J Evol Econ (2012) 22:935–956
DOI 10.1007/s00191-011-0240-6

REGULAR ARTICLE

Optimal modularity: a demonstration of the evolutionary advantage of modular architectures

Koen Frenken · Stefan Mendritzki

Received: 22 October 2009 / Accepted: 5 July 2011 / Published online: 2 August 2011
© The Author(s) 2011. This article is published with open access at Springerlink.com

Abstract Modularity is an important concept in evolutionary theorizing but lack of a consistent definition renders study difficult. Using the generalized NK-model of fitness landscapes, we differentiate modularity from decomposability. Modular and decomposable systems are both composed of subsystems, but in the former, these subsystems are connected via interface standards, while in the latter, subsystems are completely isolated. We derive the *optimal level of modularity*, which minimizes the time required to globally optimize a system, both for the case of two-layered systems and for the general case of multi-layered hierarchical systems containing modules within modules. This derivation supports the hypothesis of modularity as a mechanism to increase the speed of evolution. Our formal definition clarifies the concept of modularity and provides a framework and an analytical baseline for further research.

Keywords Modularity · Decomposability · Near-decomposability · Complexity · NK-model · Search · Hierarchy

JEL Classifications D20 · D83 · L23 · O31 · O32

1 Introduction

Simon's (1962) seminal work on complex systems emphasized the modular and hierarchical structure of most complex systems, both natural and artificial.

K. Frenken (✉) · S. Mendritzki
Eindhoven Centre for Innovation Studies (ECIS), School of Innovation Sciences,
Eindhoven University of Technology, Eindhoven, The Netherlands
e-mail: k.frenken@tue.nl

The modular nature of complex systems refers to the nearly decomposable architecture of the interaction between elements. In modular systems, the great majority of interactions occur within modules and only a few interactions occur between modules.

Modular architectures offer evolutionary advantages because, in most instances, the effect of a change in a given module is confined to that module. Due to this localization of the effects of changes, the probability of a successful change is greatly enhanced. Each module can be improved more or less independently of other modules. For example, modular technologies allow for innovation in each module without the risk of creating malfunctions in other modules. Similarly, modular organizational designs allow different departments to change their operating routines without creating problematic side effects in other departments. More generally, the typical feature of modular systems holds that they can more easily be improved by random mutation and natural selection than other complex systems.

The NK-model, originally developed by Kauffman (1993) and generalized by Altenberg (1994), is a common tool to analyze the evolutionary dynamics of complex systems, including organizations and technologies (Levinthal 1997). In the economics and management literatures, several simulation studies have been carried out to analyze the conditions under which modular systems favor adaptation compared to other complex systems (Frenken et al. 1999, Marengo et al. 2000; Ethiraj and Levinthal 2004; Dosi and Marengo 2005; Brusoni et al. 2007; Rivkin and Siggelkow 2007; Ciarli et al. 2008; Geisendorf 2010; McNerney et al. 2011; cf Bradshaw 1992; Baldwin and Clark 2000). These studies tend to confirm the central idea that modular systems are improved by random mutation and natural selection at a faster rate than other complex systems. Yet, the exact results of the simulation exercises differ across these studies, as they utilize different assumptions regarding search behaviour and memory constraints, as well as differing definitions of modularity.

In the following, we propose a formal definition of modularity that distinguishes it from decomposability. Though many use the terms decomposability and modularity interchangeably, we argue that modular systems differ from decomposable systems; while decomposability requires a full decomposition of a complex system into subsystems, modularity requires a system architecture in which subsystems are still connected via interface standards. Conceptually, the problem of the decomposability concept is that a decomposable system is no longer one system, but simply a collection of several smaller systems. As a representation of a technology, or an organization, it falls short in conceptualizing the fact that elements in a technology or organization always act together and are collectively subject to selection. The idea of a decomposable system is thus better understood as an analytical construct or as an approximation of reality rather than a precise representation of a real-world system. The concept of modularity overcomes these conceptual issues. A modular system cannot be partitioned into completely independent subsystems, but rather contains nearly independent subsystems (modules) which are connected via interfaces. These interfaces are elements of a system that connect subsystems

such that the only interdependencies between the subsystems are via the interface standards. This definition corresponds quite closely to the concept of near decomposability introduced by Simon (1962, 1969, 2002), as well the more recent notion of modularity in complex networks (Newman 2006).

The applied literature on modularity has drawn similar distinctions between modularity and decomposability. For example, Baldwin (2008) compares perfect modularity (similar to our definition of decomposability) with near decomposability (similar to our definition of modularity). Langlois and Garzarelli (2008, p.128) differentiate between decomposable systems and modular systems which are "nearly decomposable system that preserves the possibility of cooperation by adopting a common interface". This paper, then, is best seen not as creating a novel distinction, but as adopting an existing distinction and expressing it formally.

We will argue below, using a generalized NK framework developed by Altenberg (1994), that modular systems, defined in this way, can be optimized globally given the right sequence of problem-solving. Though a decomposition strategy is not feasible, modules can be optimized independently as long as interface standards between modules are left unchanged. This means that, contrary to decomposable systems, optimization of modular systems requires *hierarchical* problem-solving, where interface standards are defined first, followed by module design within the constraints of the standards.

Following this definition, we will proceed to derive the optimal level of modularity for systems of a given size, where the optimum is defined by the search time required for global optimization. This result is shown to be extendable to multi-layered hierarchical complex systems, where modules are defined recursively. We find this extension important since hierarchical complex systems are ubiquitous in technological artefacts and organizational design, yet have not been analyzed thus far in the NK-modelling framework.

The reader will note that the model we propose is quite simple. For example, it adopts a global search strategy. This is done purposively for the purpose of creating a framework and a baseline. The framework offers the possibility of comparability of results derived from different assumptions. The baseline of a simple model provides an anchor for comparison with more complex models. This approach of using a simplified model as a baseline is common in NK modelling. It should be noted, then, that the purpose of this model is not to make the empirical claim that it reflects actual behaviour. It rather should be interpreted as a tool to be useful in integrating and reconciling various models on modularity. The importance of creating common frameworks is discussed in terms of the ongoing debate as to whether over-modularity has evolutionary advantages.

2 Decomposability and modularity in a generalized NK-model

We define a system as consisting of N elements (n = 1,…,N). For each element, n, there exist A_n possible states. The number of possible system

designs that make up the system's *design space* (Bradshaw 1992) is given by the product of the number of possible states for each element:

$$S = \prod_{n}^{N} A_n \tag{1}$$

In the following, we will assume that $A_n = A$ for all n, which implies that the size of the design space equals A^N.

