Proceedings of DETC'01
2001 ASME Design Engineering Technical Conferences
September 9-12, 2001, Pittsburgh, Pennsylvania, USA

DETC2001/DAC-21099

PROMOTING MODULARITY IN EVOLUTIONARY DESIGN

Hod Lipson[1,2] lipson@cs.brandeis.edu
Jordan B. Pollack[1] pollack@cs.brandeis.edu
Nam P. Suh[2] npsuh@mit.edu

[1]Computer Science Department, Brandeis University, Waltham MA 02454, USA
[2]Mechanical Engineering Department, MIT, Cambridge MA 02139, USA

ABSTRACT

Evolutionary design systems apply principles inspired from biological evolution to automate machine design. These systems have been shown to generate simple designs for simple tasks – but their practical ability to scale up to higher complexities remains questioned. One of the keys to accomplishing higher-level evolutionary design is the ability of the process to identify and reuse knowledge discovered at lower levels, thus scaling its search capacity. One way to capture this knowledge is in the form of reusable building blocks – modules. In this paper we define modularity and discuss several approaches to promoting modularity in evolutionary design systems. In particular, we propose a new mechanism that can enhance modularization. This mechanism is based on the observation that designs that exhibit modularity have higher adaptability and consequently have better survival rates under changing requirements. Contrary to other techniques, this is a weak (indirect) formulation that does not require representation of partial solutions or definition of a genotype from which a design is developed. We demonstrate this principle on an abstract general design problem on which modularity can be statistically quantified.

INTRODUCTION

Research in design automation can be broadly categorized into two schools of thought: the first approach attempt to emulate the top-down decomposition process that characterizes human engineering. The alternative approach attempts to emulate the bottom-up self-organizing process that is characteristic of natural evolution. Both approaches have their merits and shortcomings; primarily, the top-down approach is relatively efficient but requires much knowledge, while the bottom-up approach is based on inefficient trial-and-error, but requires little knowledge, and even generates it. One of the challenges in design automation is automatically capturing this bottom-up generated knowledge and reusing it in a top-down process. This paper addresses this challenge by focusing on the self-organization of modularity in evolutionary design systems.

Biological evolution is characterized as a process applying to a population of individuals, that is subject to selective replication with inheritance and some variation (Smith and Szathmary, 1999). Evolutionary design systems use the same principles of biological evolution to achieve machine design, yet add a teleological facet in which there is a target to the evolution – the functional requirements specified by the designer. Candidate designs in a population are thus still replicated under inheritance and variation, but selection is governed by an external design criteria. After a number of generations – depending primarily on the complexity of the required design – the selective evolutionary process may breed an acceptable design.

There are still many poorly understood factors that determine the success of this process. The choice of starting conditions, primitive building blocks, operators, selection and replacement mechanisms, fidelity of physical simulation and existence of gradient and path to a solution all affect the ability of the process to find solutions, and introduce biases of their own.

Genetic algorithms (GAs) – a subset of evolutionary computation involving mutation and crossover in a population of fixed length bit strings – have been applied for several decades in many engineering problems as an optimization technique for a fixed set of parameters. But *open-ended* evolutionary design systems, in which the process is allowed to add more and more building blocks and parameters, seem particularly adequate to design problems requiring *synthesis*. Such open-ended evolutionary design systems have been demonstrated for a variety of simple design problems, including structures, mechanisms, software, optics, robotics, control, and many others (for overviews see, for example,

