have been identified to correct the errors hidden in abstract rules.

The locally optimal solutions where subjects might be entrapped when analysing a problem are therefore due to the limits in their ability to *falsify the rules* they have defined in all the related domains and to discover the hidden errors. When subjects perceive a new example that can be resolved more efficiently with the activation of a new rule in place of the standard rule, they have two possibilities: consider the new example as an exception to the previous general rule or try and define again the building blocks of the system of the rule, resolving the conflicts between the rules. This is extremely mind-consuming and therefore, although a single mutation may improve the *performance* of the system, it would probably be considered an exception so abstract sub-problems would not be redefined. In the event the number of "exceptions" becomes too high - and this happens systematically during a game - individuals cannot simply continue to keep new exceptions in mind: they must in fact "restructure" the space of the rule, re-codifying the information. In other words, they must modify the representation. Such modification might be very discontinuous since it generally requires that the problem decomposition is de-structured, and the problem has to be reconstituted with new building blocks.

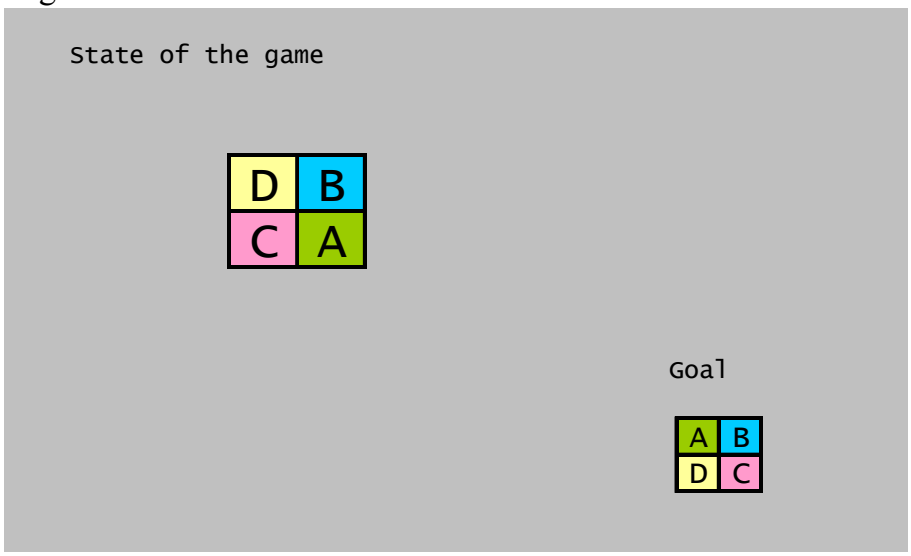
[2] because some simple abstract conditions can encompass all ignored configurations.

The errors hidden within sub-optimal representations can essentially be found and corrected while "re-thinking" the representation, that is formulating a new division into sub-problems - and modifying it. [14]

4 An application: Minirubik .

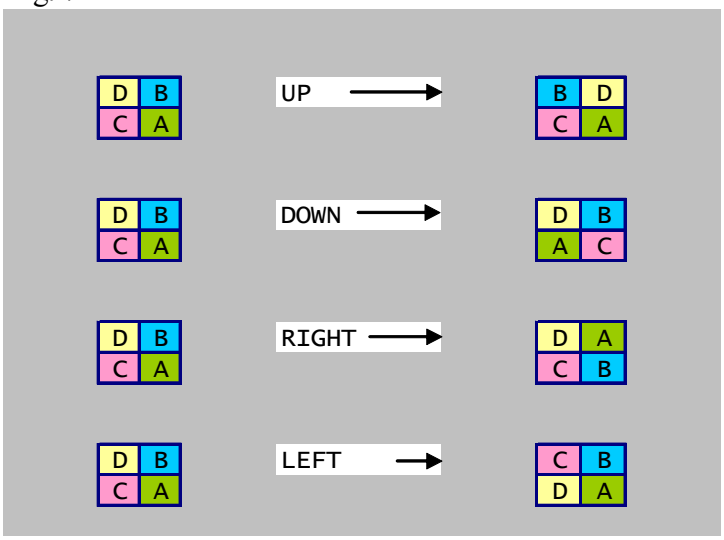
Minirubik is a simplified version of Rubik cube. In the experiments that we will illustrate, the player sits in front of the screen of a computer where two squares composed of four tiles of different colours are displayed (fig.6). On the bottom right of the screen, the arrangement of tiles to be reached as a goal is displayed. In the centre, tiles are arranged differently and the player can modify the arrangement, exchanging tiles horizontally or vertically, until the goal configuration is achieved. (For the sake of simplicity, when necessary, we will use letters, A, B, C, D instead of colours).

Fig. 6



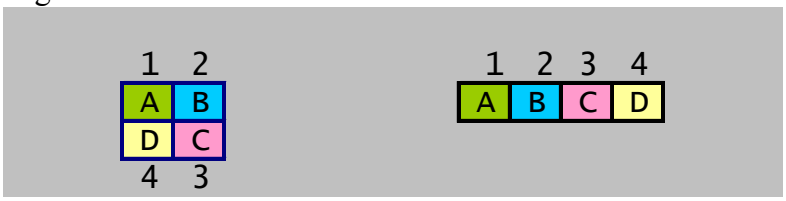
As indicated in Fig.7, tiles can be moved and exchanged in the horizontal or vertical direction. The players must reach the final configuration

Fig. 7



The players gain points according to the number of moves they make to reach the goal: they are granted an initial sum which diminishes at every move they make: the lower the number of moves made to resolve the problem, the greater the residual amount that remains at the end of the game.
In order to simplify the description in the discussion that follows, a configuration can be indicated as a sequence of four letters (or colours) instead of a square of four letters (or colours), applying the following rule: start from the upper left corner of the square and list the elements in the square, moving clockwise. The positions of A, B, C and D in Fig. 8 are positions 1, 2, 3 and 4, respectively.

Fig. 8



With this representation strings can be written like groups of four letters (or colours), like for example CBDA or A##C that represents the set of configurations where A is the first position (upper left corner) and C is the last (lower left corner). The transition matrix is illustrated in fig. 9