We assume that each pair of elements is either interdependent or not. Interdependence between a pair of elements means that, if a mutation is carried out in one element, the functioning of the other element is also affected. Decomposability means that a system can be partitioned into non-overlapping subsystems such that no interdependencies exist between subsystems. This implies that subsystems can be optimized independently and in parallel. The time required to globally optimize a system is then bounded by the size of the largest subsystem.

For example, consider a system with $N = 5$ and a binary design space ($A = 2$). The number of possible designs is $2^5 = 32$. If the functioning of all elements is dependent on the state of all other elements, global optimization requires exhaustive search: one has to evaluate the fitness of all 32 possible designs to determine which design has the highest fitness. Assuming one evaluation per time period, the search time is 32 periods. Now, consider the case in which the functioning of the first and second elements are interdependent, and the functioning of the third, fourth and fifth elements are interdependent. In this case, the subsystem containing the first and second elements can be optimized independently from the subsystem containing the third, fourth and fifth elements. Since search can proceed in parallel, the search time required to globally optimize the system is bounded by the size of the largest subsystem, in this case $2^3 = 8$ periods. The computational complexity of a system, as defined by the search time required to globally optimize a system, can then be expressed as a function of the number of elements of this largest subsystem (three in this example), also known as the cover size of a system (Page 1996).

2.1 Altenberg's generalized NK-model

To formally model modular systems, the original NK-model as developed by Kauffman (1993) has to be generalized to allow for interface standards. The distinguishing feature of modular systems is that some elements of the system (the interface standards) have no direct contribution to the system's fitness, but solely mediate the interdependencies between modules. However, in the original formulation of the NK-model by Kauffman, elements in a complex system by definition have a fitness value. As such, the Kauffman-type of NK-model is ill suited to deal with modular systems. The generalized NK-model developed by Altenberg (1994) allows a more general treatment in which elements are not required to have inherent fitness values, which allows the inclusion of mediating elements.

Altenberg's generalized NK-model describes a system by N elements ($n = 1, \ldots, N$) and F fitness elements ($f = 1, \ldots, F$). In biological systems, for which this generalized NK-model was conceived, an organism's N genes are the system's elements and an organism's F traits are the selection criteria. The string of genes is collectively referred to as an organism's genotype, while the set of traits is collectively referred to as an organism's phenotype. A single gene affects one or several traits in the phenotype, and a single trait is affected by one or several genes in the genotype. The vector of genes affecting a trait is called a polygeny vector, while the vector of traits affected by a gene is called a pleiotropy vector. The structure of epistatic relations between genes and traits is represented in a "genotype-phenotype map", which is represented by a matrix of size $F \cdot N$ with:

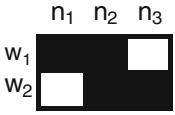
$$M = [m_{fn}], \quad f = 1, ..., F, \quad n = 1, ..., N \tag{2}$$

Analogously, a technology can be described in terms of its N elements and the F functions it performs *i.e.* the quality attributes taken into account by users (Frenken and Nuvolari 2004). The string of alleles of elements describes the "genotype" of a product, and the list of functions describes the "phenotype" of the product (e.g., speed, weight, efficiency, comfort, safety, etc). The genotype–phenotype map of a product is generally called a product's architecture (cf. Henderson and Clark 1990).

The original NK-model can now be understood as a special case of the generalized genotype–phenotype matrices. Three restrictive assumptions are operative in the original NK-model, namely N-F symmetry, N-F reflexivity, and polygeny symmetry. N-F symmetry is the condition that the number of functions F equals the number of elements N. This assumption is necessary in order to enforce N-F reflexivity, which is that each element (n_x) affects its counterpart function (f_x); in terms of the genotype–phenotype matrix, this implies that the diagonal is always characterized by presence of a relation between element and function. Polygeny symmetry is the requirement that each function is affected by the same number of elements. In the NK-model the polygeny of each function is assumed to be exactly $K + 1$, with pleiotropy of each element being determined randomly (with pleiotropy being on average equal to $K + 1$). Dropping these restrictions (i.e. allowing $N \neq F$, not enforcing $n_x \rightarrow f_x$ interdependencies, and allowing polygeny to differ from $K + 1$ for individual functions) provides a generalized NK-model of complex systems.

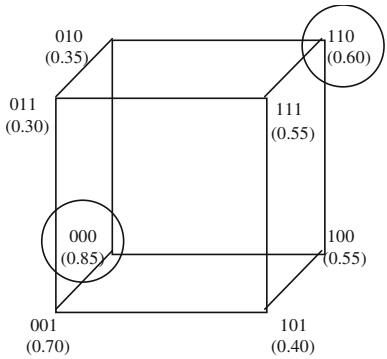
In Altenberg's generalized NK-model, the fitness landscapes are constructed in the same way as in the original formulation by Kauffman (1993). An example of a genotype–phenotype map is given in Fig. 1(a). In this example, the fitness of the first function w_1 is affected by the first and second elements, and the fitness of the second function w_2 is affected by second and third elements.

In this example, we assume, without loss of generality, that $A = 2$. Given the matrix specifying the system's architecture and the design space of all possible designs, the fitness landscape of a system can be simulated as in Fig. 1(b). A fitness landscape is a mapping of fitness values onto all possible designs.



(a) Example of a genotype-phenotype map

	w_1	w_2	W
000:	0.8	0.9	0.85
001:	0.8	0.6	0.70
010:	0.4	0.3	0.35
011:	0.4	0.2	0.30
100:	0.2	0.9	0.55
101:	0.2	0.6	0.40
110:	0.9	0.3	0.60
111:	0.9	0.2	0.55



(b) A simulation of the genotype-phenotype matrix in (a)

Fig. 1 Altenberg's generalized NK-model (**a**, **b**)

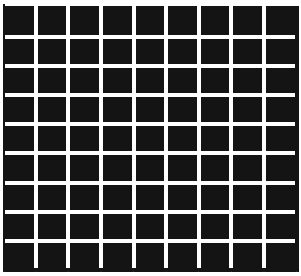
As with Kauffman's original NK-model, the fitness landscape is generated by randomly drawing a fitness value from a uniform distribution between 0 and 1 for each possible setting of alleles of the elements affecting a function f. Total fitness is then derived as the normalized sum of the fitness values of all functions:

$$W = \frac{1}{\mathrm{F}} \cdot \sum_{f=1}^{\mathrm{F}} \mathrm{w_f} \tag{3}$$

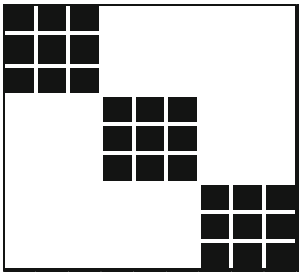
2.2 Non-decomposable, decomposable and modular systems

Using Altenberg's generalized NK approach, one can conceptualize interface standards as elements that do not have an intrinsic function, but solely affect functions that are associated with other elements (Frenken 2006). Figure 1 provides an example of a modular system, albeit the most elementary one. The second element affects both functions, each of which is associated with one of the other two elements. Once the choice of the second element is made (i.e. the interface standard), each function can be optimized independently by tuning the element affecting it. Depending on whether the standard is 0 or 1, the designer ends up in either 000 or 110 (circled in the figure).

