Interdependencies, nearly-decomposability and adaptation

Koen Frenken[1], Luigi Marengo and Marco Valente

Abstract

In this paper we discuss some limitations that selection mechanisms face when the entities subject to selection are complex systems of interdependent elements. We briefly present Kauffman's NK model which addresses this problem in biological systems. It is argued that, contrary to the myopic search behaviour, underlying biological fitness landscapes, social organisations are not bound in their search dynamics. This amounts to say that the problem of finding optima on a fitness landscape can be decomposed in many different ways. Following work by Page (1996), we present some measures of the complexity of a fitness landscape in terms of the complexity (size) of the algorithm that decomposes the problem most accurately, while still being able to locate the global optima with full certainty. We then extend this measures to allow for nearly-decomposability in a sense close to Simon (1969). Finally we study some evolutionary properties of populations of agents characterised by different decompositions of the same given problem.

[1] We thank Esben Andersen and the participants in the workshop "Agent-based and Population-based Modelling of Learning in Economics", Max-Planck-Institute, Jena, March 1998, for useful comments and suggestions. Koen Frenken gratefully acknowledges the financial support received from the European Commission (TMR-grant ERB4001GT961736).

1. Introduction.

The power of market selection forces in eliminating inefficiencies has long been a widespread but rather casual belief among economists. For instance, this belief has played the central role in the so-called *as-if* justification of the strong rationality assumptions of neoclassical economics (cf. Friedman 1953). This justification claims that no matter the actual decision algorithms employed by real decision makers, only those decision makers whose behaviour cannot be distinguished from the behaviour which would be followed by an optimising agent will eventually prevail under market selection. This argument has undergone various criticisms (e.g. Winter 1986), but some recent models of selection developed in biology pose some more fundamental questions on the validity of this rather casual faith in the power of selection mechanisms. Whenever selection does not operate on one-dimensional entities but on entities which have some complex[2] inner structure characterised by strong interdependencies among their constituent elements, selection is usually unable to eliminate inefficiencies of components which are tightly linked with others. Kauffman (1993) shows that as interdependencies grow, the selection surface (or "fitness landscape" in his terminology) tends to become more and more rugged, with many local optima. In this situation, evolutionary dynamics based on local ("marginal") search and selection will simply allow individuals to climb up to a local optimum near their initial position. The global optimum can be attained only by agents whose initial conditions are, by chance, within its basin of attraction, which in turn becomes smaller as interdependencies increase. The system shows strong properties of path-dependence (which local optimum is found depends on the starting position) and lock-in (local optima trap exploring agents preventing them from exploring any other point). Path-dependency and lock-in properties are stronger, the tighter selection forces are. Actually, only "weak" selection forces which spare individuals with decreasing fitness could eventually allow them to escape from bad local optima and climb higher ones.

This simple argument has an immediate appeal for organisational economists: business firms are undoubtedly complex systems of highly interrelated elements and Kauffman's model provides an appealing metaphor of organisational evolution and the limits of market selection. This line of research has been already pursued by some papers, e.g. Kauffman and McReady (1995) Westhoff *et al.* (1996), Levinthal (1997), Gavetti and Levinthal (1998).

However, the evolution of social organisations differs in important respect from the biological model of mutation and selection. The one which is the main concern in the present study is the possibility of social organisations of exploring their fitness landscape by applying search algorithms which are not necessarily limited to the one-gene-mutation algorithm which characterises biological systems. In other words, agents can virtually follow any kind of problem-solving strategies, and evaluate their efficiency and effectiveness. Following Simon's (1969) theory of problem-solving, we focus on strategies that decompose a problem in sub problems that are treated independently. Let us imagine that finding the global optimum of a given fitness landscape is the problem that a social organisation composed of N components (bits) has to solve, one-component (one-bit) mutation corresponds to a decomposition of such a problem into the smallest possible sub-problems. In fact, this corresponds to trying to optimise each component separately: this is potentially a very quick search strategy because if each component has h possible states, the search requires at most hN steps. Unfortunately,

[2] We ask the reader who - rightly so – gets rather nervous whenever he or she finds some casual use of the word "complexity", to bear a little patience: providing a rigorous definition of what we mean by complex structure is one of the main purposes of this paper.

Kauffman's model shows that if there are interdependencies among components, this maximum degree of decomposition does not generally allow the system to reach the optimal solution but only the local optimum which is closer to the initial position. At the other extreme, the problem could not be decomposed at all, that is the organisation could apply search algorithms which explore at the same time all N dimensions of the problem: this search strategy corresponds to mutating (at most) all N components and leads with certainty to the global optimum, but it does it only by examining all h^N possible configurations. In between there are many possible decompositions of the problem.

Following the methodology described by Page (1996), this paper provides some rigorous measures of complexity of a problem in terms of the size of the smallest algorithms which can locate the optimal solution starting from any initial condition. This provides a formal definition of the decomposability of a problem. This methodology is then extended to account for nearly decomposability: as suggested by Simon (1969), problem solvers are forced to decompose problems which are not fully decomposable, since their computational resources are limited. By separating interdependent elements into different sub-problems, a problem whose complexity is far beyond available computational resources is reduced to smaller sub-problems which can be handled, but it generally fails to provide the optimal solution. This paper provides a formal measure of nearly-decomposability: we show that if problems-solvers are ready to accept algorithms which lead to less than optimal ("satisficing") solutions they can decrease the size (and thus the execution time) of the algorithm required to find it. In other words, there is a trade-off between optimality and complexity.

This last point raises some interesting questions on the evolutionary properties of this kind of system: we simulate populations of agents characterised by different decompositions of the problem they face and we let them evolve in a very basic selection environment. In this way we can investigate whether, and under which conditions, agents with the "right" problem decomposition invade the population, or whether agents employing search algorithms based on sub-optimal decompositions. The latter type of agents have a high probability of being locked-in into local optima but they locate such optima relatively "quickly". Sub-optimal decompositions, though not effective in finding the optimal solution, may well be most efficient when one discounts for search time.

The paper[3] is organised as follows. The next section briefly reviews the basic features of Kauffman's NK model of fitness landscapes. Section 3 introduces two complexity measures, cover and Ascent Size, based on the decomposability of the search problem on a landscape. In section 4 we extend the measures for nearly-decomposability. A simulation model in section 5 explores some evolutionary properties of populations of agents applying different decompositions of a given problem.

[3] In this paper we treat the subject in a very abstract manner, without any direct reference to concrete problems. Two companion papers, Marengo (1998) and Frenken *et al.* (1998), present some applications respectively to a model of division of labour and organisational evolution and to the problem of technological design.

2. Kauffman's NK-model

The *NK*-model simulates the evolution of complex systems in which the elements function interdependently. Kauffman (1993) developed the *NK*-model primarily to simulate the evolution of populations of organisms that are described by a string of genes, but its formal structure allows for various applications in other domains.