Fig. 9

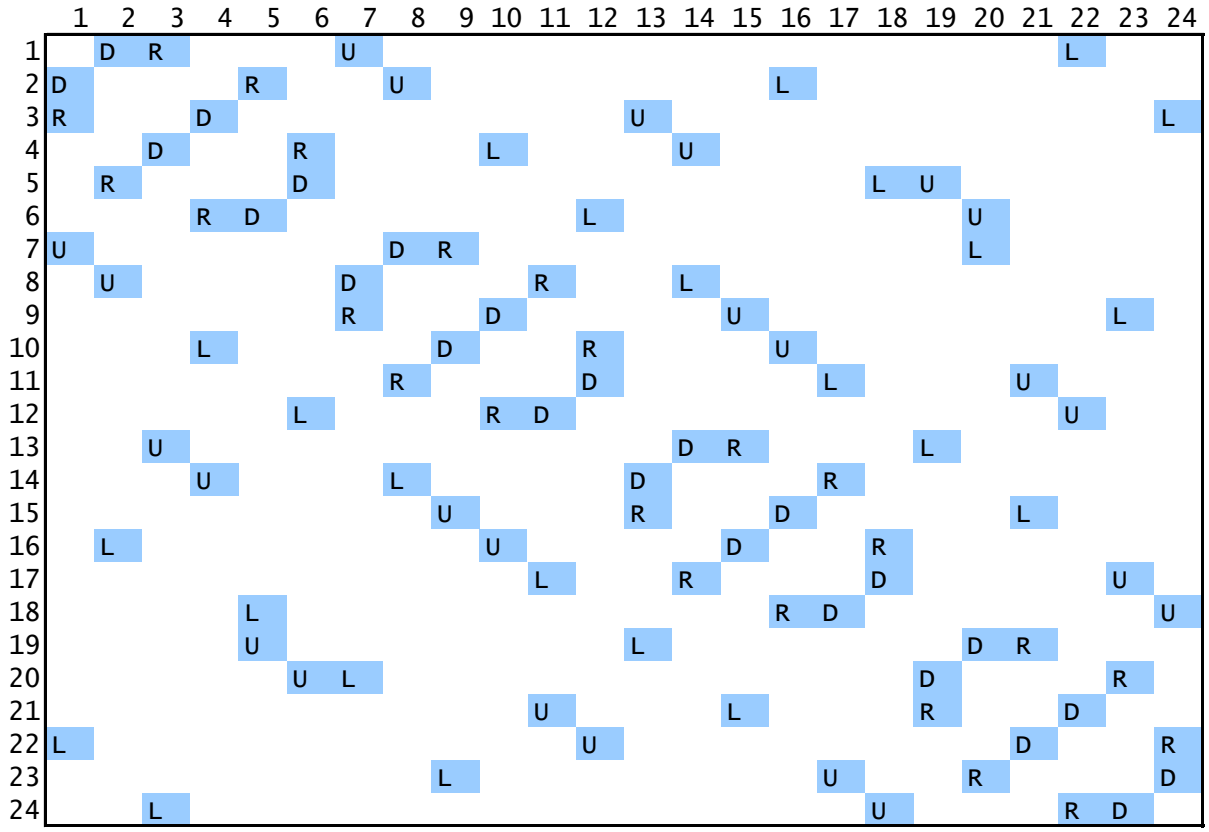


Fig.11 shows the game's optimal solution obtained with the *backward branching* method discussed earlier. By permuting the rows and columns of the matrix according to the indexes of the layers given in fig.10, we obtain the matrix of fig. 11, in which, as shown above, the coefficients in the sub-matrixes A_{hh-1} (to the left of the main diagonal) indicate the adjacency matrix of the game's shortest paths.

Fig. 10

Layers	Distance from G	States
G	0	1
S_1	1	2,3,7,22
S_2	2	4,5,8,9,12,13,16,20,21,24
S_3	3	15,19,18,11,14,10,23,6
S_4	4	17

Fig. 11

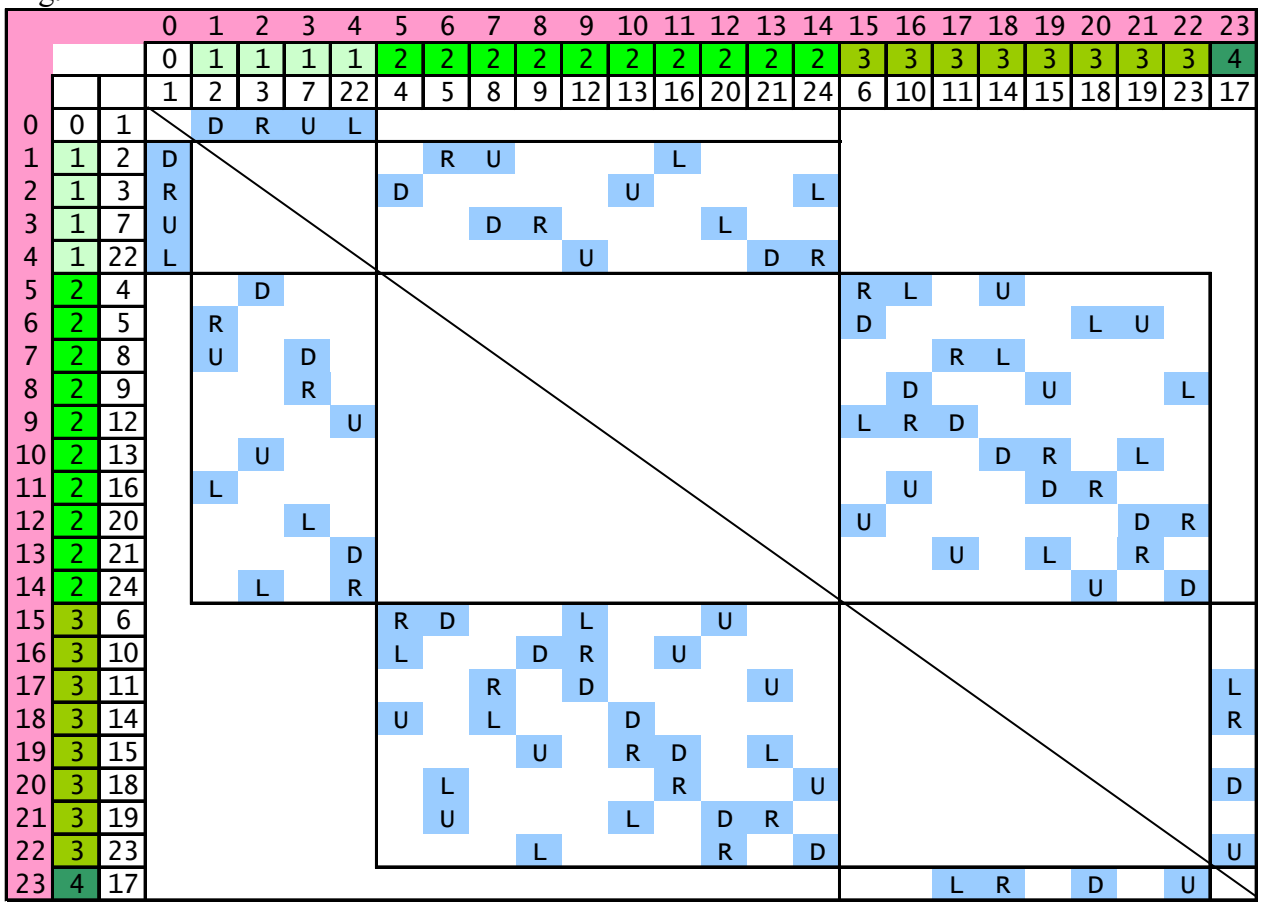
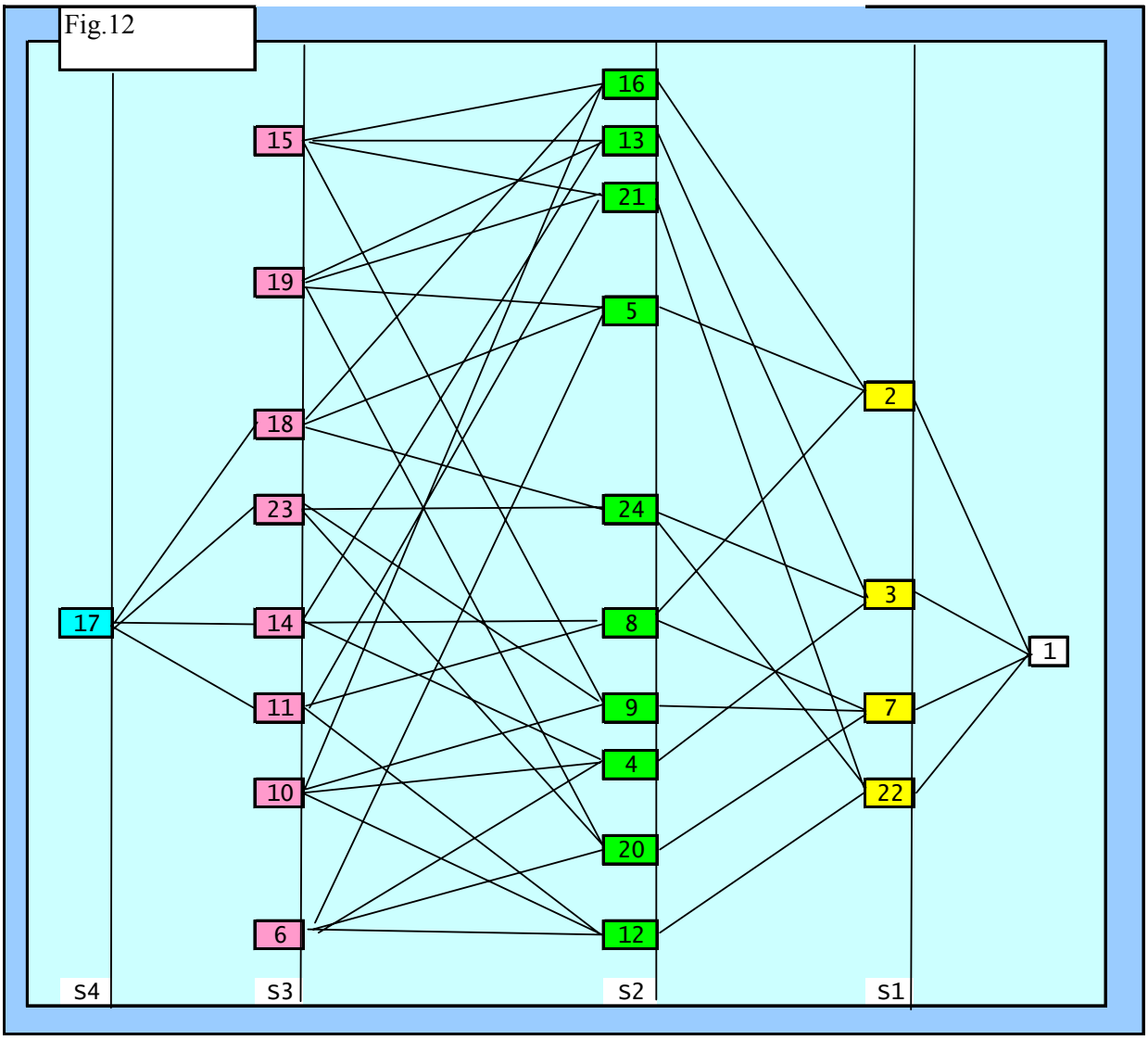