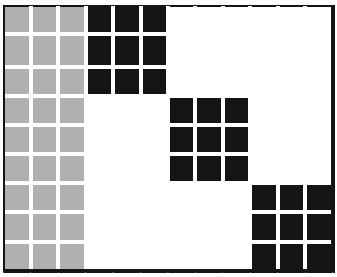
In Fig. 2, an example is given of three types of systems that can now be distinguished. System (a) is an NK-system in the sense of Kauffman's Kauffman (1993) original NK-model. For this system, $N = 9$ and $K = 8$ (maximum polygeny). Since the system is not decomposable, the time required to



(a) Non-decomposable system, polygeny = 9



(b) Decomposable system, polygeny = 3



(c) Modular system, polygeny = 6
(interface standards indicated in gray)

Fig. 2 Three complex systems (*rows* polygeny vectors; *columns* pleiotropy vectors) (**a**–**c**)

globally optimize the system equals the size of the design space. Assuming again that A = 2, the size of the design space is $2^9 = 512$. System (b) is a decomposable $N = 9$ system that can be decomposed into three, equally sized with a polygeny of three ($K = 2$). The time required to globally optimize the system equals the size of the design space of each subsystem ($2^3 = 8$), because search can proceed in parallel (Frenken et al. 1999). Of course, a decomposable system with subsystems of size one, which corresponds to minimum polygeny ($K = 0$), would only require two periods to globally optimize (in fact, there are no local optima is such systems). The optimal level of decomposability with regard to the search time required to globally optimize the system, is a fully decomposable system with subsystems of size one.

System (c) is a modular system according to our previous definition with three subsystems of size three, which are mediated by three interface standards yielding a polygeny of six (each function is affected by three elements in the subsystem and the three interface standards). The total number of elements in the new system, denoted by N', is 12. Though the number of elements has been increased from nine to 12, the number of trials required to globally optimize the system *hierarchically* is much less that in case (a). For each set of interface standards, there exists an optimal setting of subsystems that can be found in $2^3 = 8$ periods (as for system b). As there are three standards, and thus $2^3 = 8$ settings of interface standards, the total time required adds up $8 \cdot 8 = 64$ periods. Thus, comparing system (a) with system (c), an increase in the number of design dimensions in a system actually simplifies the search for its optimal solution. A modular system can thus be constructed by *increasing* the number of elements in the system such that the elements become organized in modules, thereby *decreasing* the complexity of a system in terms of the search time required for global optimization. Contrary to decomposable systems, the optimal level of modularity with regard to the search time required to globally optimize the system is non-trivial.

3 Optimal modularity in two-layered hierarchies

We first investigate the case of two-layered hierarchies (precluding modules within modules) before proceeding with the generalized case of multi-layered hierarchies (allowing modules within modules) in the next section. The number of modules in which a system can be modularized varies between a single module (absence of modularity) and N modules (maximum modularity). The question becomes how many modules should be created as a function of the original size N of a non-decomposable system. Following our example of Fig. 2(c), we make three assumptions.

Assumption 1 *The number of interface standards in a modular system equals the number of modules in a modular system (given that modular systems contain two or more modules).*

Assumption 2 *An interface standard affects all functions, i.e., the pleiotropy of a standard equals F.*

Assumption 3 *All modules are of equal size, the possible sizes ranging from one module of size N (absence of modularity) to N modules of size one (maximum modularity).*

The first assumption is not crucial to our argument, and can be relaxed. The reasoning behind this assumption is that more modules require more interface standards. The second assumption defines a standard as an interface between all elements. As an interface standard affects all functions, all the

fitness values of the non-modular system are redrawn to obtain the fitness values of the modular system. Note that this implies that the fitness values of a modular system are uncorrelated with the fitness values of the original non-modular system. The third assumption follows from the principle of cover size (Page 1996), which states that the search time to globally optimize a system is bounded by the size of the largest subsystem. Thus, optimal modularity requires the partitioning the system in equally sized modules.

To minimize the number of trials required to solve the system, one needs to compute the optimal level of modularity. Let N stand for the size of original non-decomposable system, as in the original NK-model. Let N' stand for the size of original non-decomposable system plus the number of interface standards. Let M stand for the number of interface standards. Finally, let S stand for module size. It follows from assumption 1 that $N' = N + M$ and from assumption 3 that $S = N/M$.

Within this framework, it can be shown that maximum modularity, unlike maximum decomposability, can never be optimal in terms of minimizing the time required to find the global optimum. Consider the case of modifying the example in Fig. 2(a) to be of maximum modularity as in Fig. 3, by adding nine interface standards to the system (of nine elements). Assuming $A = 2$, there are 512 unique options for interfaces and two unique options for each module. Thus, optimization requires $512 \times 2 = 1024$ periods, compared to the 512 periods to optimize the original NK-system as depicted in Fig. 2(a). It holds that, for any N, polygeny in a maximally modular system is N + 1 (N interface standards plus the function itself), compared to a polygeny of N for non-decomposable systems. So maximum modularity can never be the optimal solution. It will be shown that optimal modularity is determined by minimization of polygeny, which stands in a nonlinear relationship with level of modularity.