A system is described by a string of loci which refers to the set of elements that make up the system. The variable N refers to the number of elements (genes in biology) ($i=1,...,N$). For each element, there exist A_i possible states (alleles in biology). The number of all possible strings among a system's element is called the possibility space of a system. The size of the possibility space S is given by:

$$S = A_1 \cdot A_2 \cdot .... \cdot A_N \qquad (1)$$

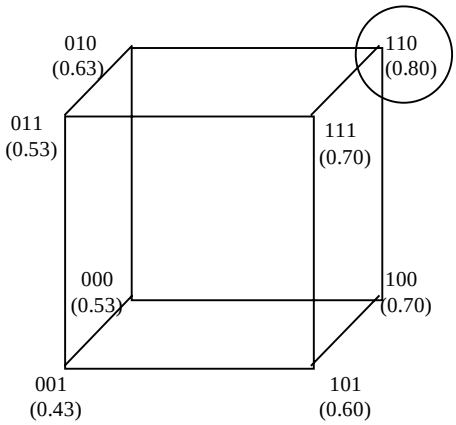
The K-value of a system refers to the number of "epistatic" relations among elements. The ensemble of epistatic relations describe the system's inner structure. The epistatic relations between elements imply that the contribution of one element to the overall fitness of the system is dependent both upon its own state and upon the state of K other elements. Two limit cases of complexity can be distinguished: *minimum complexity* when $K=0$, and *maximum complexity* when $K=N-1$.

Consider the example of a binary system consisting of three elements ($N=3$, $A_i=2$) in which the elements are independent ($K=0$). The fitness contribution of each element to the overall performance of the system is solely dependent upon its state (either "0" or "1"). Following Kauffman (1993), we draw the fitness values of elements for the two possible states randomly from an uniform distribution between 0.0 and 1.0. The fitness of the system as a whole F is calculated as the mean value of the fitness values of elements. *Figure 1* presents a detailed example.

The distribution of fitness values of all possible strings is called the *fitness landscape* of a system. By means of one-bit mutations, an agent may be able to improve its fitness as it moves through the fitness landscape. In the model, selection holds that when a new string is found that has a higher fitness, an agent will remain there. As long as there exists at least one neighbouring string that has a higher fitness than the fitness of the agent's current string, an agent can "climb" the fitness landscape by random one-bit mutations until it reaches an optimum. In this case of $K=0$, the fitness landscape always contains only one optimum (in our example string 110). For this reason, the global optimum is always found by a series of random one-bit mutations.

Figure 1: Fitness landscape of a N=3-system (K=0).

	f_1	f_2	f_3	F
000:	0.2	0.6	0.8	0.53
001:	0.2	0.6	0.5	0.43
010:	0.2	0.9	0.8	0.63
011:	0.2	0.9	0.5	0.53
100:	0.7	0.6	0.8	0.70
101:	0.7	0.6	0.5	0.60
110:	0.7	0.9	0.8	0.80
111:	0.7	0.9	0.5	0.70

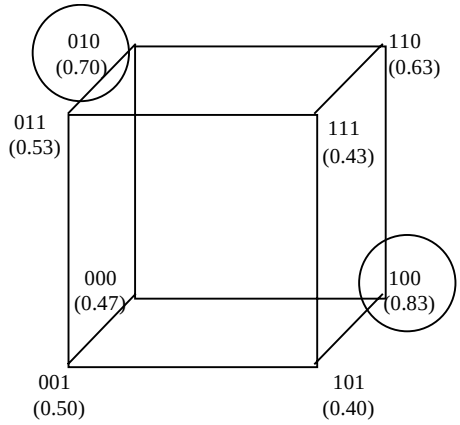


In the case of maximum complexity (*K=N-1*), the functioning of elements in a system depends upon all other elements. In this case, the fitness contribution of each element is dependent upon the states of all other elements. The value of the fitness contribution of each element has to be randomly drawn *for each string separately*. In other words, for each possible string of elements, the nature of their interdependencies is different. An example of a fitness landscape of a *N=K-1*-system is given in *Figure 2*. This fitness landscape contains two evolutionary optima: 100 and 010. String 100 is a global optimum and string 010 is a local optimum. Kauffman (1993) speaks of "rugged" landscapes containing several peaks (optima). Whether an agent ends up in either 100 or 111 depends upon its initial starting point: their search dynamics are strongly *path-dependent*. For both optima, it holds, an agent that occupies one of these strings, cannot improve the fitness of the system by one-bit mutations (we discuss the case of multi-bit mutations below). Once an optimum is found, an agent is *locked-in* forever. For this reason, contrary to *K=0*-systems, random one-bit mutations do not necessarily lead to the global optimum.

The local optima are *evolutionary attractors*: once an agent occupies a local optimum, it cannot escape this optimum by one-bit mutation. An important property of evolutionary attractors is the number of starting points that can end up in the attractor state (including the optimum itself). This number of strings is called the *basin of attraction* of the local optima. In the example, the basin of attraction of the global optimum covers only four strings, while the basin of attraction of the local optimum covers seven strings. For this reason, agents that move within this particular fitness landscape are more likely to become trapped in the local optimum rather than ending up in the global optimum.

Figure 2 : Fitness landscape of N=3-system (K=2).

	f_1	f_2	f_3	F
000:	0.6	0.3	0.5	0.47
001:	0.1	0.5	0.9	0.50
010:	0.7	0.9	0.5	0.70
011:	0.3	0.5	0.8	0.53
100:	0.9	0.9	0.7	0.83
101:	0.7	0.2	0.3	0.40
110:	0.6	0.7	0.6	0.63
111:	0.4	0.8	0.1	0.43



In between the two limit cases of minimum and maximum complexity, there are systems for which *K* is positive but smaller than its maximum value (*0<K<N-1*). For these systems, the fitness landscape usually contains several local optima, but less than in the case of maximum complexity. Kauffman (1993) analyses the structure of fitness landscapes for different K- and N-values by means of simulations to compare the evolutionary dynamics of systems with different degrees of complexity. We list five important results:

1. The number of local optima increases exponentially as the K-parameter is tuned from its minimum value *K=0* to its maximum value *K=N-1*.

2. The locations of local optima in the possibility space are *correlated* for low K-values (*0<K<5*). Put another way, the local optima of low-K systems have many states of elements in common.
3. The fitness values of local optima of low-K systems are highest on average. Thus, the interdependencies between a system's elements is strongest if the number of epistatic relations is small (but positive).
4. The basins of attraction of local optima of low-K systems are larger on average, and, generally, the higher their fitness, the larger their basin of attraction. As a consequence, a population tends to evolve towards strings of local optima with a high fitness.
5. When the number of elements *N* increases over time through the addition of new elements, the fitness of local optima of low-K systems tends to increase more than the fitness of high-K systems (Altenberg 1995). For this reason, one expects only systems with a low K-value to be able to expand in size *N*.

On the basis of these strong results, Kauffman (1993) suggested that systems with low K-values enjoy at least two evolutionary advantages over systems with high K-values. Not only, as mentioned above, do their local optima possess, on average, higher fitness values, but more importantly, the correlated structure of the fitness landscape and the large basins of attraction of "good" local optima make individuals more likely attain high fitness levels in low K systems. For this reason, systems with a low K-value are expected to outperform systems with a high K-value. Evolution then, will self-organise the composition of populations into one that is dominated by systems with low K-values.

In the context of human problem solving, the underlying biological assumption of random one-bit mutation is of limited relevance. The nature of search strategies in artificial evolution differs from the blind mutations in biological evolution. There exist no *a priori* constraints for human agents to proceed along a *multi-bit* mutation strategy. This search strategy is of importance since it may allow agents to escape strings that are the local optima with respect to one-bit mutations, and extend their search process towards better solutions.