Copyright © 2001 by ASME

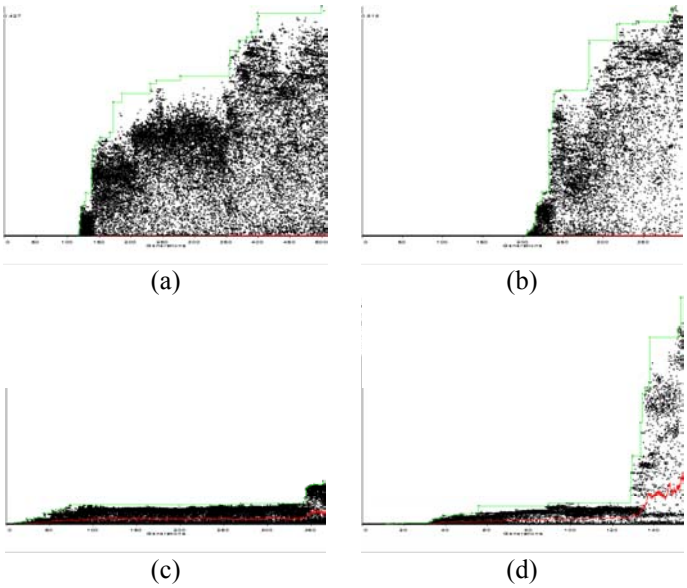


Figure 1. Progress of our typical evolutionary design run. The abscissa represents evolutionary time (generations) and the ordinate measures fitness. Each point in the scatter plot represents one candidate design. In general, a logarithmic slowdown in progress can be observed, characterized by longer and longer durations between successive step-jumps in the fitness (a,b). Occasionally, however, progress is made after long stagnation (c,d).

Bently, 1999; Husbands *et al*, 1998). Yet these accomplishments remain simple compared to what teams of human engineers can design and what nature has produced. Evolutionary design approach is often criticized as scaling badly when challenged with design requirements of higher complexity.

The apparent difficulty in scaling evolutionary design problems to higher complexities is that the design space is exponentially large, because there is an exponentially increasing number of ways a linearly increasing set of components can be assembled. Consequently, design approaches that are based on assembly at a fixed level are hopelessly intractable.

Indeed, in our own experiments in open-ended evolutionary design of mechanisms and controllers for locomotion (Lipson and Pollack, 2000), the process has produced physically viable mechanisms. However, the design progress seemed to have reached a plateau after a while. Figures 1a and 1b show the progress of a typical run. The abscissa represents evolutionary time (generations) and the ordinate measures fitness (net movement on a horizontal plane). Each point in the scatter plot represents one candidate design. In general, after an initial period of drift with zero fitness we observe rapid growth followed by a logarithmic slowdown in progress[1], characterized by longer and longer durations[2] between successive step-improvements in the fitness.

We thus hypothesize that the primary challenge in evolutionary design research is that of allowing the process to identify and re-use discovered knowledge. This reuse would, in turn, lead to acceleration in the discovery process, leading in turn to higher level of search, and so forth. This scaling in knowledge and unit of construction – hierarchical modularity – is observable in the history of engineering (Ulrich and Tung, 1991; Huang and Kusiak, 1998), economic organization (Langlois, 2001) and in nature (Hartwell *et al*, 1999). Theoretical analysis reveals that allowing an evolutionary process to repeatedly aggregate low-level building blocks into higher-level groups in a hierarchical fashion, potentially transforms the design problem from exponential complexity to polynomial complexity under certain conditions (Watson, 2001). The challenge then becomes the identification of the correct set of modules.

This paper proposes one approach that allows a set of building blocks to self-organize, or decompose, into potentially useful modules.

MODULARITY

We propose the following definition:

Modularity is the separability of a design into units that perform independently. Since performance can only be defined with respect to some function, it follows that modularity is only defined with respect to functional requirements. Modularity of any unit can be measured as the ratio of function-related information that moves within the unit to the function-related information communicated across the boundary of the unit. High modularity suggests separability into units that have sparse interface compared to their internal workings.

Note that the actual *re-use* of a module gives rise to *regularity*. Regularity is any pattern in the composition of a design. Duplicated structures, symmetry, self-similarity, smoothness, hierarchy, and other forms of correlation are all examples of patterns of regularity. Regularity reduces the information needed to describe a design, and can be measured in information entropy or in the compressibility of a description of a design. Modularity and regularity are independent properties: Consider a design that is composed of several units, each performs a different sub-function and in a different way. Such a design may be modular, but has no regularity because each unit is different.