Global optimization of a module requires exhaustive search, that is, the testing of all possible combinations. Optimizing a module, given a set of standards, thus equals $A^{N/M}$ periods. As modules can be searched in parallel, the time required to optimize all modules is equal to the time required to optimize a single module. The number of possible sets of standards equals A^{M}. Optimal modularity, i.e. the optimal number of modules, can now be derived as the number of modules that minimizes search time required to globally

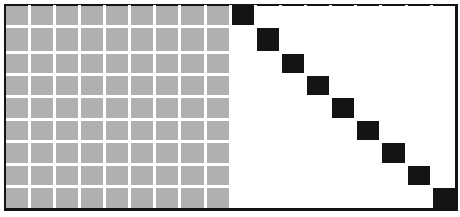


Fig. 3 Maximum modularity

Maximum modularity, N'=18, F=9, S=1, polygeny = 10
(interface standards indicated in gray)

optimize the system. The time required to globally optimize a modular system, C_{time}, is given by the product of time required to solve a module ($A^{N/M}$) and the time required to design all possible architectures (A^{M}):

$$C_{time} = A^{N/M} \cdot A^{M} = A^{(N/M)+M} \tag{4}$$

This process is guaranteed to find the global optimum as it optimizes each module for all possible architectures. Note that the exponent of Eq. 4 represents the polygeny of the elements. The optimal number of modules can be derived by minimizing (4) with respect to M, which yields:

$$M = N^{1/2} \tag{5}$$

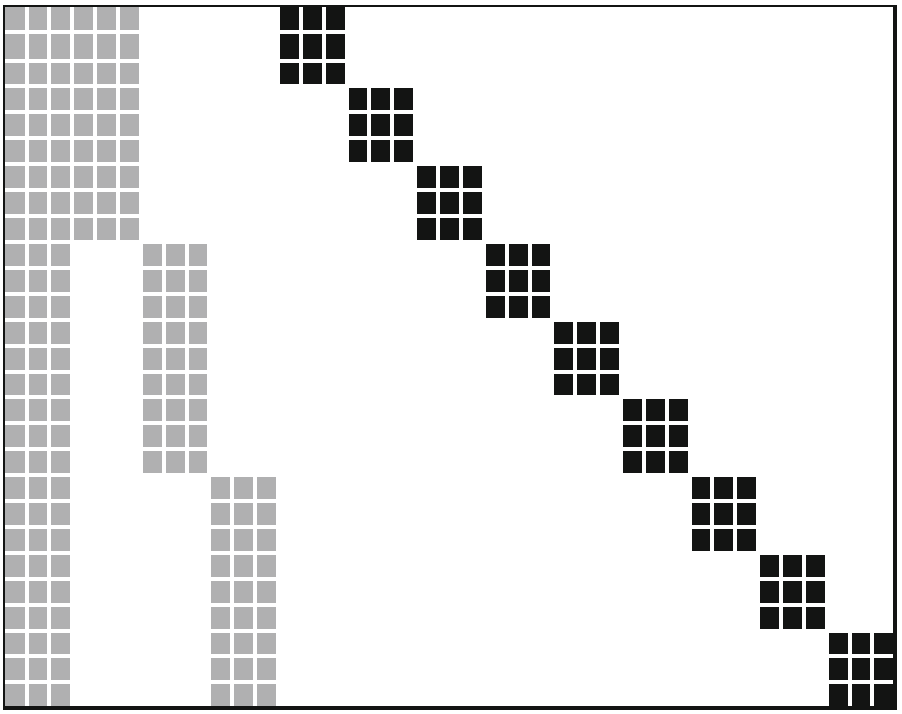
Thus, the optimal number of modules to be created in a non-decomposable system that originally has N elements equals the square root of N (a result independent of A). Given $S = N/M$ it follows that the optimal module size also equals the square root of N. The resulting time required to globally optimize the optimal modular system, equals:

$$C_{optimal} = A^{2(N)^{1/2}} \tag{6}$$

4 Optimal modularity in multi-layered hierarchies

The analysis has thus far only considered modular systems with two layers: a layer of interface standards and a layer of modules. Our reasoning can be generalized for modular systems with more than two layers by considering an iterated modularization process. Iterated modularization allows for the formation of more than two levels (*i.e.* for the creation of a hierarchy of modules within modules). In order to derive the optimal modularity for a hierarchy of modules, we introduce variable L, which stands for number of levels of modularization. Under this notation, $L = 1$ stands for no modularization and $L = 2$ describes the single level of interfaces considered in the previous section. We now consider the general case of $L \geq 2$.

A module is defined recursively within a perfect n-ary tree structure. This structure is a simple, analytically tractable construct from computer science for representations of hierarchical systems. Formally, it is a tree in which every internal node has exactly n children. (See Fig. 4 for an example of $n = 3$ represented as genotype–phenotype matrix and Fig. 5 for the same example represented as a perfect 3-ary tree.) Within this structure, modules are defined recursively as being formed of a set of interface standards and a set of child modules. In the limit of the bottom of the hierarchy, we reach the leaf modules (those modules which have intrinsic functions). If we take functional (leaf) modules to be Func, the size of the leaf modules to be S (i.e. $|\{\text{Func}\}| = S$), a module at level n to be mod_n, the set of modules at level n to be Mod_n, and



Three-layered modularity, N=27, N'=39, F=9, S=3, L=3, polygeny = 9 (interface standards in gray)

Fig. 4 Multi-level modularity

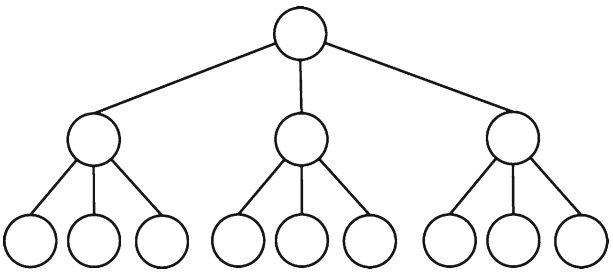
interface standards at level n to be IS_n, a hierarchical modular system can be formally written as:

$$\begin{aligned} \text{mod}_n &= \{\{IS_n\}, \{Mod_{n+1}\} : n \neq L\} \\ \text{mod}_n &= \{\{Func\} : n = L\} \end{aligned}$$

Where:

$$Mod_n = \{\{\text{mod}_{n+1}\}, \{\text{mod}_{n+1}\}, ...\}$$

Fig. 5 Perfect 3-ary Tree, Height = 3



The system in Fig. 4 graphically represents the following N = 27, L = 3 system:

$$\begin{aligned}\mathrm{mod}_1 &= \{\{IS_1\}, \{Mod_2\}\} = \{\{IS_1\}, \{\{\mathrm{mod}_2\}, \{\mathrm{mod}_2\}, \{\mathrm{mod}_2\}\}\}\\ \mathrm{mod}_2 &= \{\{IS_2\}, \{Mod_3\}\} = \{\{IS_2\}, \{\{\mathrm{mod}_3\}, \{\mathrm{mod}_3\}, \{\mathrm{mod}_3\}\}\}\\ \mathrm{mod}_3 &= \{Func\}\end{aligned}$$

Note that Fig. 4 represents a projection of the hierarchical structure shown in Fig. 5 onto the NK structure. Within this projection, the following assumptions are implicit:

Assumption 4 *The number of interface standards at any level of the hierarchy equals the size of the leaf modules (i.e.* $|\{\mathrm{IS_n}\}| = \mathrm{S}$*).*

Assumption 5 *A standard affects all functions in the level at and below the standard in the hierarchy (i.e. top level standards affect all functions).*

Assumption 6 *Division into modules is symmetrical across levels (i.e.* $|\{\mathrm{Mod_n}\}| = \mathrm{S}$*).*

All functions are thus affected by at least one standard in the multi-layered case. As for the single-layered case, this implies that the fitness values of a multi-layered modular system are uncorrelated with the fitness values of the original non-modular system.