Consider again the example of a fitness landscape for *K=2* in *figure 2*. The local optima are the strings which cannot be improved by one-bit mutation, and, as a consequence, agents may be trapped in sub-optimal solutions. However, a two-step mutation strategy allows an agent to escape these local optima by mutating the first bit from 0 to 1 and the second bit from 1 to 0, thus moving to the global optimum. In the example, irrespective of the initial conditions, agents with a two-bit mutation strategy will end up in the global optimum. Put another way, if we assume that agents follow a two-bit search strategy, then the global optimum has a basin of attraction equal to the size of the possibility space, and there is only one optimum present in the fitness landscape. It is clear that the number of local optima (i.e. the "ruggedness") of the landscape is not solely an objective feature of the fitness function as such, but depends also on the search strategy used.
The local optima are not given by the model, but are depending upon the behavioural assumptions underlying the model regarding the search behaviour of agents (as emphasised by Jones 1995, the topology of a fitness landscape is defined only as a function of a given search algorithm).

A powerful human device to solve complex problems is the construction of mental models which map the epistatic relations between elements. Using these representations, agents decompose the system in subsystems which are then solved independently (Simon 1969, 1973). If their models map approximately the epistatic relations between elements, the decomposition strategy allows them to economise considerably on search time. Thus, the complexity of a system is related to its degree of decomposability which in turn depends on the system's architecture. Complexity indicators therefore, should relate to the decomposability of the

specific system's architecture. The next section present two complexity indicators which reflect decomposability properties of a system.

3. Schemata and decomposability.

In this section we introduce alternative measures of the complexity of a search for the global maximum of a function defined on binary strings such as the *NK*-fitness landscape described in the previous paragraph[4]. The measures we propose here are based on a notion of complexity of a problem in terms of its decomposability into sub-problems which are constructed in such a way as to contain all and only interdependent elements. That is, the overall solution can be obtained by solving *independently* each sub-problem. Conversely each sub-problem is made of a set of interdependent elements which cannot be searched separately from all other elements of the same sub-problem.

This methodology has been first proposed by Page (1996), and it is based on the notion of schema (or, equivalently, hyperplane), introduced within Genetic Algorithms by Holland (1975). We first provide an informal description and then spell out the formal details.

In short, a schema is a subset of bits of a string, and is called *dominant* if by projecting on it all other strings we obtain a new string with higher (or equal) fitness.[5] Of course, only schemata which are part of the strings with maximum fitness can be globally dominant. By definition, a string with maximum fitness is itself a dominant schema (of maximum size): if we mutate all bits of whatever string into those of the global maximum, we obtain the global maximum itself. However, this is a schema of maximum dimension, and therefore, in order to find it with certainty, we have to test all the possible 2^N such schemata: if this was the only dominant schema, it would not allow for any reduction of dimension (i.e. decomposition) of the problem.

Imagine instead, at the other extreme, that all N schemata of size 1 are dominant. This is the case when all schemata have only one defining bit. This implies that we can tune each bit at a time. In that case one is sure that, if for instance mutating a 0 into a 1 in the *i*-th position increases the fitness of a string, then the optimal string has a 1 in the *i*-th position. Thus, we can decompose the problem into N problems of size one and we are guaranteed to find the global optimum in at most N steps by using N one-bit mutations.

A *cover* is a set of dominant schemata such that the union of all their defining bits returns all the N bits. If we construct the cover with minimum size schemata, this will give us the decomposition of the problem into minimum size sub-problems. This means that we can optimise each subset of bits separately, defined by each of such schemata. The global optimum will be simply the union of such sub-strings.

The *Cover Size* of a function defined on binary strings is exactly the size of the largest schema contained in the minimum size cover. It represents a measure of complexity of the problem in terms of the size of the largest of the smallest elements in which it can be decomposed. A cover gives a "perfect" decomposition, in the sense that each subset of bits

[4] As noted above, for complexity of the search we refer to the number of local optima, or ruggedness, that account for a complexity of a landscape. This depends on both the fitness function ("nature") *and* on the search strategy applied.

[5] The projection of a string on a schema will be defined below, but it can be easily understood by means of an example: the projection of the string 10001 on the schema 0##11 gives as a result the string 00011.

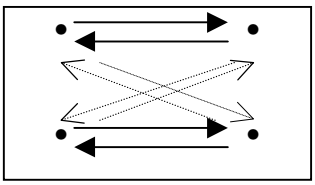
defined by a schema in a cover can be optimised independently from the others and only after solving each sub-problem can the pieces be put together. Thus the problem can be solved *in parallel*, and the largest sub-problem gives the upper bound to the complexity of the system, i.e. the time required to find its optimal solution.

Cover Size might over-estimate the complexity of the solution for a search algorithm because, if some schemata which form the minimum cover have bits in common, solving one of them reduces the size of those with overlapping elements. Of course, the bits which are common to two or more schemata in the cover must take the same value at the optimum, thus we can search for such values in the smallest of the overlapping schemata and then reduce the dimension of the larger ones. This is the idea behind the second complexity measure, which is called *Ascent Size* of the cover and gives the minimum number of bits we have to mutate at the same time in order to be sure to find - sooner or later - the maximum of the function.

Cover Size and Ascent Size of a system give us the number of bits which have to be mutated in order to be able to climb the function up to its optimum either by parallel or sequential problem-solving. It is clear that this represents a generalisation of Kauffman's (1993) one-bit mutation walks on *NK* fitness landscapes as discussed above. It tells us which kinds of algorithms are appropriate for which kind of fitness landscape. In particular, it turns out from our simulations which are listed below that except in the limit cases *K=0* and *K=N-1*, the K-value is not a good indicator of the complexity of the landscape. We will show that both Cover Size and Ascent Size are significantly higher than the K-value suggests: for instance 100 randomly generated fitness landscapes with *N=12* and *K=4* have an average Cover Size of 11.89 and average Ascent Size of 11.60. Thus, they are of almost the same complexity as a totally uncorrelated landscape with *K=N-1*. Only for the limit cases *K=0* and *K=N-1* we have that both cover and Ascent Size are, respectively, 1 and N.

An important property of the Cover Size and Ascent size is that they require *perfect* decomposability: we want schemata of a cover to contain *all* the bits which interact. This is a very demanding requirement. Usually we are looking for *nearly-decomposability* (Simon 1969, 1973): we want to isolate only the elements whose interactions bear a large influence on the performance of the system, and ignore the elements whose interactions do not have a large effect on the performance. Thus, elements with weak interactions are treated as if they were independent elements. By doing so, one may be able to decompose the function in more detail. In *figure 3* a schematic representation of a nearly-decomposable system (*N=4*, *K=2*) is given, where the dotted lines stand for weak structural relations.

Figure 3: Example of a nearly-decomposable system (N=4, K=2)



In the following, we show how one can build covers of ε-dominant schemata, where ε is the degree of "satisficing", i.e. the deviation from optimality we are ready to accept due to the neglect of weak interactions. As we increase the ε-parameter, both cover and Ascent Size

sharply decrease. In this way, we can rigorously measure the trade-off between precision and complexity of a search algorithm.

We now turn to a formal definition of the notions discussed so far:

S is the **set of all binary strings** of size n (in this section we use n for *N*):

$$s \in S;\ s = s_1 s_2 ... s_n \qquad s_i \in \{0,1\} \qquad |S| = 2^n$$

We call I={1,2,.....,n} the corresponding **set of indexes**.