[1] However, note that because of the stochastic nature of the process, it is hard to determine definitely whether progress has actually halted, and improvements may still occur after long periods of apparent stagnation (Figures 1c and 1d).

[2] This lingering is amplified by the fact that evaluation time, or duration of a generation (in simulation or in physical reality) also increases as solutions become more complex

Copyright © 2001 by ASME

Existing approaches for modularity

The need to promote modularity as a key to scalability of evolutionary design is well recognized in the literature. Briefly, one can observe two main approaches to promoting modularity:

- **Promotion of modularity by enforcement of regularity.** Because of the indirect link between regularity and modularity, the enforcement of regular structures might lead to modularity. Regularity is typically promoted using genotype that can encode for duplication or patterns in the phenotype. Examples are generative grammars like L-systems (Hornby *et al*, 2001), cellular encoding (Gruau and Quatramaran, 1997) and tree architectures (Genetic programming, Koza 1994).
- **Promotion of modularity by partial specification.** Here modules are represented explicitly. This is a direct method in which the unit of selection is not a complete solution but part of it, that is represented independently. A module is evaluated in context of other modules, together forming a complete design. There are several approaches to representation of partial designs and to decision of when assemblies of building blocks should be fixed as a new higher-level module. Examples include variable length chromosomes (Messgy GA, Goldberg *et al*, 1989), automatically defined functions (ADFs, Koza, 1994), Module libraries (GLib, Angeline and Pollack, 1993), partial commitment (SEAM, Watson and Pollack, 1999), and modular inductive learning (MIL, Juillé, 1999).

EVOLUTION OF MODULARITY

Proposition

In this paper we propose a new mechanism that can generate modularization. This mechanism is based on the known observation that designs that exhibit modularity have higher adaptability and consequently have better survival rates under changing requirements. Consequently, modularity can evolve as response to changing environment, or conversely, we hypothesize, **modularity can be promoted by actively changing the environments or requirements to be met over time**.

Contrary to cited methods above, this approach for promotion of modularity does not rely on the ability of the substrate to represent partial solutions or have a genotype-phenotype developmental scheme. Instead, it provides a weak (indirect) formulation that applies to complete solutions. It can therefore also operate top-down by gradually decomposing a working but coupled solution into modules.

Model

We demonstrate this principle on an abstract general representation that describes design as a transformation of resources into functionality. Although we realize that not all designs can be abstracted in this way, and that true design problems are much more elaborate and nonlinear, we argue that this abstraction captures basic connectivity aspects that are necessary for the statistical quantification of the modularization process (DSM, Ulrich and Eppinger, 2000).

Let an individual design candidate be modeled as a transfer matrix **A** that is required to transform a vector of given building blocks **B** into some vector **F** representing a set of arbitrary functional requirements:

$$F = A \times B \qquad (1)$$

The figure of merit *f(***A***)* of the design candidate **A** is how well it satisfies the above vector equation, given a vector of requirements **F** and a vector of building blocks **B**. This figure of merit (or *fitness*, in evolutionary terms) can be quantified as the magnitude of deviation |**F-AB**| or as the Hamming distance between **F** and **AB**.

The representation of a design as a dependency matrix conforms with well established systems engineering methods (Warfield, 1973; Steward, 1981) and conventions such as Design Structure Matrices (DSM, Ulrich and Eppinger, 2000), as well as matrix domain mappings in the design process (Quality Function Deployment, Akao, 1990; Axiomatic Design, Suh, 1990). In biological systems too, an individual creature can be modeled as transforming a set of resources available in its environment into some performance measurable against a set of arbitrary requirements for "survival".

