Spontaneous evolution of modularity and network motifs

Nadav Kashtan and Uri Alon*

Departments of Molecular Cell Biology and Physics of Complex Systems, The Weizmann Institute of Science, Rehovot 76100, Israel

Edited by Curtis G. Callan, Jr., Princeton University, Princeton, NJ, and approved August 2, 2005 (received for review May 10, 2005)

Biological networks have an inherent simplicity: they are modular with a design that can be separated into units that perform almost independently. Furthermore, they show reuse of recurring patterns termed network motifs. Little is known about the evolutionary origin of these properties. Current models of biological evolution typically produce networks that are highly nonmodular and lack understandable motifs. Here, we suggest a possible explanation for the origin of modularity and network motifs in biology. We use standard evolutionary algorithms to evolve networks. A key feature in this study is evolution under an environment (evolutionary goal) that changes in a modular fashion. That is, we repeatedly switch between several goals, each made of a different combination of subgoals. We find that such "modularly varying goals" lead to the spontaneous evolution of modular network structure and network motifs. The resulting networks rapidly evolve to satisfy each of the different goals. Such switching between related goals may represent biological evolution in a changing environment that requires different combinations of a set of basic biological functions. The present study may shed light on the evolutionary forces that promote structural simplicity in biological networks and offers ways to improve the evolutionary design of engineered systems.

Biological and engineered systems share general design features: they display modularity, defined as the separability of the design into units that perform independently, at least to a first approximation (1–3, 5).† Furthermore, they show reuse of certain circuit patterns, termed network motifs (6–11), in many different parts of the system. These features allow construction of extremely complex systems by using simple building blocks (12).

These features of biological networks are not captured by most current models of biological evolution. For example, many models of biological evolution use computers to evolve networks to attain a defined goal. In these simulations, networks in a population are varied by means of mutations, crossover, and duplication (13–15). Networks that perform better are selected in the next generation. A well known feature of these computational models of biological evolution is that the evolved systems are usually intricately wired and nonmodular. The nonmodular solutions are often more highly optimized than their human-engineered counterparts (16, 17, 20).

The fundamental reason for the lack of modularity in these evolved networks is that modular structures are usually less optimal than nonmodular ones. Typically, there are many possible connections that break modularity and increase fitness. Thus, even an initially modular solution rapidly evolves into one of many possible nonmodular solutions.

Lack of modularity is one of the reasons that computational evolution can currently generate designs for simple tasks, but has difficulty in scaling up to higher complexity. In the field of evolutionary design of engineered systems, approaches have been developed to promote modularity. These include enforcement of regular structures by using tree architectures (1) or generative grammars (21). Other methods explicitly represent subsystems as modules and use them as building blocks, such as module libraries (22) and automatically defined functions (23). These approaches significantly improve artificial design. The main difference between these approaches and natural biological evolution is that they use high-level processes to preserve modularity against mutational forces. Many of these high-level processes are not currently known in biology.

To understand the origin of modularity and network motifs in biology one has to understand how these features can spontaneously evolve. Several studies suggested that duplication of subsystems (24) or selection for stability (25) or robustness (26) can promote modularity. H. Lipson and coworkers (27, †) suggested that modularity can spontaneously arise under changing environments. This suggestion is based on the expectation that designs with higher modularity have higher adaptability and therefore higher survival rates in changing environments. However, computer evolution simulations under randomly changing environments do not seem to be sufficient to produce modularity (25, 27).

Here, we build on the suggestion of Lipson *et al.* (25, 27, †). The key feature in our study is evolution under an environment (evolutionary goal) that changes with time in a modular fashion. That is, we repeatedly switch between several goals, each made of a different combination of subgoals. We find that such modularly varying goals lead to the spontaneous evolution of modular structure and network motifs.

Methods

Electronic Circuit Evolution. Circuits were represented by a binary genome with a fixed number of genes that encode NAND gates and one gene for each output. Each generation of the best L circuits passed unchanged to the next generation [elite strategy (28)]. Each circuit was randomly mutated (mutation probability $P_m = 0.7$ per genome). A fitness penalty of 0.2 was given for every gate above a predefined number of effective gates (11 gates for the circuits in Fig. 2), where we define "effective gates" as gates with a directed path to the output. Genome and genotype–phenotype mapping are described, respectively, in Fig. 6 and Table 1, which are published as supporting information on the PNAS web site. For Fig. 2, the population size was $S = 1000$ and $L = 300$, and for Fig. 4, $S = 2000$ and $L = 500$. Similar results were found under a range of the parameters, such as larger populations and smaller mutation rates, and when using both a crossover operator (15) and mutations.

Neural Network Evolution. The neural network (29) genome was of a fixed size of 15 genes each encoding a neuron (see Fig. 7 and Table 2, which are published as supporting information on the PNAS web site). The neurons were set in four layers with eight, four, two, and one neuron per layer. Connections were only between neighboring layers in a feed-forward manner. Connections from the retina were to the first layer only. The output was defined as the neuron at the fourth layer. Each neuron was given a maximal number of incoming connections as follows: three inputs for a neuron in the first, second, and third layer and two inputs for a neuron in the fourth layer. Each connection had weight -1 or 1. If two of the inputs were from the

PHYSICS

This paper was submitted directly (Track II) to the PNAS office.

*To whom correspondence should be addressed. E-mail: urialon@wisemail.weizmann.ac.il.