Be F: $S \rightarrow \Re$ a real valued (fitness) **function** that we want to maximise. We require that F be defined on the entire set S. For simplicity we also require for the time being that F have a *unique* global optimum (this assumption will be dropped in a following section).

A **schema** (or hyperplane) h is a ternary string of size n:

$$h = h_1 h_2 ... h_n \quad h_i \in \{0,1,\#\}$$

The **defining bits of a schema** are those bits which differ from #. Let d(h) be the set of such defining bits:

$$d(h) = \{i;\ h_i \neq \#\}$$

The **size of a schema** is the number of its defining bits:

$$sz(h) = |d(h)|$$

As in Holland (1975) we say that a string s belongs to the schema h iff:

$$s_i = h_i \qquad \forall i \in d(h)$$

To every schema h belong $2^{n-sz(h)}$ strings.

The **projection** $\wedge$ of a string s on a schema h is a new string z which has the same bits as the schema's defining bits and the string's bits in the positions where the schema has a #:

$$s \wedge h = z \qquad \text{where} \quad z_i = h_i \ \text{ iff } i \in d(h)$$
$$z_i = s_i \text{ otherwise}$$

A schema h is **dominant** for the function F iff:

$$F(s \wedge h) \geq F(s) \qquad \forall s \in S$$

A **cover** is a decomposition scheme composed only by dominant schemata. Formally, a set of schemata $C=\{h^1,h^2,....h^k\}$ is a **cover** for the function F iff:

a) every h^i is dominant for every $s \in S$

b) $\bigcup_{i=1}^{k} d(h^i) = I$

Note that it is not necessary that the defining bits of the schemata forming a cover partition the set I (i.e. schemata may overlap in their defining bits, and, as we will see, this might actually reduce the complexity of the problem).

The **size of a cover** C is the size of the largest schema in it:

$$sz(C) = \max_i \{sz(h^i)\}$$

For instance, imagine that a function defined on the set of 5 bit strings has the following cover:

C = {1####, ###10, #10#0}

then we can be sure that, by allowing 3 mutations at a time, we are sure to reach the global optimum.

The main question turns out to be that of finding the *cover of minimum size*, because this gives us the finest possible decomposition of the function into sub-problems which still ensures that the global maximum can be reached from every starting point. This problem can be easily solved recursively according to the following algorithm:

0. be *s** the string with maximum fitness

1. *sz* := 1

2. find all the schemata of size *sz* containing *s**

3. for each such schema check if it is globally dominant, if so put it in the set *C*
4. if the union of defining bits of the schemata in *C* gives the set of all n bits then STOP
5. *sz* := *sz*+1
6. return to step **2.**

In this way we obtain the set C, i.e. the cover of minimum size. We are always sure that the algorithm will halt because at worst, by definition, *s** is a dominant schema which, alone, forms a cover. Note also that in general there exist multiple covers of a function, but using this procedure we are sure, by construction, to find all those of minimum size.

Cover Size is a measure of the finest possible decomposition of a problem into sub-problems which can be solved in parallel. But it can overestimate the complexity of the problem for a climbing algorithm which works sequentially[6]: in fact, if the cover contains overlapping schemata, an algorithm can start locating lower dimension schemata and in this way reduce the dimension of larger ones. Consider, as an example, that a problem has been decomposed in the following cover:
C = {1####, 11###, 110##, 1101#, 11010}
The Cover Size of this function is 5, thus, if we refer only to this measure, we conclude that the function has maximum complexity. But, actually, if we explore the function with a one bit mutation algorithm we are able to locate the first schema (of size one). Once we locate such a schema we are able also to locate the second - still with a one bit mutation algorithm - and so on. All in all, a one bit mutation algorithm is able to find the global optimum of the function, although its Cover Size equals n. We express this by saying that the *Ascent Size* of the cover is 1, implying that all schemata, also those of high dimension, can be located by a one bit mutation algorithm by using those bits which are in common with lower dimension schemata as intermediate steps. Of course, in order to exploit such a reduction of complexity we have to solve the problem sequentially, as higher dimension schemata have to be located only after the lower dimension ones have already been discovered.

Let C be the minimum size cover of a function, the corresponding Ascent Size can be constructed by processing the schemata belonging to C in the following steps:

0. be I={1,2,.....,n} the set of indexes
1. *sz* := 1, I'=I
2. find all the schemata of size equal to or smaller than *sz* over the set I'
3. if no such schemata then *sz* := *sz* + 1 and go to 2
4. cancel from I' all the indexes corresponding to the defining bits of such schemata
5. if I' is empty then STOP
6. else go to 2.

The value *sz* computed by such an algorithm corresponds to the Ascent Size of the cover of the given function.

4. Satisficing and nearly-decomposability.

[6] We have here a more formal appreciation of the intuition that working sequentially rather than in parallel can sometimes be advantageous if the sub-problems are nested or at least partly overlapping. Then, having already solved one sub-problem may reduce the complexity of the others.

When building a cover for a function we look for perfect decomposability, in the sense that we require that all schemata in the cover be dominant on all strings of the domain. In this way, we are guaranteed to decompose the problem into perfectly isolated components (in the sense that each of them can be solved independently). We can soften the requirement of perfect decomposability into one of *nearly-decomposability* by using a "satisficing" rule (cf. Simon 1969). In this case, we do not want the problem to be decomposed into completely separated sub-problems, i.e. sub-problems which fully contain all interdependencies in a system, but we want sub-problems to contain only the most "fitness-relevant" interdependencies while less relevant ones can persist across sub-problems. Put in another way, a satisficing rule aims at decreasing the effects of interdependencies in the sense that the structural relations that only weakly influence the performance of the system, are ignored. In this way, optimising each sub-problem independently will not, in general, give the global optimum, but a solution which lies within some error level ε from the global optimum itself, where $\varepsilon>0$ is a therefore a measure of the degree of satisficing.

In this way, we can construct an ε-Cover of the function and the corresponding Cover Size and Ascent Size measures. As ε increases the size of the schemata of the ε-Cover decreases. Thus, we have a trade-off between precision, complexity and time in finding a solution (remember that every unitary reduction in the size of the decomposition blocks halves the solution time if the problem is solved in parallel). Of course a cover based on strict dominance, or a 0-Cover in this terminology, is a special case.

Let us spell out the details of this method. So far, in order to construct the minimum size cover of the function, we have looked for schemata which are dominant with respect to every string in the domain of the function. This implies that by applying the schemata of a cover we follow a fitness-increasing path that will inevitably lead to the (unique) global optimum. Suppose now that instead of the global optimum we want to be sure to reach a set of "satisficing" solutions, which we define in the following way:

The set of **ε-satisficing solutions** is the set of strings whose value is at most ε lower than the global optimum:

$\Sigma = \{s \in S;\ F(s) \geq (1-\varepsilon)F(s^*)\}$

with $0 \leq \varepsilon \leq 1$ and s^* being the string corresponding to the global optimum.

In order to construct a cover which leads to such a set Σ we have to look for schemata which are not globally dominant, but are so only on strings which do not belong to Σ.

We can therefore give the following definition:

A set of schemata $C = \{h^1, h^2, \ldots h^k\}$ is a **ε-cover** for the function F iff:

a) every h^i is dominant for every $s \notin \Sigma$

b) $\bigcup_{i=1}^{k} d(h^i) = I$

Of course, if Σ contains only the global optimum we go back to the case previously analysed. Moreover we can now also treat functions with multiple global optima, if this is the case we just define Σ as the set of all global optima.