To globally optimize this multi-layered modular system, one has to search *hierarchically* via multiple cycles fixing the standards at the top level, then fixing the standards at the middle level, and then optimize each leaf module. Since the top level interface consists of three interfaces, there are 2^3 possible standard settings at the first layer requiring 2^3 cycles of exploration. For each of the settings at the first layer, testing the middle layer interfaces also involves 2^3 cycles because each subsystem can be searched in parallel. Finally, optimizing each individual module takes 2^3 periods. Thus, total search time is $2^3 \cdot 2^3 \cdot 2^3 = 2^9 = 512$ periods, which is a small fraction of the time required to globally optimize a system of $N = 27$ (which requires 2^{27} periods).

Again, we look to minimize the time to globally optimize the system. Since, by Assumption 4, the number of interface standards equals the number of elements in the leaf modules, each level of the hierarchy takes the same amount of time to optimize. Thus, the time to global optimization is simply the product of the time required to optimize each level. We have, for modular systems:

$$C_{time} = (A^S)^L = A^{SL} \tag{7}$$

Due to symmetry, we may replace the size of the leaf modules (S) with an equivalent term utilising N and L. This relationship is (as shown by detailed proof in Appendix A):

$$S = N^{1/L} \tag{8}$$

Combining (7) and (8) gives:

$$C_{time} = A^{LN^{1/L}} \tag{9}$$

The previous, non-iterated optimization result may thus be seen as a specific case of this result with $L = 2$. Minimizing (9) with respect to L gives:

$$N^{1/L} + L(-L^{-2})N^{1/L}\ln(N) = 0$$

$$L = \ln(N) \tag{10}$$

Then the time that is required to globally optimize the optimal modular system is:

$$C_{optimal} = A^{(\ln N)N^{1/\ln N}} \tag{11}$$

Note again that optimal level of modularity is independent of A.

Given (8) and (10), one can derive the optimal module size:

$$S = N^{1/\ln N}$$

$$\ln S = 1$$

$$S = e \tag{12}$$

Given optimal module size, one can now derive the values of N at which optimal modularity requires the introduction of a new layer. One should introduce a new layer of modules moving from $L = x$ to $L = x + 1$ if:

$$N = e^{x+1} \tag{13}$$

At this size, the system can be symmetrically divided into $x + 1$ layers with modules of optimal size.

It follows from Eq. 13 that, as N increases exponentially, L increases linearly; a corollary is that, as N increases linearly, L increases logarithmically. Modularity thus represents a mechanism for coping with exponential system growth. It also suggests a hypothesis that early in linear growth processes of a complex system, modularity structure will change regularly, while later in the process, changes in modularity structure will be increasingly rarer. Introduction of a modular structure slows the growth of polygeny relative to system size.

In order to understand the consequences of deviating from optimal modularity, we examined whether under- or over-modularization is more costly in terms of the additional search time required. Using Eq. 9, we plot computational complexity as $log_A(C_{time})$ for different values of L and N in Fig. 6. Note that we express the search time required for global optimization in terms of the logarithm of A, which render values of search time to be independent of A. The figure shows that computational complexity sharply decreases with addition of layers of modules, reaches the minimum, and then slowly increases. This suggests that it is a more robust strategy to over-modularize than under-modularize.

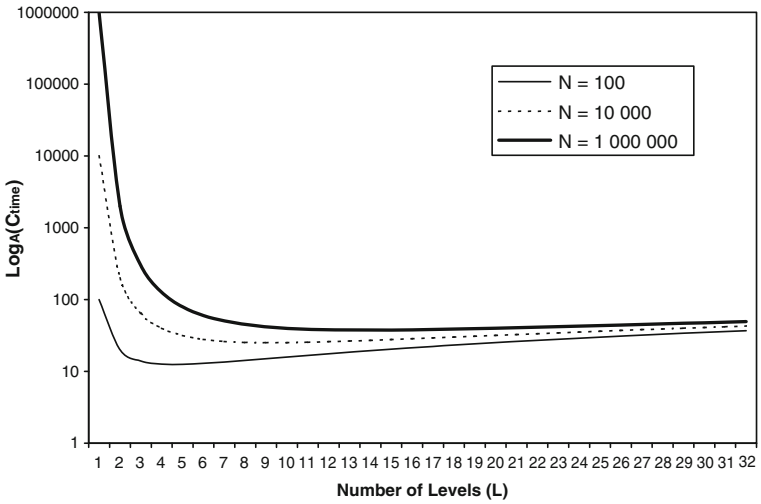


Fig. 6 Search time required to find the global optimum

The question of whether under- or over-modularity is to be preferred has important general implications. They suggest, for example, heuristic strategies for product design under conditions of uncertainty. However, different models within the NK tradition exhibit conflicting results on this question. Geisendorf (2010) summarizes the debate as between those (Dosi and Marengo 2005; Brusoni et al. 2007) who find speed of evolution advantages to over-modularization and those (Levinthal 1997; Ethiraj and Levinthal 2004; Geisendorf 2010) who do not. There are also other papers which address the question which do not explicitly frame their results in terms of over-modularization (e.g. Frenken et al. 1999). These many papers vary significantly in their assumptions. These differences in assumptions (or model specifications) are important in explaining these divergent results. As an example of the understanding which can be gained from detailed comparison of specifications, we compare our model to the model by Ethiraj and Levinthal (2004) which did not find benefits to over-modularization. Here we present only the conclusions of this comparison, the full comparison being available in Appendix B. Our model focuses only on search time and thus only looks at the theorized advantages of modularity through reduced polygeny. The Ethiraj–Levinthal model features parallel search with no co-ordination regarding mutations in interface standards, meaning that increasing modularity leads to increasingly chaotic fitness dynamics. Given the many differences between the models, it is difficult to decide which model is likely to exhibit the more robust results.[1] It highlights the value of developing common frameworks in which differences between specifications can be tested more explicitly.