†Lipson, H., Pollack, J. B. & Suh, N. P., DETC'01 ASME Design Engineering Technical Conference, Sept. 9–12, 2001, Pittsburgh.

same neuron then the effective weight of the connection was the sum of the weights of the two connections. A penalty of 0.01 was applied for every additional neuron above a predefined number of neurons (here we used 13 neurons). Mutations and crossovers were used as evolutionary operators, with elite strategy, with $S = 600$ and $L = 150$. Crossover probability was $P_c = 0.5$. Mutation probability was $P_m = 0.5$ per genome.

Quantitative Measure of Modularity. To quantify network modularity we used a measure based on the approach of Newman and Girvan (31, 32). Briefly, the Newman and Girvan algorithm finds the division of the nodes into modules that maximizes a measure Q. This measure is defined by the fraction of the edges in the network that connect between nodes in a module minus the expected value of the same quantity in a network with the same assignment of nodes into modules but random connections between the nodes (33):

$$Q = \sum_{s=1}^{K}\left[\frac{l_s}{L} - \left(\frac{d_s}{2L}\right)^2\right], \quad [1]$$

where K is the number of modules, L is the number of edges in the network, l_s is the number of edges between nodes in module s, and d_s is the sum of the degrees of the nodes in module s. The rationale for this modularity measure is as follows (following ref. 33): a good partition of a network into modules must comprise many within-module edges and as few as possible between-module edges. However, if we try to minimize the number of between-module edges (or equivalently maximize the number of within-module edges), the optimal partition consists of a single module and no between-module edges. Eq. **1** addresses this difficulty by imposing $Q = 0$ if nodes are placed at random into modules or if all nodes are in the same module.

We further refined this measure by normalizing it with respect to randomized networks. We defined a normalized measure Q_m:

$$Q_m = (Q_{real} - Q_{rand})/(Q_{max} - Q_{rand}), \quad [2]$$

where Q_{real} is the Q value of the network, Q_{rand} is the average Q value of randomized networks, and Q_{max} is defined as the maximal possible Q value of a network with the same degree sequence as the real network. To compute Q_m, we first converted the network into a nondirected graph by ignoring edge directionality and calculated its Q_{real}. To measure Q_{rand} we used two different controls that yielded similar results. For the first control, we used randomized networks that preserve the degree sequence of the real network. For the second control, we computed the Q of networks coded by random genomes that mapped to networks with the same number of nodes as in the real network, using the same genome definition and genotype–phenotype mapping as in the experiment. We used 1,000 random networks for computing Q_{rand}.

To estimate Q_{max} we repeated the evolution simulations, with exactly the same settings (i.e., network size and evolution parameters), where instead of evolving the networks toward the original information processing goal, we define the goal as maximizing the modularity measure Q. Q_{max} was defined as the average Q over 100 simulations of the best evolved network.

The present Q_m measure of modularity normalizes out the effects of the details of the simulations, such as the genotype–phenotype mapping, and the evolutionary algorithm. We also used this measure to quantitate the modularity of three biological networks. All showed high modularity: The transcription network of the bacterium *Escherichia coli* (7) had $Q_m = 0.54$, the neuronal synaptic network of the nematode *Caenorhabditis elegans* (7) had $Q_m = 0.54$, and a human signal transduction interaction network (34) had $Q_m = 0.58$. Modularity measurements results are summarized in Tables 3–5, which are published as supporting information on the PNAS web site.

Detection of Network Motifs. Network motifs were detected by using previously described algorithms (6, 7). To detect network motifs, we counted the number of appearances of different subgraphs in the evolved ("real") network and compared it with the number of times they appeared in randomized networks. The randomized networks preserved the single-node characteristics of the real network, such as the incoming and outgoing degree sequence (and the layered structure in the case of neural networks, see *Supporting Text*, which is published as supporting information on the PNAS web site).

To display the network motifs and antimotifs for each network we computed a nonnormalized subgraph significance profile as described (8). Briefly, for each real network we used MFINDER1.2 software to find the significance of all three- and four-node subgraphs (network motifs detection tool). The significance profile is composed of the Z-score for each subgraph [$Z = (N_{real} - N_{rand})/\sigma$, where N_{real} and N_{rand} are the counts in the real and average count in the randomized networks, respectively, and σ is the SD in the randomized networks]. To correct for multiple hypothesis testing, we also evaluated the Z-score distribution by choosing a randomized network as the real network and comparing it with an ensemble of 100 other randomized networks. In all cases this process yielded $|Z| < 0.2$, much lower than the values of motifs found in the real networks.

Supporting Information. Further information is available in Figs. 6–9, *Supporting Text*, and Tables 1–6, which are published as supporting information on the PNAS web site.

Results

We used two well studied model systems. The first system was electronic combinatorial logic circuits. The circuits are composed of a single type of gate, NAND (the NOT-AND function). The circuit has several inputs, labeled X, Y, Z, W, etc. NAND-gates are universal, in the sense that any logical function can be implemented by circuit composed of NAND gates.

The goal for evolution is a logical function G. The fitness of a circuit is the fraction of times it gives the correct output, G, when evaluated over all possible combinations of Boolean values of the inputs. We applied a standard evolutionary algorithm (13–15) to seek a circuit that maximizes fitness and thus achieves the goal (20, 35). We start with a population of random genomes that represent randomly wired circuits. For each generation, each circuit has a preset probability (mutation rate) of a random change in one of the connections between its gates. Circuits with low fitness are removed from the population, and circuits with a high fitness are replicated, keeping a fixed population size. The process is then repeated.