Note that in the previous case only schemata belonging the global optimum were possible candidates for inclusion in C, now instead all schemata belonging to the strings contained in Σ have to be tested for dominance.

The properties of ε-Covers are stated by the following propositions (cf. Marengo 1998 for the proofs):

Proposition 1: the cardinality of ε-Covers weakly increases as ε increases.

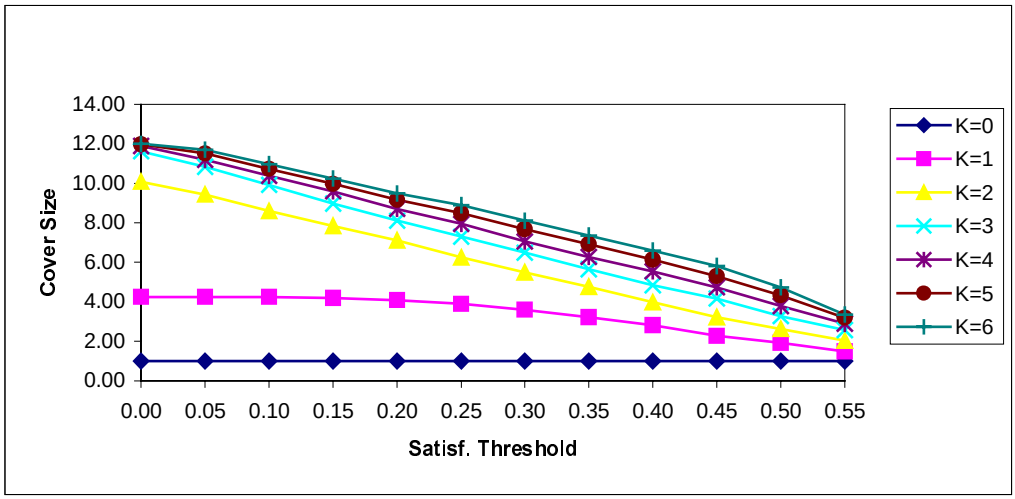
Proposition 2: the size of ε-Covers weakly decreases as ε increases.

Taken together these two propositions establish that as the satisfcing threshold ε increases the complexity of the satisficing solution (weakly) decreases.

In order to give an idea of the complexity reduction we can attain by accepting less than optimal solutions, we made some test on random fitness landscapes generated using the *NK*-model (Kauffman 1993, chapter 2) with bi-directional epistatic links.[7] For each value of *K*, we generated 100 different random landscapes and computed cover and Ascent Sizes for growing values of ε (the case ε=0 being the case of strict dominance and "perfect" decomposability). Figures 4 and 5 below present, respectively, the average values of Cover Size and Ascent Size over the 100 landscapes. Note that in these figures ε is measured in terms of relative deviation from the global optimum of the function: e.g. ε=0.2 means that we are ready to accept that our search algorithm halts at a local maximum which is at most 20% below the global maximum. From these results we can conclude that:

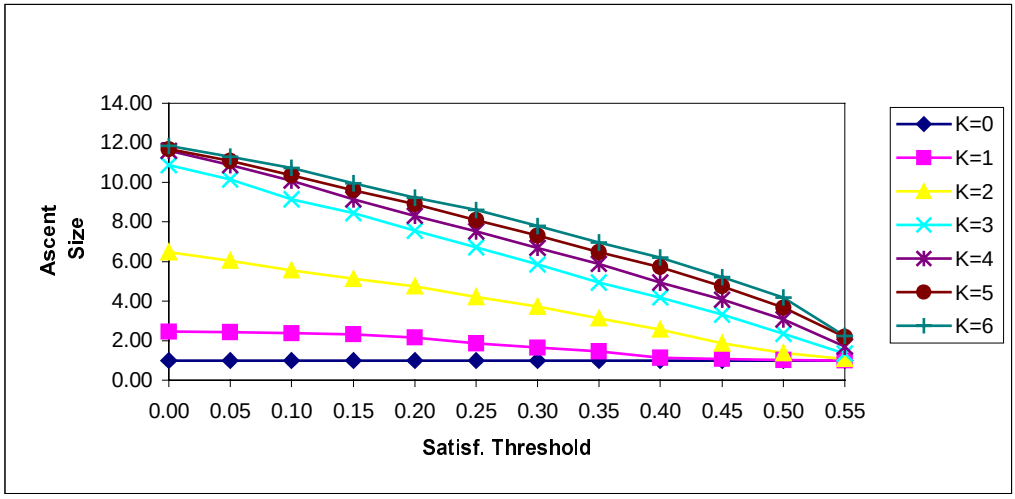
1. Cover Size ≥ Ascent Size ≥ (*K+1*), with equality holding only in the two extreme cases of *K=0* and *K=N-1*.
2. With ε=0, both cover and Ascent Size increase sharply as *K* gets bigger than 0: already with *K=3* the corresponding landscapes are hardly distinguishable - in terms of complexity measures - from completely random (i.e. *K=N-1*) ones.
3. as ε increases, i.e. when we lower the required performance of the algorithm, both cover and Ascent Size sharply decrease. Note that every reduction by 1 in these two measures corresponds to a decrease by half of the time required to complete the search. Therefore, in evolutionary terms, optimising strategies could be selected away by satisficing ones because the latter are much faster in reaching a "good" performance. This point will be developed in the next section.

Figure 4: Cover Size for various K and satisficing thresholds



[7] Originally, Kauffman's model is based on uni-directional links, but the use of bi-directional links does not change the qualitative properties of NK-systems with regard to their decomposition.

Figure 5: Ascent Size for various K and satisficing thresholds



5. Evolution of populations of problem-solvers.

Cover and Ascent Sizes set bounds to the complexity - in terms of decomposability - of search strategies within given selection environments. Of course, such measures of complexity are not known by agents who search adaptively the landscape with conjectural search strategies based on a given hypothesis on the decomposition of the search space. What are then the evolutionary properties of populations of agents (firms) which compete on the basis of search strategies based on conjectural decompositions?

In order to give some preliminary answers to this questions we simulated[8] a population of agents, where each agent is characterised by one out of a limited set of decomposition strategies. We let them compete in an environment defined by some simple rules of selection: worst scoring agents are removed from the population and replaced with "copies" of the best scoring agents. Copies are new agents that inherit the parent's decomposition scheme, but explore the landscape starting from a different – randomly assigned - point.

Agents start from a random point in the landscape and explore it according to their individual decomposition scheme. Each agent is defined by a given decomposition of the system into sub-systems and generates new points by choosing randomly one of the sub-systems, and mutating its bits (possibly all of them) randomly.

For instance, consider an agent which follows the decomposition strategy D={{1,2,3},{4,5,6,},{7,8,9},{10,11,12}}. Thus, the search space is decomposed into four subspaces of equal size. In order to generate a new point, the agent chooses randomly one of these 4 sub-spaces, then some (possibly all) of the bits in the chosen schema are mutated. The fitness value of the new point is observed: if such a fitness is higher than the one of its current position the agent moves to it, and the latter becomes the new starting point, otherwise the

[8] All simulations have been developed using the simulation platform called "LSD" (Laboratory for Simulation Development) (Valente, 1998) which provides a programming environment where simulations can be easily run also by inexperienced computer users. The interested reader can find downloadable code, user's manuals, more details of these and other simulations and programs to run simulations with different parameters and settings at the site: http://www.business.auc.dk/~mv/Lsd1.1/Intro.html

agent remains where it is. This step is iterated. Thus, what differentiates agents is only the way they generated new points to be tested, i.e. their search strategy, which in turn is determined by their decomposition of the search space.