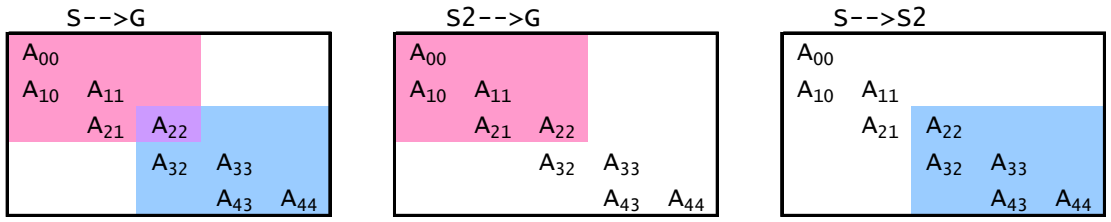
Fig. 12 below illustrates the graph of the game. The configurations are ordered in relation to the distance from the final goal, as in the matrix of shortest paths (to the left of the main diagonal in fig. 11). The layers are clearly identified by vertical planes which separate groups of nodes in the graph. The shortest paths are determined as a consequence.



4.1 Invariant decomposition patterns

The reader can easily ascertain the properties demonstrated in paragraph 3.2: in particular, when decomposing a problem into sub-problems having layers as separators, the metric of the decomposition is unvaried. If the set of nodes at distance 2 is chosen as an intermediate set, that is S_2 ={4, 5, 8, 9, 12, 13, 16, 20, 21, 24} and problem S → G is decomposed into two sub-problems, S → S_2 and S_2 → G , (S= $S_4 \cup S_3$) the matrices of optimal links are in turn reported in the table: one can immediately verify that the optimal path from every node $s_1 \varepsilon$ S to G is exactly the sum of the optimal path from s_1 to a node of S_2 plus the optimal path from S_2 to G. This decomposition pattern has therefore preserved the metric of the game. The matrices of shortest paths of sub-problems S → S_2 and S_2 → G are reported in Fig.13.

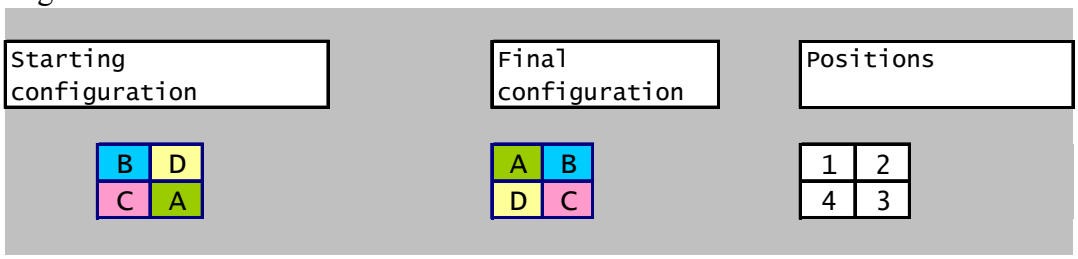
Fig. 13



4.3 Saliency-guided decomposition patterns

Obviously players do not calculate the optimal distance the way we have seen. They try to represent a problem in a simplified way, concentrating on some properties of the game and creating, by abstraction, the elementary building blocks. There are many possible decomposition patterns (every structure of aggregation of states is a decomposition pattern) but some are simpler to represent and are therefore preferable for the low number of calculations needed to implement them. A strategy that has been frequently implemented by players in some preliminary experiments, is based on a simple and sequential decomposition of the problem, and players use it to place pieces orderly, one at a time until the final position is achieved. Let us see an example and suppose that the starting and final configurations are strings BDAC and ABCD respectively, as indicated in Fig.14