We first evolved circuits toward a fixed goal,

$$G1 = (\text{X XOR Y}) \text{ AND } (\text{Z XOR W}), \quad [3]$$

where XOR is the exclusive-or function. The fitness of the population increased over time. A prefect solution was found within 10^5 generations in 36 of 50 experiments. In the successful experiments a perfect solution was found within 9,000 (+19,000, −2,000) generations (Fig. 1*a*, fitness vs. generations). Many different circuits that achieve a perfect solution were found in these experiments. Most perfect solutions used $n = 10$ gates. A typical solution is shown in Fig. 2*a*.

Despite the fact that the goal G1 can be decomposed into subproblems (e.g., two XOR and one AND operations), the architecture of the evolved circuits was almost always nonmodular. To quantify modularity of the circuits we used a modularity measure Q_m (see *Methods*). Nonmodular networks show $Q_m \approx 0$, whereas modular networks typically show Q_m values between 0.3 and 1. The evolved circuits show very low modularity: $Q_m = 0.12 \pm 0.02$.

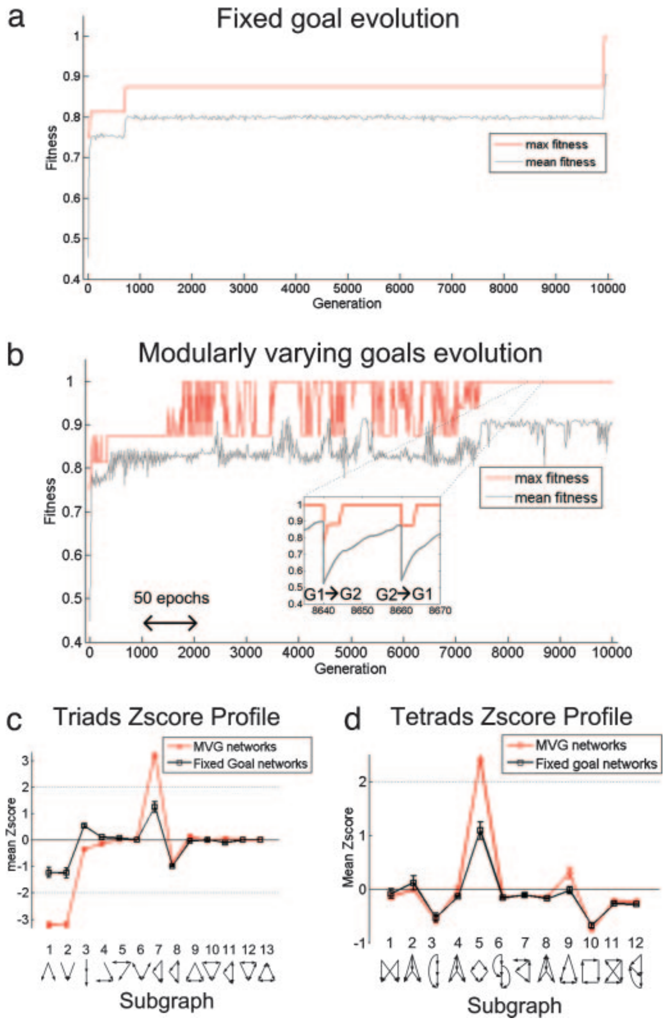

Fig. 1. Evolution of electronic circuits toward fixed and modularly varying goals. (*a*) Fitness as a function of generations under a fixed-goal G1 as defined in the text. Red indicates the best circuit in the population; gray indicates mean fitness. (*b*) Fitness as a function of generations under modularly varying goals evolution toward goals G1 and G2. The goal was switched every 20 generations. Fitness is shown in 20 generations resolution, just before every goal switch event. (*Inset*) Zooms into fitness around two switching events. The fitness drops and then recovers after each switch of the goal. (*c*) Significance of all three-node subgraphs compared with randomized networks. Mean Z-score $\pm$ SE is shown for $\approx$100 networks. Red circles indicate networks evolved under modularly varying goals (MVG); black squares indicate networks evolved under fixed-goal evolution to G1 and G2. The absolute significance of all subgraphs in corresponding random networks for both types of networks is $<$0.2. (*d*) Significance of four-node subgraphs. The four-node subgraphs displayed were selected as follows: for each type of network the 10 subgraphs with highest absolute Z-scores were selected. The 12 subgraphs shown are the union of the selected subgraph sets for the different networks.

Because of their nonmodular structure and intricate wiring, it was challenging to intuitively understand the way that these circuits function.

Next, we performed evolution in which the goal changed periodically between two different functions. Importantly, the two functions had shared subproblems. We term this approach "modularly varying goals." For example, we used goal G1 for an epoch of $E = 20$ generations, and then switched the goal to a similar function G2 in which two XOR computations are linked by an OR rather than an AND:

$$G2 = (\mathrm{X\ XOR\ Y})\ \mathrm{OR}\ (\mathrm{Z\ XOR\ W}). \quad [4]$$

We continued to switch between the goals every $E = 20$ generations. These switches were rapid in comparison to the total number of generation in the experiment, so that every evolutionary experiment included many switches. A perfect solution was found in all 50 experiments in $<$10,000 generations (mean time to perfect solution was 1,400 $\pm$ 1,000 generations).

We found that the evolved circuits were able to adapt to perfect solutions for each of the two goals. Every time the goal switched the population was able to reach a perfect solution to the new goal within about five generations and remained perfect until the next switch. These evolvable solutions lasted for many epochs (Fig. 1*b*).