Of course there is a huge number of possible decompositions, but in order to restrict such a number[9] we imagine that agents "know" that only some "well-behaved" decompositions are possible, in particular we imagine that schemata of an admissible decomposition must have all the same dimension and that they form a partition of the search space. We are thus left with only six possible decomposition and, correspondingly, six types of agents, named after the dimension of the sub-problems into which they decompose the problem:

Agent type 1: D={{1},{2},{3},{4},{5},{6},{7},{8},{9}{10},{11},{12}}
Agent type 2: D={{1,2},{3,4},{5,6},{7,8},{9,10},{11,12}}
Agent type 3: D={{1,2,3},{4,5,6},{7,8,9},{10,11,12}}
Agent type 4: D={{1,2,3,4},{5,6,7,8},{9,10,11,12}}
Agent type 6: D={{1,2,3,4,5,6},{7,8,9,10,11,12}}
Agent type 12: D={{1,2,3,4,5,6,7,8,9,10,11,12}}

We first checked whether agents whose decomposition perfectly reflects the structure of the underlying landscape are actually able to find its global optimum irrespectively of the initial conditions. To test this hypothesis we built five[10] kinds of random landscapes with a given structure, determined by the following covers (where + stands for either 1 or 0 and landscapes are named after the size of the corresponding cover):

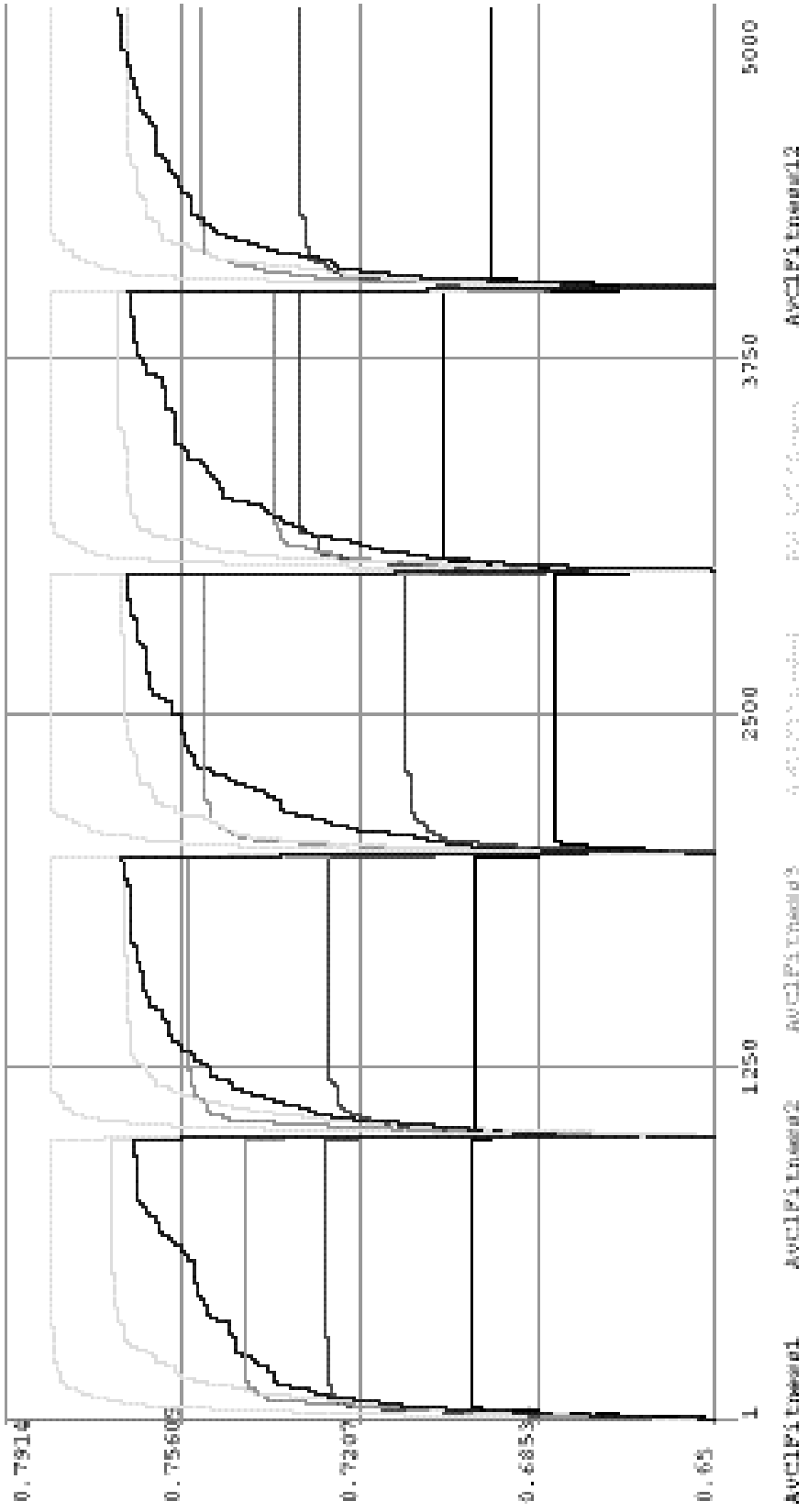
Landscape type 2:
{(++##########); (##++########); (####++######); (######++####); (########++##); (##########++)}
Landscape type 3:
{(+++#########); (###+++######); (######+++###); (#########+++)}
Landscape type 4:
{(++++########); (####++++####); (########++++)}
Landscape type 6:
{(++++++######); (######++++++)}
Landscape type 12:
{(++++++++++++)}

For each type of landscape we let 180 artificial agents, 30 for each type, evolve without any selection. Agents are initialized at a random point of the landscape, and then they pursue their search strategy as described above (i.e. they choose one sub-problem and mutate randomly at least one bit in it). To avoid the effect of lucky initial conditions (of course any agent can find the global optimum if it starts "close" enough to it) we periodically "shake" the population: when fitness values have settled, we reposition all existing agents in randomly chosen points, from which they have to start again their search.

In figure 6 we report the average fitness values for each class on a landscape of Cover Size 4 (we present the results obtained using one single landscape, though the same results have been reproduced with several different landscapes of the given Cover Size).

[9] As we want to study the evolutionary properties of optimal decompositions, we have to compare them with some "reasonable" ones, this is the reason why we restrict so much the number of possible decompositions.
[10] We omit landscapes of Cover Size one because they are trivial.