[1] Robustness refers to a result that survives changes in specification.

5 Discussion

This paper has focused on the implications of modular structure from an evolutionary time savings perspective. The question has been what kind of modular architecture is optimal with respect to the speed at which trial-and-error search can find the global optimum. In line with recent ideas on evolvability (Ethiraj and Levinthal 2004; Rivkin and Siggelkow 2007), creating an architecture which allows efficient search may be as important as the search strategy applied to a given problem.[2] The interaction between the processes of architectural search and search within the current architecture is an interesting, though non-trivial, problem. Our analysis indeed shows that the choice of the right modular architecture creates strong advantages in the subsequent evolutionary search process towards the global optimum. If a designer is able to create a modular design with modules of optimal size, she realizes huge savings in the time required to find the global optimum by trial-and-error.

Our approach has been based on two important simplifications. First, we assumed that the creation of modular architectures did not itself involve time. The time devoted to creating a modular architecture will generally increase with the degree of modularity of that architecture (as more interface standards need to be introduced to separate elements into distinct modules). Once the problem of minimizing search time is translated into a cost minimization problem using a monetary value of time, the minimization problem can be extended to include the cost of the construction of an architecture, with such construction costs increasing with the degree of modularity as indicated previously by M. The cost perspective may have an impact on the desirability of over-modularization, as it would tend to attenuate the benefits of modularization.

A second simplification in our analysis is that our derivation of optimal modularity only takes into account the search time required to globally optimize the system and ignores the effects of modularization on the fitness value of the global optimum obtained. In our model, optimal modularity is achieved by minimizing the search time required for global optimization. Put differently, in designing a system with optimal modularity, one aims at minimizing the number of times the fitness values are redrawn. In the case of a non-modular system, for example, fitness values are redrawn A^N times, while for a system with optimal modularity, fitness values for each module are redrawn only $A^{(S+M)}$ times. Since fitness values are redrawn less often for a system with optimal modularity compared to other systems, this implies that the global optimum of a non-modular system has a higher fitness than the global optimum of a system with optimal modularity, since N is greater that S+M (Kaul and Jacobson 2006). Thus, the advantage of modular systems in terms of search time may be offset by lower fitness, depending on how much weight is given to fitness obtained compared to search time required.

[2] A similar argument has been made by Wagner and Altenberg (1996) in the context of biology.

Note that the fitness of the global optimum of a non-modular system and the fitness of the global optimum a system with optimal modularity approach 1 asymptotically as system size N goes to infinity. Thus, the difference in their fitness values will start decreasing at some point as system size N increases. For sufficiently large systems, the negative effect of modularization on the fitness of the global optimum is only marginal and can be neglected. This conclusion is tied to the global search strategy, which is employed here. Whether fitness effects can similarly be taken as minimal when alternative search strategies are employed is an open question.

A final area for future research is to consider different search strategies within the framework we have discussed. Similar to our argument about the status of decomposable systems, a global optimization strategy is an analytical construct rarely seen in reality. In reality, shifts in the underlying landscape, cognitive and power limitations of agents, and opportunity costs (i.e. the tension between exploration and exploitation) mean that search processes are in reality satisficing as opposed to optimizing (Simon 1969). This then poses the question as to what hierarchical search might look like in a satisficing context. Two possibilities are discussed which suggest the utility of this framework in terms of future research potential.

Gavetti et al. (2005) explore the idea of analogy as a search strategy within an NK context. Their conceptualization of search is the resolution of "high level" choices through analogical knowledge flowing from experience followed by resolution of "low level" choices through local search. In this case, analogy is a tool that leverages past experience in order to suggest promising segments of the landscape within which to search using local search. In the original paper, the idea was to explore knowledge derived from different regions of the fitness landscape. In the case of the modular structure proposed here, it would be interesting to explore experiential knowledge of architectures. This would mean setting the interfaces on the basis of analogy, followed by local search within these interfaces. A first step in this setup might be to set standards randomly and proceed with low-level search. This would provide a baseline for assessing the impact of architectural knowledge.

A second possibility is to model recursive problem solving (a concept inspired by Arthur 2007). The departure point of this is to frame invention as a recursive problem solving process, in which work on a solution proceeds between levels and focuses on the most problematic component. This could be abstracted as a hierarchical extremal search wherein the lowest functioning module is the focus of search. If a satisfactory solution can be found at the level of the module, then it is resolved at that level. If this is not the case, search proceeds down the hierarchy in a recursive manner. After sufficient exploration of sub-modules, if a satisfactory solution is still not found, then the problem is elevated to the level of the hierarchy above which the problem occurred. In our terms, search begins at level N, moves down the hierarchy to level L, and then is elevated to exploration of the interfaces at level $N + 1$.

These discussion points indicate that, though the model we have presented is rather restrictive in the assumptions it utilized, it offers interesting possibilities

for further research, which relaxes these assumptions. It represents the crucial first step of offering a formally consistent framework wherein an analytical baseline can be defined. Further, it confirms the primary hypothesis of the modularity literature: that modularity increases speed of evolution (Simon 2002). It does so by formally linking modular structure to a decrease in interdependencies between elements.

6 Concluding remarks

We had aimed to define modularity formally and explore the hypothesis that it represents a mechanism for increasing the speed of evolution. We have derived the optimal level of modularity with respect to the time required to globally optimize a system, both for two-layered hierarchies and multi-layered hierarchies. Our approach has taken advantage of rather restrictive assumptions in order to generate analytically tractable results. We have discussed several logical routes to relax these assumptions in future work.

A second line of research is to conduct empirical research on the levels of modularity of systems varying in size, so as to provide an empirical basis for the formal theory. For example, further work might be conducted into the suggestions that modularity of problem decomposition is observable in entrepreneurs who are involved in rapidly expanding enterprises (Sarasvathy and Simon 2000).

In the longer run, we hope our approach to modular systems contributes to a consistent formal approach to modularity in the fields of economics, innovation studies and organization science in a way that renders the results from different modelling exercises mutually comparable.

Open Access This article is distributed under the terms of the Creative Commons Attribution Noncommercial License which permits any noncommercial use, distribution, and reproduction in any medium, provided the original author(s) and source are credited.