The structure of the evolvable circuits found in these experiments is highly modular. Their modularity is readily apparent by eye and can be also be quantified by the modularity measure yielding significant modularity $Q_{\mathrm{m}} = 0.54 \pm 0.02$. Two different examples are shown in Fig. 2 *b* and *c*. The solutions are composed of two identical modules, each performing a XOR computation, and a third module that performs AND or OR on the output of the XORs. Notably, the XOR modules, each made of four NAND gates, are precisely the minimal XOR implementation used in electronic engineering (36, 37). In each experiment, the difference between the perfect solutions for the two goals differ by two connections. This small difference explains how the population can rapidly adapt when the goal is switched (Fig. 1*b*).

The evolvable solutions were larger than the fixed-goal solutions and usually used $n = 11$ gates, as opposed to $n = 10$ gates in the solutions evolved under fixed-goal evolution.

We also analyzed the network motifs in the circuits that evolved under fixed goals and modularly varying goals. Network motifs were detected by comparing the subgraphs found in the evolved networks to those found in randomized networks with the same number of connections per gate. We found that solutions evolved under modularly varying goals have significantly more network motifs than fixed-goal solutions (Fig. 1 *c* and *d*): they display recurring patterns such as feed-forward loops and diamonds (subgraph 7 in Fig. 1*c* and subgraph 5 in Fig. 1*d*, respectively).

We also evolved circuits under varying goals that were nonmodular. For this purpose, we used random logic functions as goals and switched between them as described above. We found that the evolved networks typically had nonmodular architecture and no network motifs (see *Supporting Text*). These networks seemed to "forget" the previous goal and attempt to find a solution to the new goal from scratch. The same applies to cases in which one problem was G1 or G2, and the other problem was a random four-input logic function.

Next, we performed experiments where evolution started with a modular circuit, but under a fixed goal. We find that modularity decreased rapidly within a few tens of generations provided there is even a slight selection pressure for small circuit size (Fig. 3). This result highlights the role of modularly varying goals in preserving modularity in the face of more optimal nonmodular circuits.

The same conclusions were also found for larger circuits and more complex problems. For example, evolution of a six-input, three-output problem, under modularly varying goals produced circuits with modular architecture and network motifs (Fig. 4). These adapted circuits included three separate modules that compute XOR and were able to evolve to solve each of four modularly varying goals.

In addition to electronic circuits, we applied our approach to a second well studied computational model, pattern recognition using neural networks (38). We used a genotype–phenotype mapping that ensures that the neural networks are feed-forward networks composed of four layers of nodes, linked by weighted edges. Each node sums its weighted inputs and activates its outputs if the sum exceeds a threshold.

In our pattern-recognition problem, neurons receive inputs from a 4-pixel-wide by 2-pixel-high retina, in which each pixel can assume a value of 0 or 1. The goal is to recognize objects in the left and right

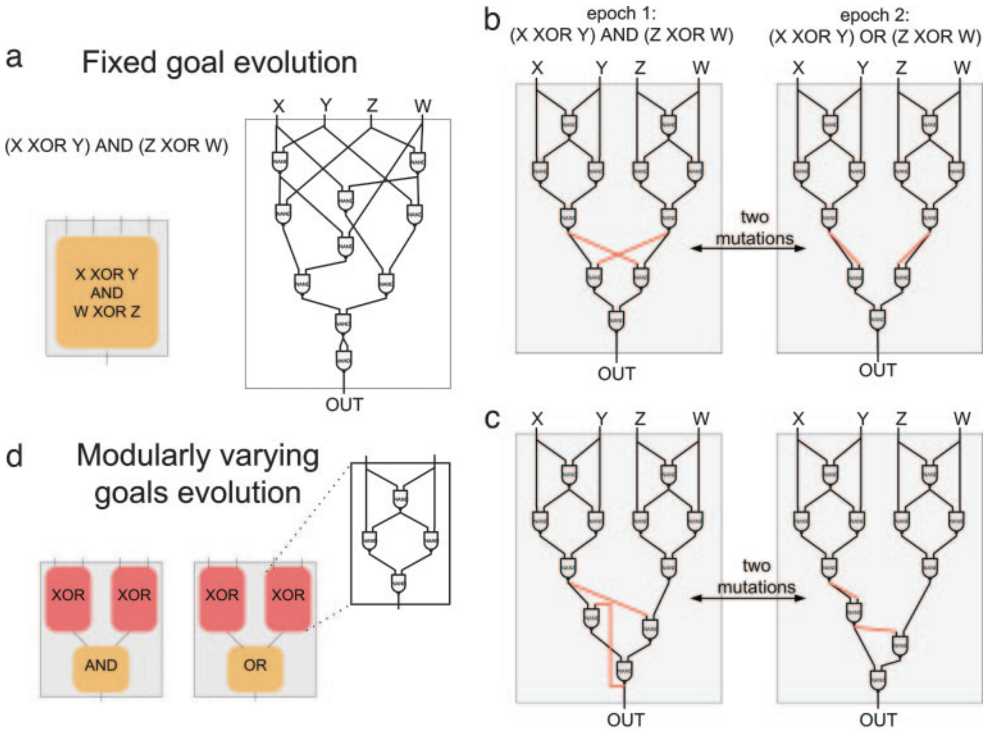

Fig. 2. Electronic circuits evolved under a fixed goal and modularly varying goals. (*a*) A typical circuit evolved by fixed-goal evolution toward goal G1. Each gate in the circuit represents a NAND gate. Similar nonmodular solutions were found for goal G2. (*b*) Circuits evolved with modularly varying goals evolution. Connections that are rewired when the goal is switched are marked in red. (*c*) Another example of a circuit evolved with modularly varying goals evolution. The circuit shows feedbacks between two of the three lower NAND gates. (*d*) Modular structure of the circuits evolved under modularly varying goals. The circuits are composed of two XOR modules that input into a third module that implements an AND/OR function, depending on the goal. Each XOR module is composed of six nodes (two input nodes and four internal gates) and display two feed-forward loops and one diamond network motif.

sides of the retina (Fig. 5*a*). A left (or right) object exists if the four left (or right) pixels match one of the patterns of a predefined set (Fig. 5*a*). We chose an object recognition problem that is relatively difficult to avoid trivial network solutions. The network has to decide whether the input fits a certain combination of left and right objects (for example, the presence of both left and right objects). The output of the neural network is a node whose value is 0 or 1.