Quantifying modularity

Based on the definitions above, we quantify modularity as inversely proportional to the amount of coupling in the system. Binary matrices are often used to describe dependencies among system components (Zero=independent, Nonzero=dependent), and methods for improving design process and for identifying modules are based on reordering this matrix into near-block-diagonal form (Warfield, 1973; Steward, 1981). In this form, blocks on the diagonal represent modules, and nonzero

$$\begin{bmatrix} -1 \\ +1 \\ -1 \\ -1 \\ +1 \\ +1 \\ +1 \\ -1 \end{bmatrix} \stackrel{?}{=} \begin{bmatrix} +0.1 & 0 & -0.6 & -0.1 & 0 & 0 & -0.4 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1.0 & +0.8 & 0 \\ -0.2 & 0 & +1.0 & 0 & 0 & 0 & +0.9 & 0 \\ 0 & -0.9 & 0 & 0 & +0.1 & 0 & 0 & 0 \\ -0.7 & 0 & 0 & +0.3 & 0 & 0 & 0 & -0.3 \\ 0 & 0 & 0 & 0 & 0 & 0 & -0.5 & 0 \\ 0 & 0 & +0.6 & -0.2 & 0 & 0 & -0.2 & 0 \\ +0.1 & 0 & 0 & 0 & 0 & 0 & -0.8 & 0 \end{bmatrix} \times \begin{bmatrix} +1 \\ -1 \\ +1 \\ +1 \\ -1 \\ +1 \\ +1 \\ -1 \end{bmatrix}$$

F — An arbitrary set of functional requirements or survival requirements

A — A design candidate or individual trying to transform building blocks to meet requirements

B — An arbitrary set of building blocks or resources

Figure 2. A general design problem, where a matrix **A** is required to transform a set of building blocks **B** to meet a set of requirements **F**. There are many **A**'s (designs) that solve this equation, and many more that *nearly* solve it. The matrix above, for example, is not doing so well. Some of the matrices are more coupled (less modular) than others. Elements of **F** and **B** are random integers ±1, and elements of **A** evolve in the real-valued range of [–1, +1].

Copyright © 2001 by ASME

elements outside the blocks represent interactions between modules. In a system represented as a *binary* matrix **A**, coupling amounts to the percentage of non-zero elements.

Real-valued system matrices are used instead of binary matrices to quantify degree of dependency or cost of iteration. Element signs denote positive or negative feedback. Such matrices can also be reordered to identify blocks on the diagonal and improve design process (Rogers and Bloebaum, 1994).

In both binary and real valued matrices, we define coupling C_A of the matrix **A** simply as the sum of magnitude of elements

$$C_A = \frac{\sum_{i,j=1}^{N} |A_{i,j}|}{N^2} \qquad (2)$$

where N is the size of the square matrix **A**. We assume all elements are dimensionless connectivity numbers the rage of –1 to +1, denoting anything from no connectivity to full positive or negative connectivity. We performed our experiments on both binary and real-valued matrices, with equally significant observable effects.

When a matrix is reordered into block-diagonal form, each block on the diagonal represents to a module. Ultimately, when the matrix is diagonal, each row corresponds to a module.

Experimental setup

A basic evolutionary design process consists of a population of candidate designs, selection criteria, duplication and variation process and replacement criteria. In a set of experiments, we used a simple steady-state genetic algorithm (see Mitchell, 1996) to try to find matrices with high fitness for a given vector of requirements **F** and a vector of building blocks **B**. We used matrix size N=8, with elements of both **F** and **B** set randomly to ±1. The only genetic operators used were mutations.

We used a population of either binary or real-valued matrices. When binary, population size was set to 500, each initialized randomly to either {–1,0,+1}. Mutation operators mutated a randomly selected element of the matrix to either {–1,0,+1}. When using matrices, population size was 200, and each matrix was initialized with elements randomly selected uniformly in the range [–1,+1]. Mutation operators mutated a randomly selected element of the matrix by an increment in the range [–0.1,+0.1], while ensuring the mutated value remains in the range [–1,+1]. A sample real-valued design candidate is shown in Figure 2.