Appendix A—Deriving Eq. 8

The relationship between the size of leaf modules S, the size of the system N, and the number of levels of modularity L may be derived via symmetry considerations.

We may start by considering the number of leaf modules, P. A property of perfect n-ary is that the number of nodes at a given height corresponds to the geometric sequence $\{1, M, M^2, \ldots, M^{H-1}\}$, where H is the height of the tree and M is the factor of division (the 'n' of the n-ary tree, but defined as M so as not to be confused with the system size N). The number of leaf nodes is just the last element of this sequence. Thus, we have:

$$P = M^{H-1} \tag{14}$$

For example, the tree shown in Fig. 5 has $M = 3$ and $H = 3$, so the number of leaf nodes is $3^{3-1} = 9$.

We can make substitutions to this general relationship using variables already introduced. By the symmetry introduced in Assumption 6, $M = S$ (i.e. the factor of division equals the size of leaf modules). Our definition of L is just the height of the tree, so $H = L$. Making these substitutions:

$$P = S^{L-1} \tag{15}$$

We now look to relate the number of leaf nodes P, to the system size N. Remember that the system size is just the number of elements, which have intrinsic functions. Given that these are assumed to be symmetrically distributed across the leaf nodes (by the definition of S), we can define the number of elements per leaf node (S) as:

$$\begin{aligned} S &= N/P \\ S &= N/S^{L-1} \\ S &= N^{1/L} \end{aligned} \tag{16}$$

Appendix B—Comparing Model Results in Terms of Degree of Modularity

Given that our model has come to an opposite conclusion to that of Ethiraj and Levinthal (2004), it is useful to compare the models to determine why different results were achieved. A summary of the comparison of the two models is provided in Table 1. As it only directly addresses the differences between these two models, it is not claimed that the classification is exhaustive. However, it does highlight differences which could be expected to arise between other models as well. Overall, the models are quite different in terms of their assumptions, despite the fact that both are NK-based models of modularity. A key result of this categorization is that the specification of such models has a large impact on the results reported. More detailed descriptions of models of modularity would make it easier to compare results and to understand divergences. It would also be helpful to compare models not only indirectly

Table 1 Comparison of Ethiraj and Levinthal (2004) and this paper

Criteria	Ethiraj and Levinthal (2004)	This Paper
Interface co-ordination	Independent	Hierarchical
Nature of search	Satisficing	Optimizing
Origins of modularity	Inherent	Constructive
Agency-architecture alignment	Not aligned	Aligned
Independent variables	Degree of agency-architecture misalignment, satisficing heuristics, degree of intra-module competition	Architecture
Dependent variable	Hybrid	Search time

through their descriptions but also directly through reformulation in a common framework.

A first obvious difference between the models is at the level of co-ordination of changes in the interfaces. In the Ethiraj–Levinthal model, each interface is under the control of a particular module but there is no coordination of changes of interfaces with other modules. In the model presented here, interfaces are changed hierarchically. Conceptually, modules accept changes to interfaces, which are coordinated by some agency external to the particular modules. There is obvious middle ground between these two approaches in a strategy of negotiated coordination in which module agents collectively have control of interfaces. These differences are captured under the Interface Coordination heading in Table 1.

Another important difference is the nature of the search algorithm utilized. This is just the well-known distinction between optimizing and satisficing search. It could be equivalently be categorized by whether search is global (optimizing) or local (satisficing). Of course, there is a great deal of variety among different satisficing heuristics, but that is beyond the scope of this discussion. This is the variable Nature of Search in Table 1.

A distinction which occurs at a more conceptual level is the implicit theory of the origins of modular structure. The Ethiraj–Levinthal model treats modularity structure as something, which is to be discovered, as inherent (technological) relationships between elements. However, the literature on modularity sometimes describes modularity as constructed. For example, Langlois and Garzarelli (2008) see modularity as one option for design choice for software. This latter view is also seen in the model presented in this paper. Intermediate positions exist as well, where a certain structure of interdependencies is initially proposed but may be modified via investment (i.e. Baldwin 2008). These options, classed as Origins of Modularity in Table 1, can be referred to as the inherent and constructive respectively. A heuristic to differentiate between different processes of generating modularity is whether additional interface elements are added to the system, as in our model. If this is the case, then the model is generally towards the constructive end of the scale.

An important differentiator between the Ethiraj–Levinthal model and our model is the degree to which agency aligns with architecture; that is, the degree to which the control of elements by different agents matches the underlying interdependency structure. In our model, this alignment is perfect. In the EL model, the explicit purpose is to explore issues of misalignment. This could be thought of as what might happen if important inter-module interactions are not captured by the interfaces. This distinction will separate models, which are about the benefits and costs of modularity proper from the benefits and costs of imperfect modularity. This is the Agency-Architecture Alignment in Table 1.

It is also relevant to consider the elements that vary within each model. These elements can be thought of the independent variables of the model. In the case of our model, the independent variable is the degree of modularity. In the Ethiraj–Levinthal model the independent variables are: degree of agency-interdependency misalignment, the decision making mechanisms (mutation,

module-fitness driven recombinatory, overall-fitness driven), and the number of agents per module. This is summarized as Independent Variables in Table 1.

Finally, a key distinction is the issue of how the performance of different approaches is compared within the models. Ethiraj–Levinthal primarily consider the effects of different architectures on fitness, while our model primarily considers the effect of different architectures on search time. This is one of the most important variables to consider in comparing different models because, according to modularity theory, modularity trades-off long-term fitness for speed of evolution. It would be unsurprising, then, that approaches that primarily consider search time will report more positive effects of modularity than those that primarily consider fitness. Ultimately we might be interested in examining the interplay between the two by focusing on some hybrid variable such as time-weighted fitness. This is summarized under Performance Measure in Table 1.

We can then use Table 1 to compare the model results. For the model presented here, there are two important points. First, in terms of Dependent Variable, it only considers the search time aspect of modularity, which should make modularity more advantageous. Second, Agency-Architecture Alignment is relevant. Since agency is aligned with architecture, our meaning of over-modularity only refers to architecture. These factors imply that a) that modularity is quite preferable and b) that there are not complicating factors of how agency is constituted. The Ethiraj–Levinthal model is more complex to analyze. Interface Co-ordination is certainly a factor, as evidenced by their analysis of the problems with over-modularity. They explain the increasingly chaotic fitness dynamics under increasing modularity through the fact that agents performed parallel search with no co-ordination regarding mutations (no interface co-ordination in our terms). In terms of Agency-Architecture Alignment, there is variety in degree of alignment. In fact, over-modularity is defined relative to perfect alignment. So, a) the effect seems to be driven by the de-stabilizing effects of a lack of interface co-ordination and b) their definition of over-modularity is different from ours.