Fig.14



The sequential strategy to move from this particular starting configuration to the final configuration suggested by players is composed of the following instructions:

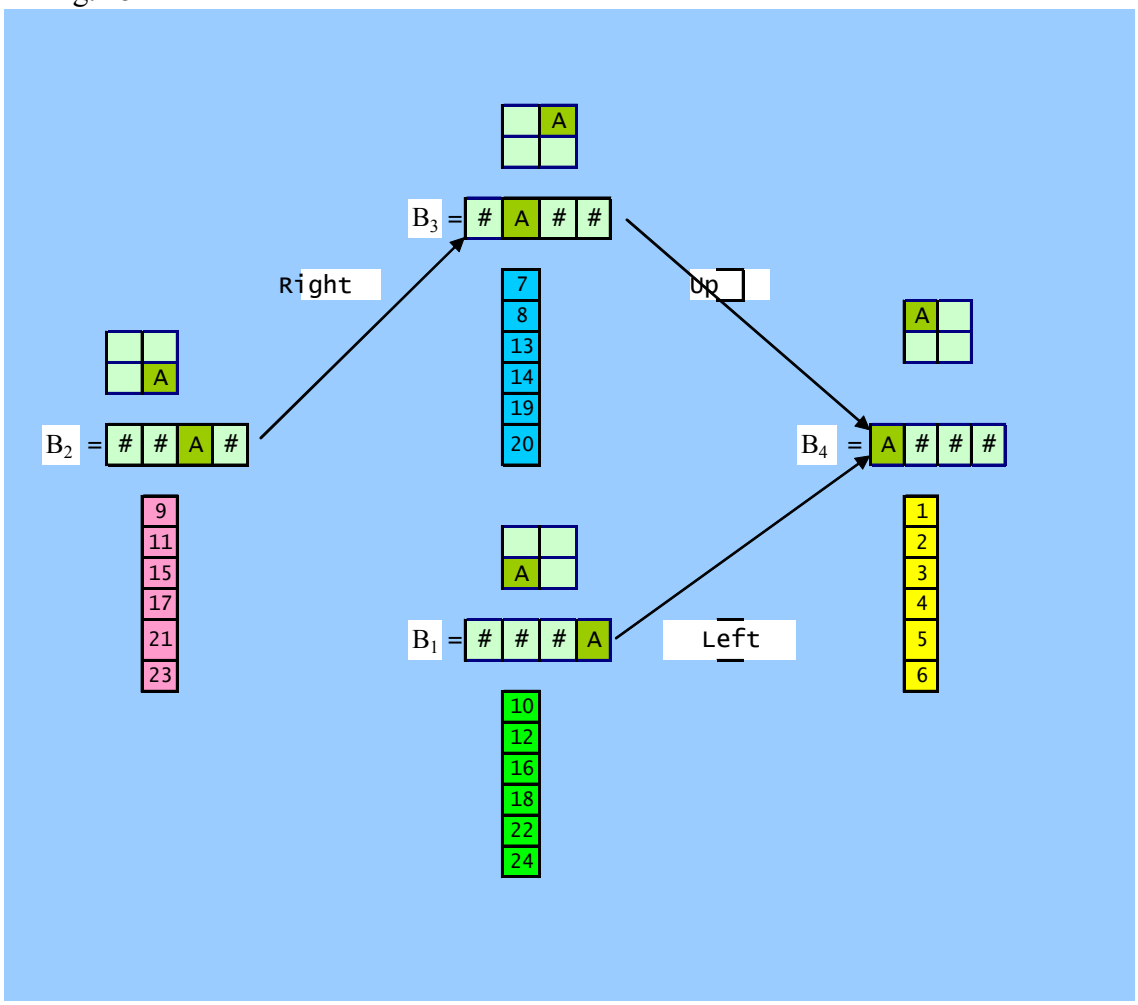
Sequential strategy "First A":

1 - Move A from the starting position, anti-clockwise, until the final position (position 1).
2 - If B is not yet in the final position, move it to position 2, leaving A in its position.
3 - If C and D are not yet in the required final positions, exchange them.

We can immediately appreciate that this sequence of instructions can be applied to any other starting configuration, only by modifying the first instruction in the most "natural" way, that is moving A from the starting position to position 1 with the least number of moves.
This strategy, where one must concentrate on the position of a tile at a time (first move A, then B and finally C and D) is based on the decomposition into three sub-problems, each one being implicitly based on an adequate categorization: the categories A###, #A##, ##A#, ###A, A#B #…. that players

order in their minds to decide how to approach the goal. These categories constitute the building blocks of the First A decomposition pattern.
Figure 15 illustrates the diagram of the first part of the strategy: the sub-problem consists in how to move A from its starting position to position 1. In our version of the problem, the player decides to move A anti-clockwise if A is in position 3, but this solution displays similar properties even if the player moves the tile clockwise.

Fig. 15



The building blocks on which the representation is based therefore include categories B_1= ###A, B_2 = ##A# B_3 = #A##, B_4 = A###. When the player plans to move A from position 3 to position 2, and finally to position 1, he implicitly *orders the three building blocks* B_1 , B_2 , B_3 in terms of distance to the goal G=ABCD. In fact the player presupposes that a configuration belonging to B_3 = #A##, is nearer the goal compared to a configuration that belongs to B_2 = ##A# or B_1= ###A. Therefore, if D is the distance between block B_1 and goal G, we will have

$$D(A###, G) < D(\#A\#\#,G) < D(\#\#A\#,G).$$
$$D(A###, G) < D(###A,G) < D(\#\#A\#,G)$$

This system of relative distances can be illustrated as follows:

B_1 = ###A = (10,12,16,18,22,24) ; Presumed distance = 1
B_2 = ##A# = (9, 11, 15, 17, 21, 23); Presumed distance = 2
B_3 = #A## = (7,8,13,14,19,20) ; Presumed distance = 1
B_4 = A### = (1,2,3,4,5,6); Presumed distance = 0