To evolve neural networks (30) we used a standard genetic algorithm with crossovers and mutations. The environment contained 100 different randomly chosen retina patterns. We started with a population of random genomes. In each generation a crossover was performed between two randomly chosen network genomes with a probability P_c. In addition, one connection, weight, or threshold in each network in the population was changed with a preset mutational probability P_m. The fitness of a network was defined by the fraction of correct recognitions in this environment. Networks with higher fitness were replicated. In the fixed-goal evolution problem, nearly perfect solutions (at least 95% correct recognition) were found after 21,000 (+29,000, −3,600) generations.

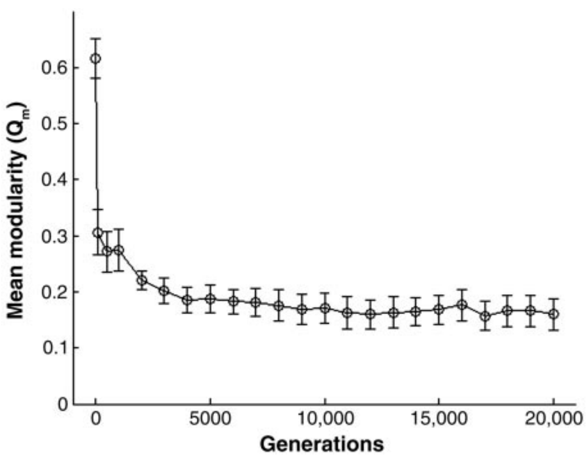

Fig. 3. An initially modular circuit rapidly loses modularity under fixed-goal conditions. Each experiment started from a population of identical modular circuits that had perfect fitness for goal G2 = (X XOR Y) OR (W XOR Z). At generation zero, the population was placed under a fixed-goal evolution with the same goal G2, with a selection pressure for small circuit size (a fitness penalty of 0.2 for every additional gate above the 10th gate). Mean modularity measure (±SE) vs. generations of best-fitness circuits is shown. Statistics are for 20 independent experiments, with four different initial modular circuits that satisfy G2.

A human engineer would easily notice the modularity in this problem and design a network that is made of two modules, one that analyzes the left side of the retina, and the other for the right side of the retina. In contrast, the structure of the evolved networks was not modular (Fig. 5*b*) (Q_m = 0.15 ± 0.02). As in the case of electronic circuits, fixed-goal evolution produces a nonmodular network even though the problem itself is readily decomposable into separate subgoals.

We then performed evolution under modularly varying goals by evolving networks while switching between two goals. The two goals used different combinations of subgoals defined on each half of the retina (Fig. 5*c*). The goals were switched every $E = 20$ generations. Evolution under these varying goals rapidly yielded nearly perfect networks with modular architecture (Q_m = 0.35 ± 0.02), within 2,800 (+9,500, −600) generations. Two distinct (nonidentical) modules evolved spontaneously in the network, each monitoring a different side of the retina (Fig. 5*e*). Thus, in contrast to fixed-goal evolution, the modular structure of the problems was "learned" and implemented in the solution, resulting in an intuitively understandable design.

The networks evolved under modularly varying goals were able to adapt to nearly perfect solutions for each new goal, within about three generations after the goal was switched. This evolvability was caused by the fact that the evolved networks for the different goals differed only slightly. For example, in many cases they differed in the threshold value of a single neuron, allowing switching between the networks with a single mutation (Fig. 5*c*).

To detect network motifs, we calculated the significance of each pattern in the network in comparison with a stringent ensemble of

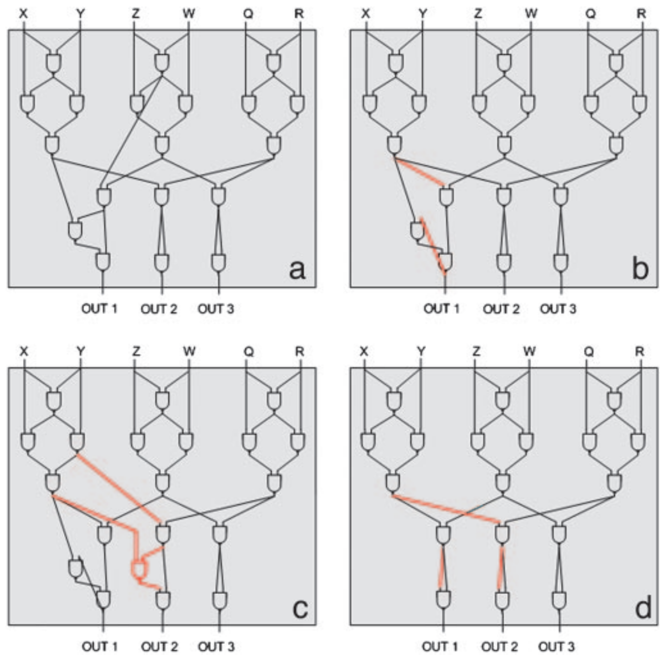

Fig. 4. An electronic circuit with six inputs and three outputs evolved under modularly varying goals, with the following four goals: G1 = (A OR B; A AND C; B AND C); G2 = (A AND B; A OR C, B AND C); G3 = (A AND B; A AND C; B OR C); and G4 = (A AND B; A AND C; B AND C), where A = X XOR Y, B = Z XOR W, C = Q XOR R. The goal was changed every 20 generations in the order G1, G4, G2, G4, G3, G4, G1, G4, etc. Perfect solutions that can rapidly switch between the goals evolved within $1.2 \times 10^5 \pm 8 \times 10^4$ generations. *a*, *b*, *c*, and *d* correspond to the networks found sequentially under goals G1, G4, G2, and G4, respectively. The lines and gates in red represent changes upon adaptation to each new goal.