This comparison highlights three important points. First, results about the advantages and disadvantages of modularity are highly dependent on the specifications of the model being used.[3] An advantage of a framework such as the one developed here is that it would make specifications more transparent and comparable. Second, it highlights the importance of being circumspect about results of a given specification showing that a given architectural choice is to be preferred to another. Comparison of our model and the model by Ethiraj and Levinthal (2004) demonstrates that conclusions about degree of modularity are contingent on the model conditions. Factors which may influence the results include: the variability of the landscape, the

[3] In fact, under one variant of the Ethiraj and Levinthal specification (2004, p.170), over-modularity is in fact preferable. Thus, even within the confines of a particular approach, small variances in specification can be important.

experience of designers with a given landscape (related to uncertainty about interdependency structures), the fixed costs vs. variable benefits of modifying architectures, etc… Third, it is advantageous to analyze the expected results of a given specification in terms of the fitness-search time trade-off theory. With greater clarity about how model results fit with theory, then we will have a much stronger sense of whether particular models exhibit anomalous behaviour.

Returning to the original question of this section, we are now in a better position to assess the relative merits of the two models on the over- vs. under-modularity question. A simple minded analysis would be that, in our model, we focused on the benefits of modularity and found that more modularity is better, while the Ethiraj–Levinthal model focused on the costs of modularity and found that less modularity is better. This does not seem definitive in either direction. It could be argued that the Ethiraj–Levinthal result should be preferred as making more realistic assumptions (local and satisficing). However, assumptions such as the complete lack of coordination around standards would seem to have rather narrow applicability. Analysis of these two models in isolation does not suggest any robust conclusion as to the optimal degree of modularity. Comparison with other models of modularity would be necessary to draw stronger conclusions.

It would be beneficial to undertake a broader project of comparison with other models of modularity, either pair wise or preferably multi-model. However, this is non-trivial. There are significant differences in the details provided by the authors about their models. An in-depth comparison, and especially a multi-model comparison, would reveal significant gaps in reporting. Direct contact with the authors would likely be necessary to fill these gaps. Even this may be insufficient, requiring instead an effort to replicate the models in a common framework. We do not attempt the implementation of such a comparison here. But we do note that this discussion re-enforces the need to do detailed comparisons of specifications and of the value of creating frameworks in which it would be possible to compare model results.

References

Altenberg L (1994) Evolving better representations through selective genome growth. In: Proceedings of the IEEE world congress on computational intelligence, pp 182–187

Arthur WB (2007) The structure of invention. Res Policy 36(2):274–287

Baldwin CY (2008) Where do transactions come from? Modularity, transactions, and the boundaries of firms. Ind Corp Change 17(1):155–195

Baldwin CY, Clark KB (2000) Design rules, vol 1: the power of modularity. Cambridge: MIT Press

Bradshaw G (1992) The airplane and the logic of invention. In: RN Giere (ed) Cognitive models of science. The University of Minnesota Press, Minneapolis, pp 239–250

Brusoni S, Marengo L, Prencipe A, Valente M (2007) The value and costs of modularity: a problem-solving perspective. Eur Manage Rev 4:121–132

Ciarli T, Leoncini R, Montresor S, Valente M (2008) Technological change and the vertical organization of industries. J Evol Econ 18:367–387

Dosi G, Marengo L (2005) Division of labor, organizational coordination and market mechanisms in collective problem-solving. J Econ Behav Organ 58:303–326
Ethiraj SK, Levinthal DA (2004) Modularity and innovation in complex systems. Manage Sci 50:159–173
Frenken K (2006) A fitness landscape approach to technological complexity, modularity, and vertical disintegration. Struct Change Econ Dynam 17(3):288–305
Frenken K, Nuvolari A (2004). The early development of the steam engine: an evolutionary interpretation using complexity theory. Ind Corp Change 13(2):419–450
Frenken K, Marengo L, Valente M (1999) Interdependencies, near-decomposability and adaptation. In: Brenner T (ed) Computational techniques for modelling learning in economics. Kluwer, Boston, pp 145–165
Gavetti G, Levinthal DA, Rivkin JW (2005) Strategy making in novel and complex worlds: the power of analogy. Strateg Manage J 26(8):691–712
Geisendorf S (2010) Searching NK fitness landscapes: on the trade off between speed and quality in complex problem solving. Comput Econ 35:395–406
Henderson RM, Clark KB (1990) Architectural innovation. Admin Sci Q 35:9–30
Kauffman SA (1993) The origins of order. Self-organization and selection in evolution. Oxford University Press, New York
Kaul H, Jacobson SH (2006) Global optima results for the Kauffman NK model. Math Program 106(2):319–338
Langlois R, Garzarelli G (2008) Of hackers and hairdressers: modularity and the organizational economics of open-source collaboration. Ind Innov 15(2):125–143
Levinthal DA (1997) Adaptation on rugged landscapes. Manage Sci 43:934–950
Marengo L, Dosi G, Legrenzi P, Pasquali C (2000) The structure of problem-solving knowledge and the structure of organizations. Ind Corp Change 9:757–788
McNerney J, Farmer JD, Rdener S, Trancik JE (2011) The role of design complexity in technology improvement. Proc Natl Acad Sci USA 108(22):9008–9013
Newman MEJ (2006) Modularity and community structure in networks. Proc Natl Acad Sci USA 103:8577–8582
Page SE (1996) Two measures of difficulty. Econ Theory 8:321–346
Rivkin JW, Siggelkow N (2007) Patterned interactions in complex systems: Implications for exploration. Manage Sci 53:1068–1085
Sarasvathy S, Simon HA (2000) Effectuation, near-decomposability, and the creation and growth of entrepreneurial firms. In: Presented at the first research policy technology entrepreneurship conference, University of Maryland. Available online at http://www.effectuation.org/ftp/Neardeco.doc., accessed on 18 September 2009
Simon HA (1962) The architecture of complexity: hierarchic systems. Proc Amer Phil Soc 106:467–482
Simon HA (1969) The sciences of the artificial, third edition. MIT Press, Cambridge, 1996
Simon HA (2002) Near decomposability and the speed of evolution. Ind Corp Change 11:587–599
Wagner GP, Altenberg L (1996) Complex adaptations and the evolution of evolvability. Evol 50:967–976