Therefore the player classifies the building blocks according to an order that defines the relative distances to the goal. This clarifies the reasons why errors appear: if such a system of distances defined at an abstract level of representation is *isomorphic* with the minimal distances that are obtained by applying the method of *backward branching* to the extended description, then the decomposition in B_i blocks is optimal while – if this is not the case - the decomposition will introduce hidden errors.
In order to verify such property a graph of the game has been drawn in the extended form (Fig. 15), associating to every node the blocks defined in Fig.14, according to the *presumed* distance to the goal. The results are quite interesting: the elements of building block B_4 = A### = (1,2,3,4,5,6) are not to be found at the same distance to the goal and they do not fulfil the properties seen in par. 3.3; they therefore introduce a systematic distortion in the system of the distances to the goal.
At the same time, if we consider the building block B_4= {1,2,3,4,5,6} as a goal, the other building blocks B_1 , B_2 , B_3 are layers, or - in other words – sub-systems of layers at a constant distance B_4 . In fact, if the system is constructed by means of the backward branching method, the set of nodes 1, 2 …from B_4 , ($B_1 \cap B_3$) is at distance 1, and B_2 is at a distance 2. The strategy "First A" effectively identifies the optimal moves that connect building blocks B_1 , B_2 and B_3 of this sub-problem.[3]
If we compare the distances from nodes in the graph to node 1 with those in set B_4 (1,2,3,4,5,6) one discovers that they coincide only in part with the nodes of the graph. In two cases, and precisely nodes 12 and 20, this does not happen; the reason for this is that 12 and 20 are the two nodes that are nearer to 6 than to 1: d(12,1)>d(12,6) and d(20,1)>d(20,6). The shortest path from node 12 to node 1, that is (12→22→1) does not coincide with the shortest path to approach B_4={1,2,3,4,5,6}, - that is (12→6) – from node 12; in other words, among the nodes in B_4, node 6 is closer to node 12 than node 1: d(12,1)>d(12,6). The same applies to node 20. The sub-optimalities are generated because configurations that belong to block B_4 do not respect the rules of distance from the final goal G, and that *distorts* all the space of relative distances.

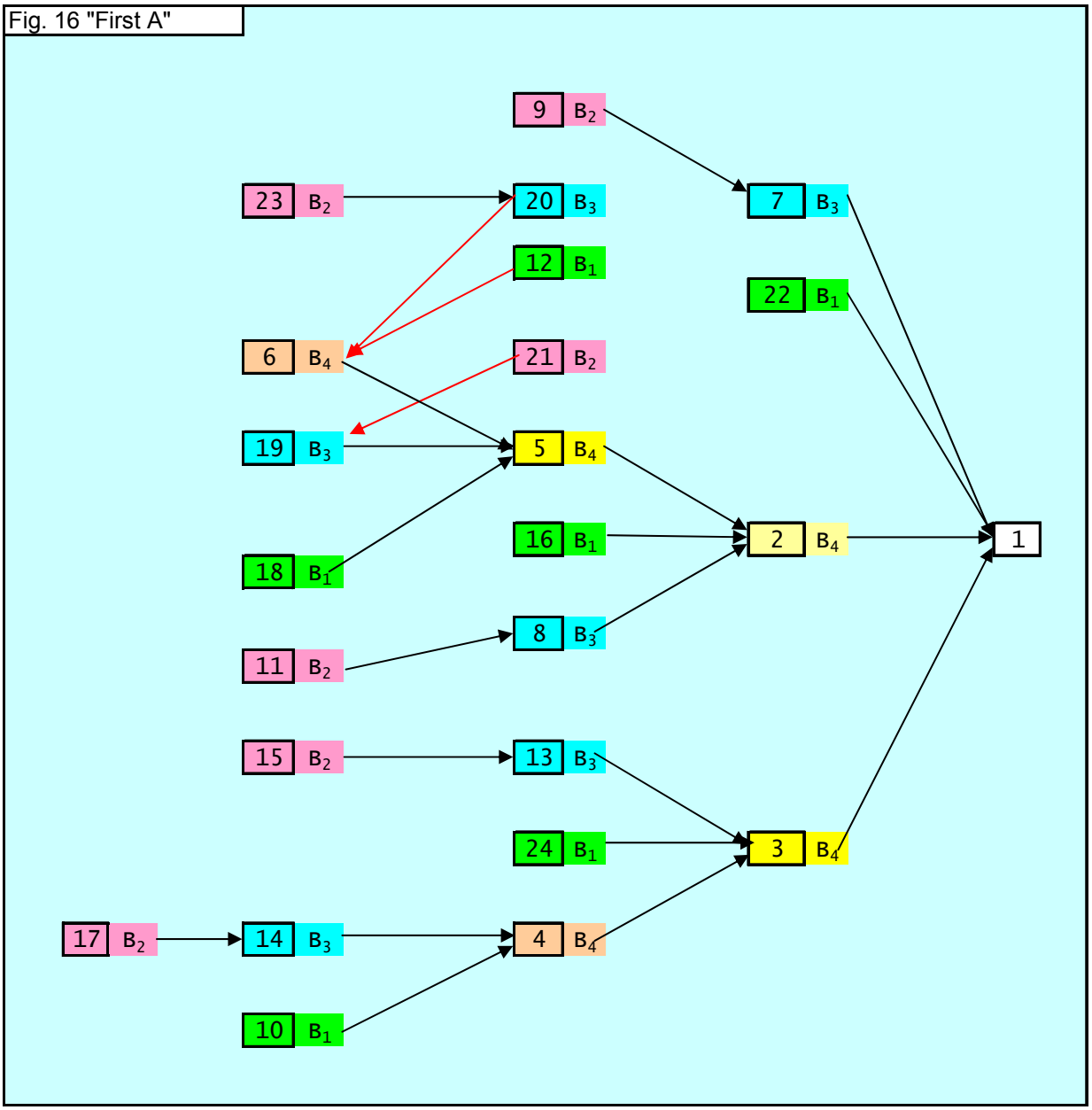
This distortion is clearly illustrated in Fig. 16 that represents the graph of the distances to building blocks B_i. By comparing Fig. 16 to Fig.12 the distortions can be seen very clearly, in the form of paths along which – in a part of the route – one moves away from the goal while believing to getting closer to it.
Consequently, the shortest path in the representation based on building blocks B_i coincides with the shortest path only in a part of the configurations - and when the two paths do not coincide, as it happens for nodes 12, 20 and 21 -, the representation "First A" provides sub-optimal paths .[4]

---

[3] with only one exception, due to B_1 and B_3 being a partition of layer S_1, at distance 1, which is imperfect: not all configurations of B_2 = ##A# are in fact solved by moving A anti-clockwise: there exists a configuration - number 21 - where the optimal move is clockwise. Hence the "First A" decomposition pattern is optimal to B_4, with the exception of only one configuration.

[4] For example: starting from configuration 20, DACB, the only optimal move is Left. Hence, 20→6→5→2→1. Let us suppose that DCBA and ABCD are the starting and final configurations, respectively. The optimal sequence - which can be calculated with the matrix of optimal links – is DCBA→Right→DACB→Left→BACD→Up→ABCD, while the rules in our procedure provide the following sequence: DCBA→Right→DACB→Left→BACD→Up→ADCB→ Down→ADBC→Right→ ABDC→Down→ABCD which obviously includes more moves.