randomized networks that preserved the degree sequence of each node as well as the layered structure of the network (see *Supporting Text* for details). We found that the neural networks evolved under modularly varying goals show much more pronounced network motifs than the networks evolved under a fixed goal. The motifs included the bifan and diamond patterns (patterns 3 and 5 in Fig. 5*d*). The networks also showed specific antimotifs (patterns that occur significantly less often than random networks) (subgraphs 1, 2, and 6 in Fig. 5*d*).

Discussion

In summary, we find that modularly varying goals can yield spontaneous evolution of modular network architectures with pronounced network motifs. In contrast, the same evolutionary process working under a fixed goal typically yields nonmodular solutions with fewer network motifs.

Networks that evolve under modularly varying goals seem to discover the basic subproblems common to the different goals and to evolve a distinct structural module to implement each of these subproblems. Evolution under modularly varying goals produces networks that can rapidly adapt to each of the different goals by only a few rewiring changes.

Not every set of changing goals leads to modular structures. Networks evolved under randomly varying goals (with no common subgoals) do not seem to evolve modular structure. In such cases, when the goal changes, the networks take a relatively long time to adapt to the new goal, as if it starts evolution from scratch. Under modularly varying goals, in contrast, adaptation to the new goal is greatly speeded up by the presence of the existing modules that were useful for the previous goal.

Why do modularly varying goals speed up evolution (in terms of the number of generations to reach perfect solution) when compared with evolution under a fixed goal? One reason that fixed-goal evolution is often slow is that the population becomes stuck in local fitness maxima. Because the fitness landscape changes each time that the goal changes, modularly varying goals can help move the population from these local traps. Over the course of many goal changes, modularly varying goals seem to guide the population toward a region of network space that contains fitness peaks for each of the goals in close proximity. This region seems to correspond to modular networks.

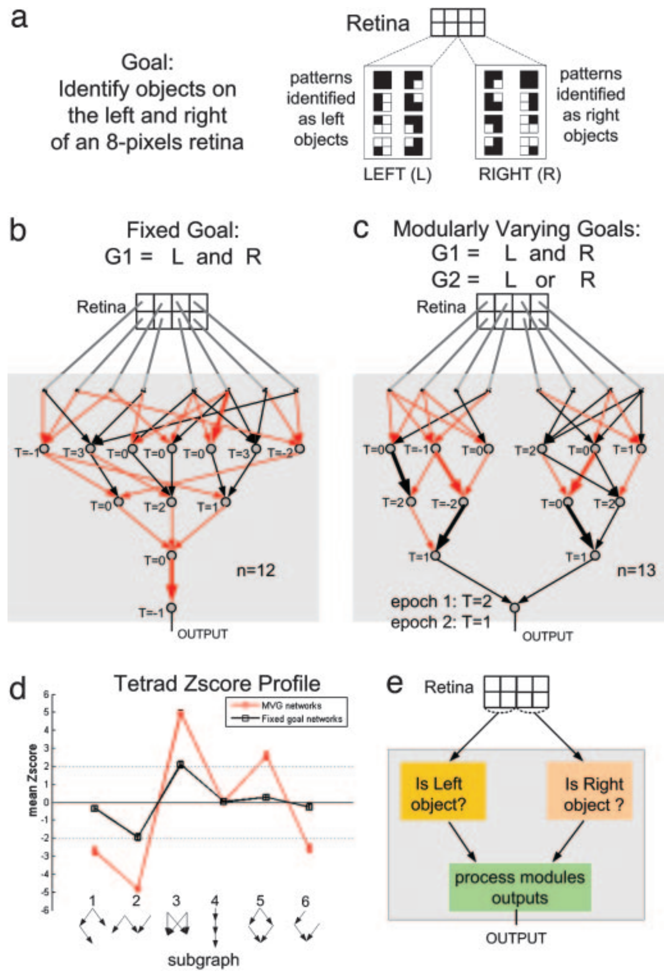

Fig. 5. Neural networks evolved for a pattern recognition task using fixed goals and modularly varying goals. (*a*) The aim is to recognize objects in the left and right sides of a 4-pixel-by-2-pixel retina. A left (or right) object exists if the four left (or right) pixels match one of the patterns of a predefined set. A left object is defined by three or more black pixels or one or two black pixels in the left column only. A right object is defined in a similar way, with one or two black pixels in the right column only. The goal is to identify when objects exist at both sides of the retina (L AND R). (*b*) A network evolved under fixed-goal evolution. Black/red lines represent positive/negative weights. Thick lines are double weights. (*c*) A network evolved under modularly varying goals evolution with two goals: L AND R, and L OR R. The network can rapidly adapt each time the goal is switched, by changing a single threshold of the lowest neuron (from $t = 2$ at the first goal to $t = 1$ at the second goal). In *b* and *c*, *n* indicates the number of neurons (nodes) in the network. (*d*) Four node patterns and their significance in the evolved networks. Shown are *Z*-scores ± SE for ≈50 networks. Strong motifs in the modularly varying goal networks are the bifan and diamond (patterns 3 and 5). Subgraphs were selected as in Fig. 1*d*. (*e*) Modular structure of the neuronal network evolved under modularly varying goals. Two distinct modules each monitor one side of the retina, and a third module processes their outputs.