Figure 6: Average Fitness of Classes of Agents – Landscape type 4

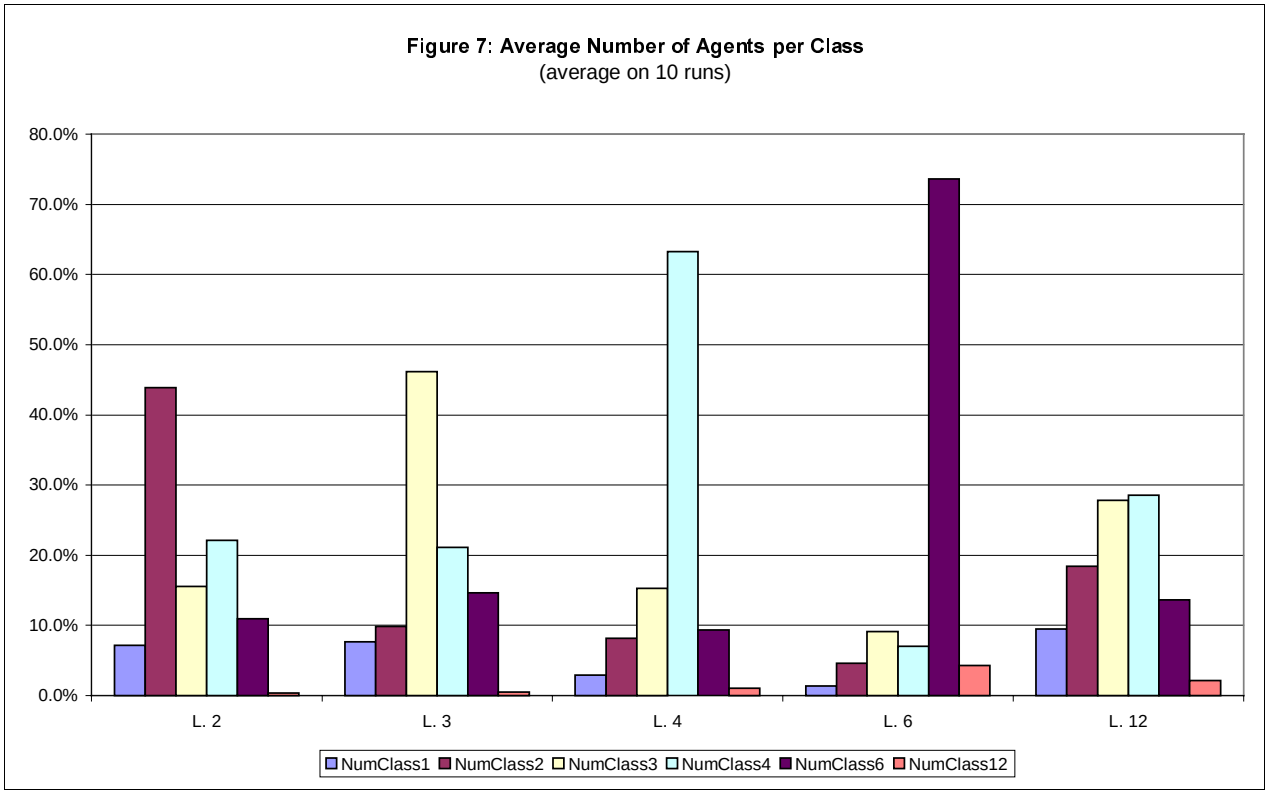


Average fitness values of each class are always non-decreasing (because agents either increase their fitness or stay put) except the sudden drops they incur every 1000 iterations, when agents are randomly repositioned. Note that the average fitness of agents of types 1, 2 and 3 quickly stops growing, because all agents in those classes become locked in local optima (some of them may actually reach the global optimum if their starting point is close enough, but this effect tends to be cancelled out by the periodic repositioning of agents). All agents in class 4 increase quickly their fitness and set on the global maximum. Agents in class 6 grow less quickly than agents in the lower classes but they cannot all reach the global optimum, since some of them are trapped in local optima. This is because at each mutation they can jump on a wider range, but their decomposition does not allow to identify with certainty the global maximum. Agents in class 12 use a purely random search, having the possibility to jump on every point at each mutation. This allows them to grow slowly but steadily, and in the end all agents in this class may also be able to find the global maximum. But this strategy is extremely slow, being clearly outperformed, in speed of convergence, by the one used by agents of class 4.

This result confirms that only classes 4 and classes 12 can always reach the optimum, but class 4 does it in much shorter time. Equivalent results have been found for landscapes of Cover Sizes 2, 3, 6 and 12. In all exercises we observe the same results: strategies using decompositions not including the one corresponding to the minimum size cover are bound to be trapped in local optima. Strategies using decompositions which include the minimum one but are bigger do always reach the global optimum but do so slowly. Search strategies corresponding to the minimum cover clearly outperform every other strategy.

But correct decomposition strategies might not always prevail in competitive environments with some form of selection, in fact they are able to always locate the global optimum with certainty, but the time required might be so long that they are actually eliminated by the selection mechanism. In order to test this proposition, we ran a set of simulations where agents competed in a very simple selection environment. That is, we used a similar set up as above (180 agents, 30 for each strategy initially located in points of the landscape randomly chosen) but are not re-positioned. Instead, every 10 mutations they are ranked according to their fitness. The 30 worst scoring agents are eliminated from the population and replaced by copies of 30 agents chosen among the other surviving agents.

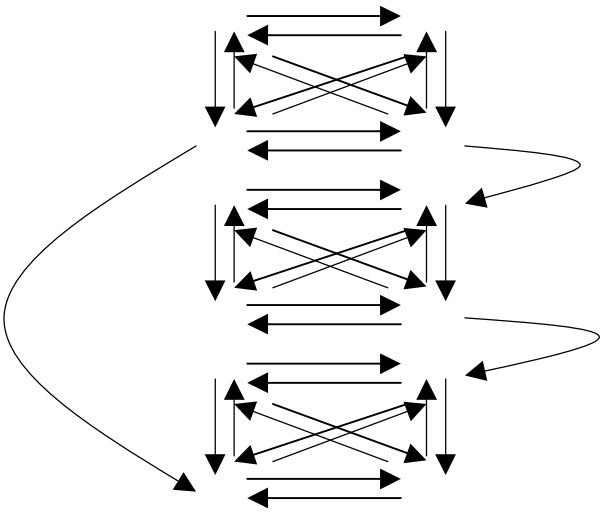
Figure 7 shows the average number of copies over 10 different simulation runs on each of the landscapes considered, to avoid possible influences of particular initial points.



It is possible to see that, as expected, for landscapes of sizes 2, 3, 4 and 6, agents whose decomposition perfectly reproduce the structure of the landscape tend to dominate the population. On the contrary, on landscapes of size 12 this does not happen, as agents of type 12, in spite of being the only ones always able to find the global optimum, are displaced by "simpler" strategies which tend be locked into local optima, but reach them relatively quickly.

The latter result is even stronger in "nearly-decomposable" or "modular" landscapes, where sub-optimal, "satisficing" search strategies tend to be evolutionary dominant. We constructed a nearly-decomposable system of size *N=12 (A=2)*, composed of 3 modules of 4 elements each. Elements within a module are all bilaterally linked to each-other, and one element of each module has a unilateral link to an element of the next one. Figure 8 provides a schematic representation of the structure of the system.