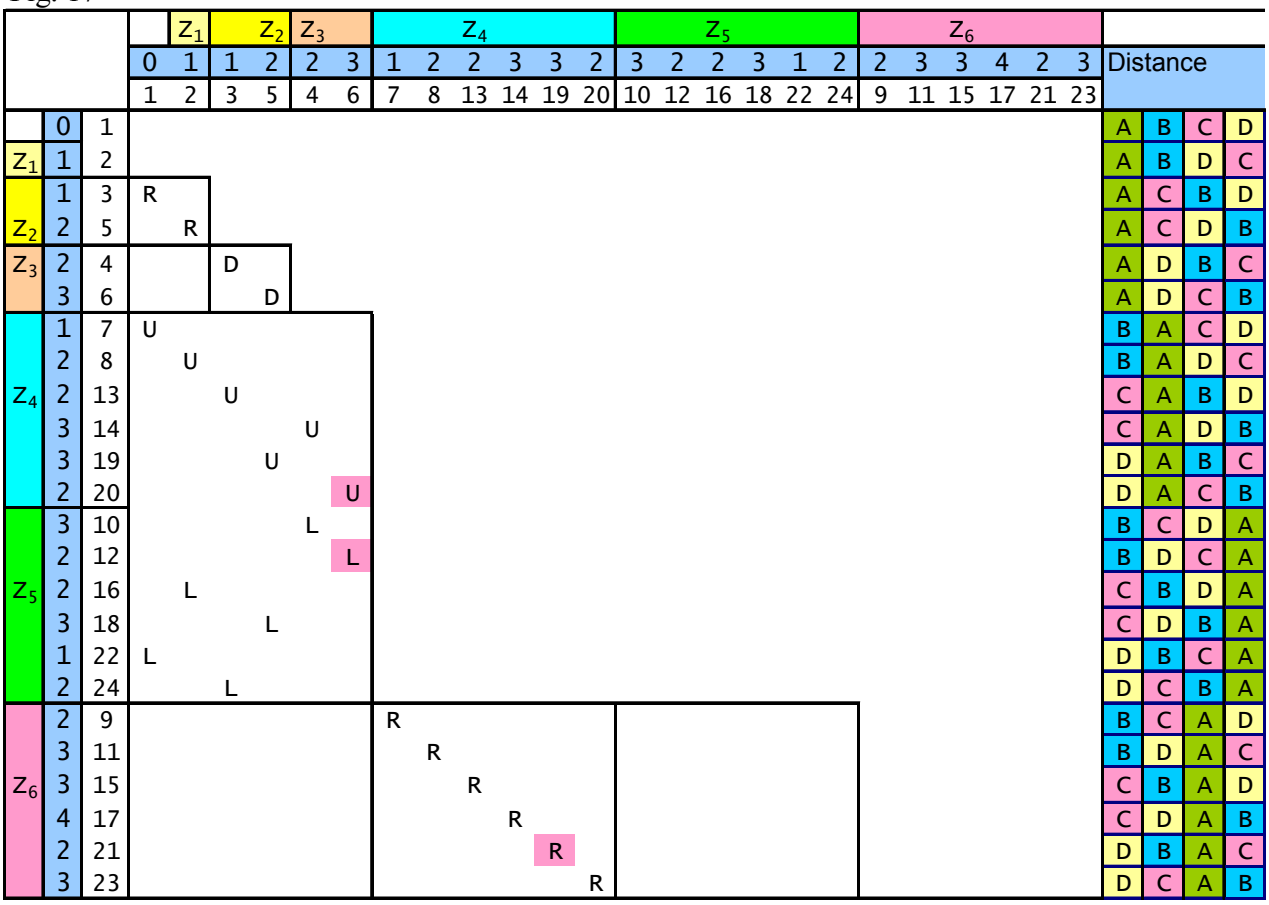
7



In this way the source of errors is explained: they are generated by the categorization players use to simplify the representation of the game and create the building blocks; the categorization - being founded on salient blocks, i.e. one tile at a time - is generally *simple* though *imperfect:* it groups "inhomogeneous" blocks since the system of distances among building blocks is not isomorphic with optimal distances.

To cross check what we have observed, a comparison can be made between the optimal strategy and the "First A" strategy expressed in the form of the matrix of shortest paths: three "discrepancies" are immediately apparent (Fig.17) , that is three specific configurations for which the "First A" strategy does not prescribes an optimal move: The configurations n. 12, 20 and 21 , shaded in fig 17

Fig. 17



The configuration n.20 is in fact at distance 3 to the goal, (the distances to the goal are reported in the third row of fig.23a) and the move U leads to the configuration n. 6 which is at distance 4 to the goal (G=1); therefore this move gets far the player from the goal. The same holds for the two other critical configurations.

In the right part of fig 17 the configurations of the game are listed in the extended format. It is evident that the sets of indexes Z_k make possible to classify the configurations in 6 different categories: ###A, ##A#, #A## , A#B#,A##B, AB##. This allow to represent the "First A" strategy in a remarkably simpler way than the optimal strategy, despite some distortions of the metric.

Figure 17 illustrates the aggregation based on the 6 categories, that is consistent with the symbolic content of configurations: the states characterized by the same optimal action are aggregated according to their common and salient features defined by the categories.

Fig. 18

ANTI-CLOCKWISE

"First A" strategy						Defined conditions
Conditions					Action	
#	#	#	A	1	Left	10,12,16,18,22,24
#	#	A	#	2	Right	9,11, 15,17,21,23
#	A	#	#	3	Up	7,8 13, 14, 19, 20
A	#	#	B	4	Down	4,6
A	#	B	#	5	Right	3 ,5
A	B	D	C	6	Down	2

Optimal strategy						Defined conditions
Conditions						
#	#	#	A	1	Left	10, 16,18,22,24
#	#	A	#	2	Right	9,11, 15,17,23
#	A	#	#	3	Up	7,8 13, 14, 19
A	#	#	B	4	Down	4,6
A	#	B	#	5	Right	3 ,5
A	B	D	C	6	Down	2
B	D	C	A	1*	Up	12
D	B	A	C	2*	Down	21
D	A	C	B	3*	Left	20

The criterion used by the player in "First A" to build the categories is that of aggregating all the configurations where the tiles A (then A and B and finally A,B and C) are in the same position. This criterion does *not* allow to cluster configurations at the same distance to the goal and therefore the signal of the distance to the goal that the player obtains while observing the position of A is *partially wrong*. The error is however limited to a few configurations, and is therefore very difficult to find.
For example, the category ###A is composed of states (10, 12, 16, 18, 20, 22, 24); by applying the move "left" to any configuration of this category, A is moved to position 1 and the player thinks to obtain a configuration that is nearer to the goal (###A→left→A###) while this is true for all the configurations except one: n. 12 (BDCA). The symbolic manipulation made on the states of the game is not therefore a safe criterion to identify the sets which are equidistant to the goal and as such this is a "natural" source of distortions.