In addition to their modular structure, the networks evolved under modularly varying goals display significant network motifs.

The same motifs are reused throughout each network in different modules. Some of these motifs are also found in biological information processing networks. For example, feed-forward loops and bifans are found in transcription networks (7). Feed-forward loops, bifans, and diamonds are found in signal transduction and synaptic neuronal networks (7). In signal transduction networks (34) and the neuronal network of *C. elegans* (39), multilayered feed-forward patterns similar to those in Fig. 5*c*, are strong network motifs. An example is multilayered protein kinase cascades, in which families of kinases in each layer activate families of kinases in the next layer (34, 40, 41).

One possible explanation for the origin of the motifs in the evolved networks is that modular networks are locally denser than nonmodular networks of the same size and connectivity. This local density tends to increase the number of subgraphs (42). To test this possibility, we evolved networks to reach the same modularity measure Q as the networks evolved under modularly varying goals, but with no information-processing goal (see *Supporting Text*). We find that these modular networks have no significant network motifs (Fig. 9). They show relatively abundant feedback loops that are antimotifs in the networks evolved under modularly varying goals. It therefore seems that the specific network motifs found in the evolved networks are not merely caused by local density, but may be useful building blocks for information processing.

How is evolution under modularly varying goals related to actual biological evolution? One may suggest that organisms evolve in environments that require a certain set of basic biological functions. When environments change, the same functions are needed but in different combinations. For example, consider the task of chemotaxis toward a nutrient (43, 44). Chemotaxis requires several functions, including sensing the nutrient, computing the direction of motion, moving the cell, and metabolizing the nutrient. Bacteria evolved specific gene modules to perform each of these tasks (e.g., a motor module, a signal-processing module, a module that transports the nutrient into the cell, etc.). When environments changed, these modules adapted over evolution to sense and chemotax toward other nutrients. Had evolution been in a fixed environment, perhaps a more optimal solution would have mixed the genes for these different tasks (e.g., a motor that can also sense and transport the nutrient into the cell), resulting in a nonmodular design.

An additional biological example occurs in development. Different cells in the developing embryo take on different fates. Each cell type needs to solve a similar set of problems: expressing a set of genes in response to a given time-dependent profile of a set of extracellular signals. However, in each cell type, the identity of the input signals and the output genes is different. Thus, in development, cells need to perform essentially the same computations on varying inputs and output: a modularly varying goal. The solution found by evolution is a modular design where signal transduction pathways (such as mitogen-activated protein kinase cascades), which are common to many cell types, hook up to specific receptors and transcription factors that are cell type specific (45). This design allows simple rewiring of the same pathways to work with diverse inputs and outputs in different cell types (46). Over evolutionary time scales, this design allows the addition of new cell types without the need to evolve dedicated new pathways for each input and output (4, 18).

Modularity also has interesting consequences for gene duplication. Gene duplication can help to duplicate a module, but it cannot explain why a modular structure would persist if a more optimal nonmodular structure exists. For example, in a fixed-goal problem, we saw that an initially modular network rapidly lost modularity and approached one of many different nonmodular solutions, given pressure to minimize the number of components (Fig. 3). Our results help explain how modules are maintained, because of their benefit in a changing environment. Once modules are established, gene duplication can be effective in generating new modules with new functions.

It would be interesting to use the present approach to try to improve evolution of engineered systems. One might be able to use modularly varying goals in conjunction with other approaches mentioned in the Introduction to enhance the modularity and regularity of evolved designs.

In summary, this study presents a possible mechanism for spontaneous evolution of modularity and network motifs. It will be important to extend this study to understand how evolution could generate additional design features of biological systems (19).

We thank M. Elowitz, H. Lipson, R. Chen, R. Kishony, and the members of our laboratory for comments. We thank the National Institutes of Health, the Human Frontier Science Program, and Minerva for support.