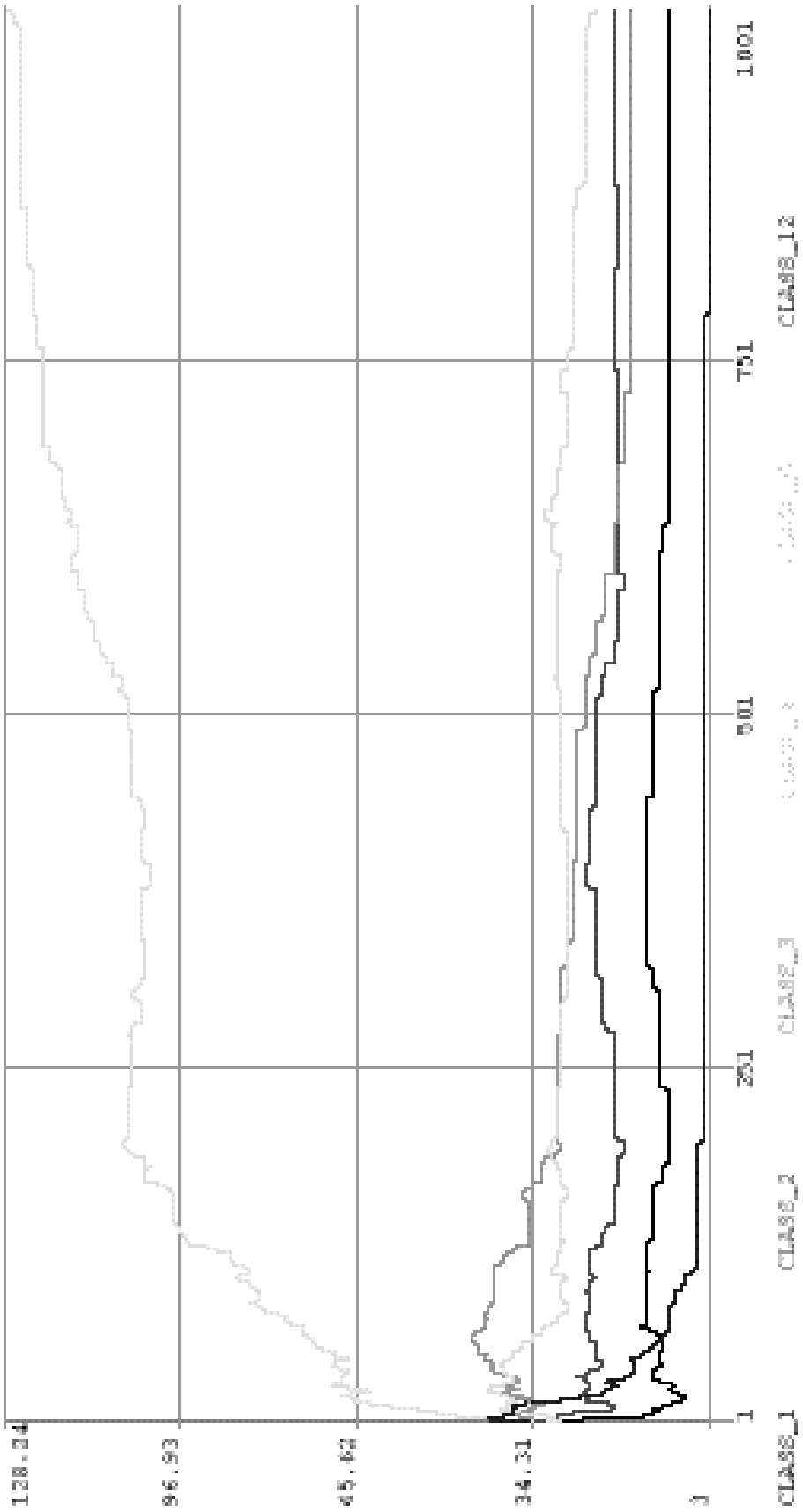
Figure 8: Example of a nearly-decomposable (or modular) system



The fitness landscape of such systems has a Cover Size of 12, thus the optimising decomposition strategy which secures the finding of the global is the 12-bit mutation strategy (which is a "decomposition" into one subsystem). However, the architecture of structural relations is highly pronounced: a decomposition strategy of size 4 is able to capture "most" of the interdependencies if the system. Therefore, it is interesting to test whether "satisficing strategies" adopting a decomposition strategy of type 4 outcompete optimising strategies which follow a 12-bit mutation strategy. As shown by figure 9, the answer is that indeed the optimising 12-bit mutation strategy is dominated by the 4-bit mutation decomposition strategy into three subsystems. It is worth pointing out that also agents using a 3-bit mutation strategy corresponding to a decomposition into four subsystems are also relative frequent in the population though their decomposition scheme is not the right one. This shows that even agents with a wrong decomposition scheme are able to outcompete the optimising 12-bit mutation strategy.

We conclude that in highly complex landscapes, optimising strategies following a search strategy corresponding to the "correct" structure of the system do not have an advantage over satisficing strategies. It is not particularly useful to base a strategy on the Cover Size since it is a very time-consuming search type. In an evolutionary environment, the decomposition of the system into sub-systems, even if it is only an approximate decomposition, allows agents to improve their fitness in very short time, even though they usually are not able to find the global optimum. Thus, satisficing strategies aiming at a high rate of improvement outcompete optimising strategies which aim at the maximum end result.

Figure 9: Number of agents per class in a nearly-decomposable landscape of size 12



References

Altenberg, L. (1995), Genome growth and the evolution of the genotype-phenotype map, in: Banzhaf, W., Eckman, F.H. (eds.), *Evolution and Biocomputation* (Berlin and Heidelberg: Springer-Verlag).

Frenken, K., Marengo, L., Valente, M. (1998), Modelling Decomposition Strategies in Complex Fitness Landscapes. Implications for the Economics of Technological Change, *Seventh Conference of the International Schumpeter Society*, Vienna, 13 –16 June 1998.

Friedman, M. (1953), *Essays in Positive Economics*, Chicago, University of Chicago Press.

Gavetti, G., Levinthal D. (1998), Looking Forward and Looking Backward: Cognitive and Experiential Search, *The Wharton School, University of Pennsylvania*, mimeo.

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

Jones, T.C. (1995), *Evolutionary Algorithms, Fitness Landscapes and Search*, PhD thesis, University of New Mexico, Albuquerque.

Kauffman, S.A. (1993), *The Origins of Order. Self-Organization and Selection in Evolution* (New York & Oxford: Oxford University Press).

Kauffman, S.A., MacReady (1995), Technological evolution and adaptive organizations, *Complexity*, vol. 1, pp. 26-43.

Levinthal, D. (1997), Adaptation on rugged landscapes, *Management Science*, vol. 43, pp. 934-950.

Marengo, L. (1998), Interdependencies and division of labour in problem-solving technologies, *Seventh Conference of the International Schumpeter Society*, Vienna, 13 –16 June 1998.

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

Simon, H.A. (1969), *The Sciences of the Artificial* (Cambidge MA. & London: MIT Press).

Simon, H.A. (1973), The organization of complex systems, in: H.H. Pattee (ed.) *Hierarchy Theory. The Challenge of Complex Systems* (New York: George Braziller).

Valente, M. (1998), Laboratory for simulation development – Lsd, http://www.business.auc.dk/~mv/Lsd1.1/Intro.html

Westhoff, F.H., Yarbrourgh, B.V., Yarbrourgh, R.M. (1996), Complexity, organization, and Stuart Kauffman's *The Origins of Order*, *Journal of Economic Behavior and Organization*, vol. 29, pp. 1-25.

Winter S.G. (1986), Adaptive Behaviour end Economic Rationality: Comments on Arrow and on Lucas, *Journal of Business*, vol. 59, pp. 5427-34.