4.4 Proprieties of the "First A" decomposition pattern

Sub-optimalities

As we have demonstrated, the "First A" procedure proves to be optimal in a sub-domain of configurations of the game. Therefore, when adopting this procedure, the players have the advantage to use a simple, abstract and complete representation but at the price of inefficiencies since the number of moves necessary to reach the goal is *on occasions* greater than the optimal value. The "First A" strategy is therefore described by a compact though sub-optimal list of instructions

Stability

This strategy has a remarkable advantage: the comparison between different alternatives to moving A first, then B and finally C in to the final position is immediate and is not very mind-consuming for the players. It is therefore quite natural to order the categories and find the optimal moves that connect them. Hence, the instructions of the strategy determine a locally stable optimum. The strategy is *locally* stable, in the sense that it cannot be improved by simply modifying the actions matching the elementary building blocks. In fact, if the instructions are modified, *with the constraints imposed by the structure of building blocks,* the programme turns out to be less efficient, i.e. the number of moves necessary to reach the goal increases. Therefore the solution based on the "First A" decomposition pattern – though being sub-optimal - cannot be improved, and is locally optimal and stable. In order to improve the strategy it will therefore be necessary to modify the building blocks.

4.5 An invariant decomposition pattern: the "First row" strategy

A different representation of the problem that some players have discovered during the different runs, is based on a more careful analysis of the properties of the game. Some players, instead of moving the tiles sequentially, as in the "First A" strategy, did take into consideration the interdependence among the various positions of the tiles, observing that moving a tile means moving at the same time a second tile associated to the action. The suggested strategy consists in taking into account the effect that an action on the tile has on tile B.
The strategy consists therefore in moving A and B to the final position, that is in the first row, in position AB ##, regardless of the positions of B and C, while the "First A" strategy suggests to move A regardless of B,C and D; in this case more comparisons are needed than in "First A" and hence more efforts in terms of memory and calculations: however, as we will see, the strategy designed in this way is optimal.

"First row" strategy

Let us take A and B and place them in the first row, in position AB ##, while comparing the number of moves needed to reach the goal whenever there are possible alternatives.
This comparison is generally easy. For example, let us consider configuration 10, B##A; it is easy to compare the two possible solutions: solution 1 consists in "moving A anti-clockwise" and it requires three moves. Solution 2 consists in "moving B to its final position and then move A": this requires two moves. The sequence of actions illustrated in Fig.18 is hence derived.
Here the categories are AB ##, A#B #, # AB #, ## AB #, BA #....., respectively; here, too, the problem consists in establishing an order in terms of proximity to the goal.

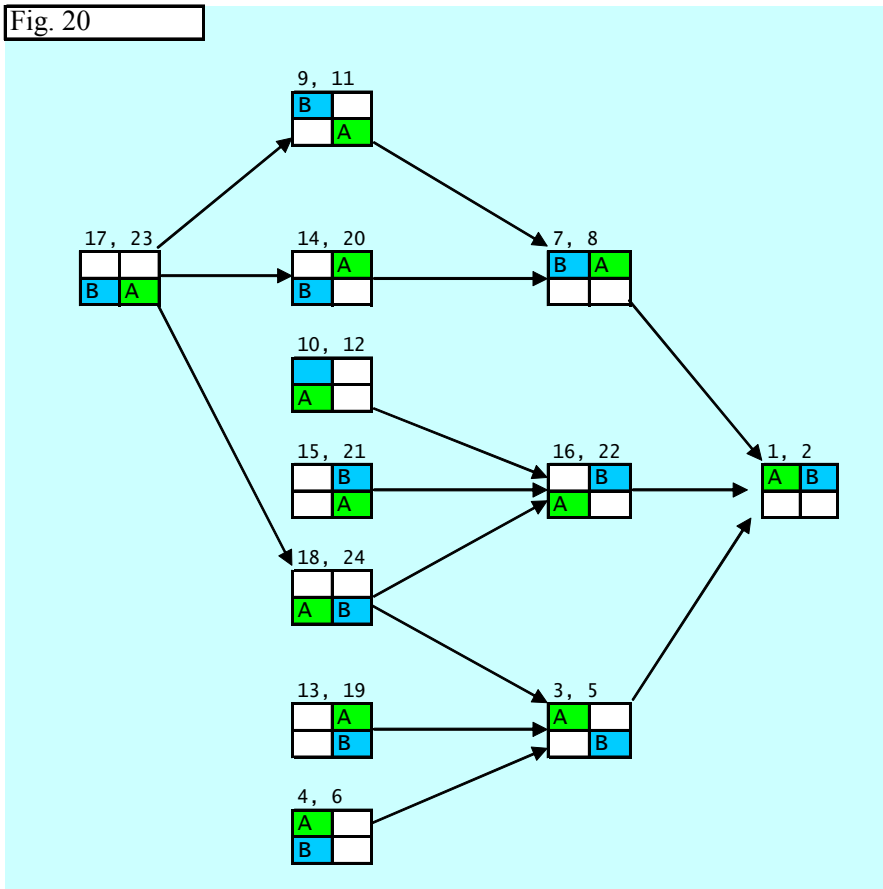
Fig. 19

States		1	2	3	4	Moves
1	2	A	B	#	#	
3	5	A	#	B	#	Right
4	6	A	#	#	B	Down
7	8	B	A	#	#	Up
9	11	B	#	A	#	Right
10	12	B	#	#	A	Up
13	19	#	A	B	#	Up
14	20	#	A	#	B	Left
15	21	#	B	A	#	Down
16	22	#	B	#	A	Left
17	23	#	#	A	B	Down, Left, Right
18	24	#	#	B	A	Left, Right

States		Moves	States		Moves
1,2	A B		13,19	A	Up
				B	
3,5	A	Right	14,20	A	Left
	B			B	
4,6	A	Down	15,21	B	Down
	B			A	
7,8	B A	Up	16,22	B	Left
				A	
9,11	B	Right	17,23		Down
	A			B A	Left,Right
10,12	B	Up	18,24		Left,Right
	A			A B	

The “First row” strategy, that simplifies the extended representation of the game, allows to maintain the metric. Based on what was demonstrated in par. 3.3, we note that this strategy identifies the building blocks that – while not fully fulfilling the conditions defined by the layers - , do generate an optimal strategy.