1. Koza, J. (1992) *Genetic Programming: On the Programming of Computers by Means of Natural Selection* (MIT Press, Cambridge, MA).
2. Simon, H. A. (1996) *The Sciences of the Artificial* (MIT Press, Cambridge, MA).
3. Hartwell, L. H., Hopfield, J. J., Leibler, S. & Murray, A. W. (1999) *Nature* **402,** C47–C52.
4. Schlosser, G. & Wagner, G. (2004) *Modularity in Development and Evolution* (Chicago Univ. Press, Chicago).
5. Wagner, G. P. & Altenberg, L. (1996) *Evolution* **50,** 967–976.
6. Shen-Orr, S., Milo, R., Mangan, S. & Alon, U. (2002) *Nat. Genet.* **31,** 64–68.
7. Milo, R., Shen-Orr, S., Itzkovitz, S., Kashtan, N., Chklovskii, D. & Alon, U. (2002) *Science* **298,** 824–827.
8. Milo, R., Itzkovitz, S., Kashtan, N., Levitt, R., Shen-Orr, S., Ayzenshtat, I., Sheffer, M. & Alon, U. (2004) *Science* **303,** 1538–1542.
9. Conant, G. C. & Wagner, A. (2003) *Nat. Genet.* **34,** 264–266.
10. Mangan, S. & Alon, U. (2003) *Proc. Natl. Acad. Sci. USA* **100,** 11980–11985.
11. Kalir, S., Mangan, S. & Alon, U. (2005) *Mol. Syst. Biol.* **1**, msb4100010-E1–msb4100010-E6.
12. Lenski, R. E., Ofria, C., Pennock, R. T. & Adami, C. (2003) *Nature* **423,** 139–144.
13. Holland, J. (1975) *Adaptation in Natural and Artificial Systems* (Univ. of Michigan Press, Ann Arbor).
14. Goldberg, D. (1989) *Genetic Algorithms in Search, Optimization, and Machine Learning* (Addison–Wesley, Reading, MA).
15. Mitchell, M. (1996) *An Introduction to Genetic Algorithms* (MIT Press, Cambridge, MA).
16. Thompson, A. (1998) *Hardware Evolution: Automatic Design of Electronic Circuits in Reconfigurable Hardware by Artificial Evolution* (Springer, New York).
17. Raichman, N., Segev, R. & Ben-Jacob, E. (2003) *Physica A* **326,** 265–285.
18. Wilkins, A. (2002) *The Evolution of Developmental Pathways* (Sinauer, Sunderland, MA).
19. Alon, U. (2003) *Science* **301,** 1866–1867.
20. Vassilev, V., Job, D. & Miller, J. (2000) in *Proceedings of the Second NASA/DoD Workshop on Evolvable Hardware* (IEEE Computer Society, Washington, DC), pp. 151–160.
21. Hornby, G., Lipson, H. & Pollack, J. B. (2001) in *IEEE Conference on Robotics and Automation ICRA*, (IEEE Computer Society, Washington, DC), Vol. 4, pp. 4146–4151.
22. Angeline, P. & Pollack, J. (1993) in *Proceedings of the Second Annual Conference on Evolutionary Programming*, eds. Fogel, D. & Atmar, W. (Evolutionary Programming Society, La Jolla, CA), pp. 154–163.
23. Koza, J. (1994) *Genetic Programming II: Automatic Discovery of Reusable Programs* (MIT Press, Cambridge, MA).
24. Calabretta, R., Nolfi, S., Parisi, D. & Wagner, G. P. (1998) in *Proceedings of the Sixth International Conference on Artificial Life*, eds. Adami, C., Belew, R., Kitano, H. & Taylor, C. E. (MIT Press, Cambridge, MA), pp. 275–284.
25. Variano, E. A., McCoy, J. H. & Lipson, H. (2004) *Phys. Rev. Lett.* **92,** 188701.
26. Thompson, A. & Layzell, P. (2000) in *Evolvable Systems: From Biology to Hardware, Third International Conference, ICES 2000*, eds. Miller, J. F., Thompson, A., Thomson, P. & Fogarty, T. C. (Springer, New York), pp. 218–228.
27. Lipson, H., Pollack, J. B. & Suh, N. P. (2002) *Evolution* **56,** 1549–1556.
28. Vasconcelos, J. A., Ramirez, J. A., Takahashi, R. H. C. & Saldanha, R. R. (2001) *IEEE Trans. Magnetics* **37,** 3414–3418.
29. Yao, X. (1992) *A Review of Evolutionary Artificial Neural Networks* (Commonwealth Scientific and Industrial Research Organization, Victoria, Australia).
30. Haykin, S. (1999) *Neural Networks: A Comprehensive Foundation* (Macmillan, New York).
31. Newman, M. E. J. (2004) *Phys. Rev. E* **69,** 066133.
32. Newman, M. E. J. & Girvan, M. (2004) *Phys. Rev. E* **69,** 026113.
33. Guimera, R. & Nunes Amaral, L. A. (2005) *Nature* **433,** 895–900.
34. Itzkovitz, S., Levitt, R., Kashtan, N., Milo, R., Itzkovitz, M. & Alon, U. (2005) *Phys. Rev. E* **71,** 016127.
35. Miller, J. F., Thomson, P. & Fogarty, T. (1997) in *Genetic Algorithms and Evolution Strategies in Engineering and Computer Science: Recent Advancements and Industrial Applications*, eds. Quagliarella, D., Periaux, J., Poloni, C. & Winter, G. (Wiley, New York), pp. 103–131.
36. Kashtan, N., Itzkovitz, S., Milo, R. & Alon, U. (2004) *Phys. Rev. E* **70,** 031909.
37. Hansen, M. C., Yaclin, H. & Hayes, J. P. (1999) *IEEE Design Test* **16,** 72–80.
38. Ripley, B. D. (1996) *Pattern Recognition and Neural Networks* (Cambridge Univ. Press, Cambridge, U.K.).
39. Kashtan, N., Itzkovitz, S., Milo, R. & Alon, U. (2004) *Bioinformatics* **20,** 1746–1758.
40. Bray, D. (1995) *Nature* **376,** 307–312.
41. Bhalla, U. S. & Iyengar, R. (1999) *Science* **283,** 381–387.
42. Itzkovitz, S., Milo, R., Kashtan, N., Ziv, G. & Alon, U. (2003) *Phys. Rev. E* **68,** 026127.
43. Berg, H. C. & Brown, D. A. (1972) *Nature* **239,** 500–504.
44. Eisenbach, M., Lengeler, J. W., Varon, M., Gutnick, D., Meili, R., Firtel, R. A., Segall, J. E., Omann, G. M., Tamada, A. & Murakami, F. (2004) *Chemotaxis* (World Scientific, Teaneck, NJ).
45. Schaeffer, H. J. & Weber, M. J. (1999) *Mol. Cell. Biol.* **19,** 2435–2444.
46. Park, S. H., Zarrinpar, A. & Lim, W. A. (2003) *Science* **299,** 1061–1064.