Fig. 20



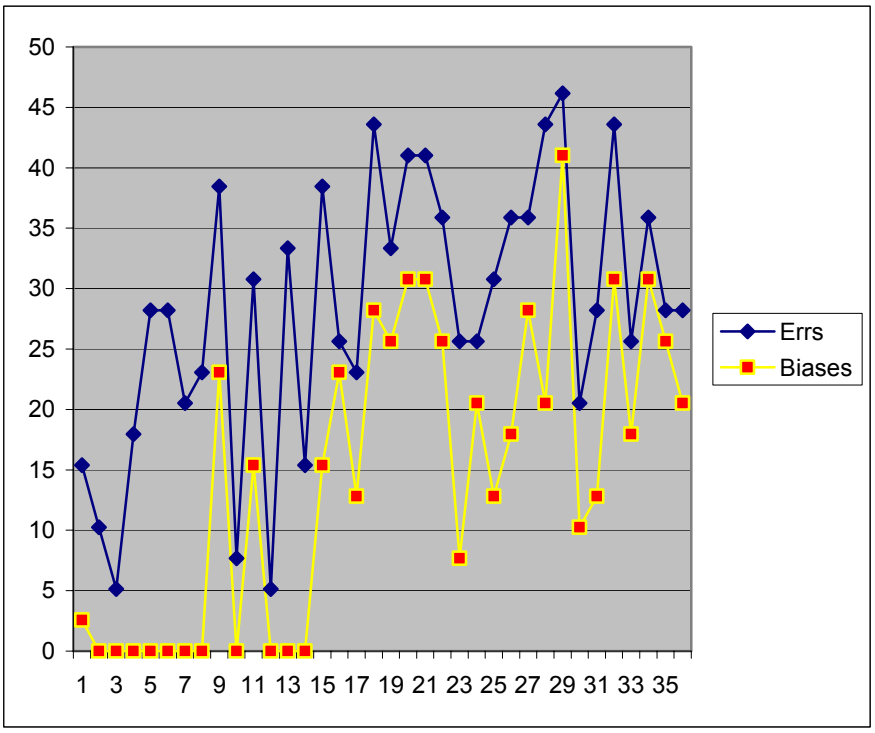
Interestingly, this decomposition pattern allows to compact the graph of the game so as to halve the number of states, which is equivalent to decomposing the graph into two perfectly identical ones. In this way we have identified building blocks that allow an invariant aggregation of the strategy.
The reader may well appreciate that besides the "First row" strategy – for which the graph was designed -, the "second row" and "first column" strategies are invariant too and allow some "savings" in the representation.

5 Preliminary experiments

An experiment on Minirubik, with 20 subjects, showed that a large number of players discovered the FirstA decomposition pattern, thereby making systematic errors in some configurations of the game. The errors were in line with the predictions (fig. 18 and fig.18) of FirstA strategy . Other players discovered the "First row" strategy, that allowed them not to make errors though "saving" in terms of memorization and calculation.

In Fig.21 the *global errors* are compared (i.e. discrepancies to the optimal strategy) to the *biases*, that is the deviations to the optimal strategy consistent with the "First A" strategy. It is clear that most "errors" are actions that are perfectly consistent with the "First A" strategy: the players have discovered the simplified representation of the game and they use it *rationally*; this is an interesting example of *bounded rationality*. In fact, the players behave in completely rational way within the FirstA decomposition pattern, which hiddenly makes them deviate from optimality.

Fig. 21



6 Concluding remarks

The decomposition of a problem allows players to successfully find the optimal strategies to the elementary sub-problems, but as we have seen, the decomposition patterns are usually non invariant and therefore the final result is not an optimal strategy. As we have seen, the processes of abstraction and categorization used by players to identify a pattern of decomposition in a puzzle use some salient features of the game as a guide, which do not usually reflect correctly the metric of the problem, i.e. the order among the categories in relation to the distance to the goal. Therefore the biases in decision making are originated by the nature of decomposition process, since the identified sub-problems do not reflect the metric of the problem correctly.

The errors are therefore *created by the representation* and can only be corrected if the representation - that is, the pattern of decomposition into sub-problems - is revised and modified appropriately. Moreover some decomposition patterns are invariant in limited sub-domains and it is therefore extremely difficult for an individual to notice errors and to correct them.

This explains the stability of the non invariant representations: in fact, correcting hidden errors should be extremely expensive in terms of calculation and memorization and therefore to correct the errors players are normally guided by exceptions that accidentally emerge. Any attempt to actively discover the errors would demand a complete and detailed description of the configurations of the game thus nullifying the "parsimony effect" obtained by the categorization, and nullifying the attempt to express the strategy in a simple manner.

We noted that the key element in the representation of a strategy is that of building blocks (that normally are represented as categories); given a puzzle, we know that there are different representations of it, each of them being defined by a different structure of building blocks. The wider

the extension of a building block, (the number of configurations involved), the lower the number of building blocks needed to represent the game and identify the winning strategy. If players, in order to get a clear and simple representation of the game, try to increase the extension of the building blocks, easily introduce hidden errors into the categorization. Generally the categories are created by the players during the runs of the game, on the basis of their direct experience. If they try to extend the rules that have experimented as optimal in a specific game context, to a larger domain, they may inadvertently include domains where the rules are suboptimal. Consequently, the errors in the mental representation of a problem can be the natural effect of the categorization and identification of building blocks beyond their "right" domain. This explains the "mechanization of thought" shown by Luchins and Luchins (1950) experiments.

Our approach explains the existence and stability of different and non optimal representations of the same problem. These alternative solutions can also be interpreted as different systems of expectations which are "rational to a limited extent", and may provide a platform for a formal approach to the treatment of boundedly rational expectations.

The discussion we have conducted so far emphasizes some important aspects of the approach of *bounded rationality*, on the one hand because suggest that the construction of a strategy is based on categorization guided by salience and simplification, on the other hand because shows that the categorization allows to simplify problems and at the same time generates biases and sub-optimalities.

This shows an intrinsic limit to rationality due to the trade-off between the simplicity of representation and the optimality of the strategy used to solve a problem.

References

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, pages 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.

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.

Egidi(2000) "Bias cognitivi nelle organizzazioni " in *Sistemi Intelligenti* Anno XII, n.2, Agosto, pp.237-270 Bologna: Il Mulino

Egidi, M. (2002) "Biases in organizational behavior" in Augier M. and March J. J. (editors) *The Economics of Choice, Change and Organization: Essays in Memory of Richard M. Cyert* Edward Elgar

Hey, J. (1991) *Experiments in Economics.* London, Basil Blackwell.

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

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

Kahneman, D., and Tversky, A. (1979) "Prospect theory: An analysis of decisions under risk". *Econometrica,* 1979, 47, 313-327.

Kahneman, D. and Tversky A. (2000), *Choices, Values and Frames*, Cambridge, Cambridge University Press.

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., and Simon, H.A. (1958). *Organizations*. New York, NY: Wiley.

Newell, A., Shaw, J.C., and Simon, H.A. (1958). "Chess-playing programs and the problem of complexity". *IBM Journal of Research and Development*, 2, 320-335.

Newell, A., and Simon, H.A. (1962). "Computer simulation of human thinking and problem solving" in M. Greenberger (Ed.), *Management and the computer of the future* (pp.94-133). New York, NY: Wiley.

Newell, A., Shaw, J.C., and Simon, H.A. (1962). "The processes of creative thinking" In H.E. Gruber, G. Terrell, and M. Wertheimer (Eds.), *Contemporary approaches to creative thinking* (pp. 63-119). New York, NY: Atherton Press.

Nillson N. J. (1986) *Problem solving methods in Artificial Intelligence* New York: McGraw Hill

Simon H.A. and Newell A. (1972), *Human Problem Solving,* Englewood Cliffs, Prentice-Hall

Simon H. A. (1979), "Rational Decision Making in Business Organization", *American Economic Review,* 69, 493-513