Starting with the randomly initialized population, the steady-state evolutionary process selected an individual, mutated it and replaced it into the population by overwriting an existing inversely selected individual. We used a normalized Boltzmann selection criterion (see Mitchell, 1996) to keep the selection pressure constant, and used an inverse Boltzmann selection for replacement.

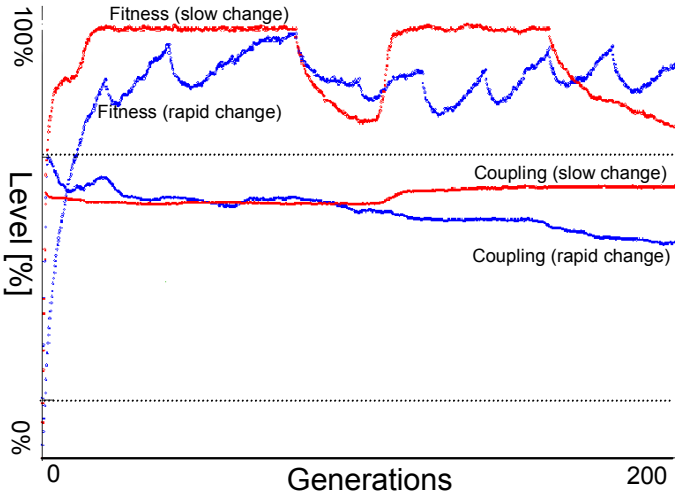


Figure 3. Two typical evolutionary runs subject to different environmental change rates. (a) rapid change: environment changes every 20 generations, (b) slow change: environment changes every 80 generations. For each of the two cases there are two curves: the top shows fitness and the lower shows coupling. Note how under slow changing environment solutions have higher average fitness but higher coupling, whereas under faster changing environment solutions have lower average fitness but decreasing coupling. Also note that when environment changes are introduces, population with higher coupling suffers a harder blow.

Environment Change

To simulate the change in the environment, random elements of the vector **B** occasionally flipped sign. The rate at which such random flipping occurred was the main parameter varied in the experiment, starting with the control experiment

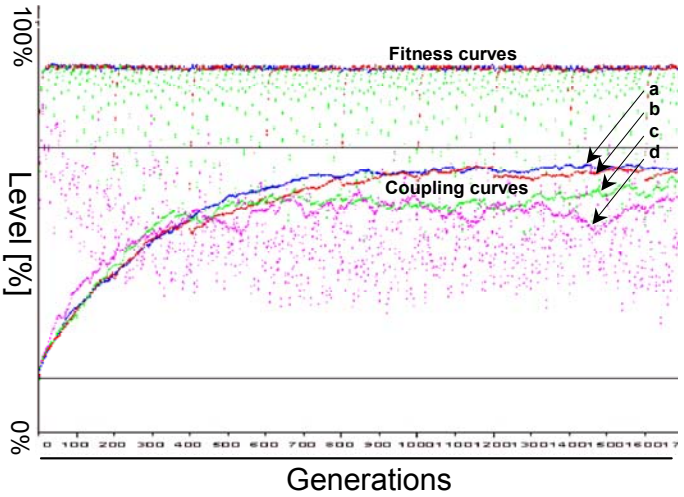


Figure 4. Four curves for different environmental change rates. (a) No change: (b) change every 500 generations, (c) change every 50 generations, and (d) change every 5 generations. Each curve shows average over 10 experiments with binary matrices, each with population of 500 matrices over 2,000 generations. The matrices were all initialized to all-zero state, and so the solutions process represent a bottom-up composition rather than top-down decomposition. Separation of the four coupling curves according to change rates is already visible after 1,000 iterations. All populations reach maximum fitness.

Copyright © 2001 by ASME

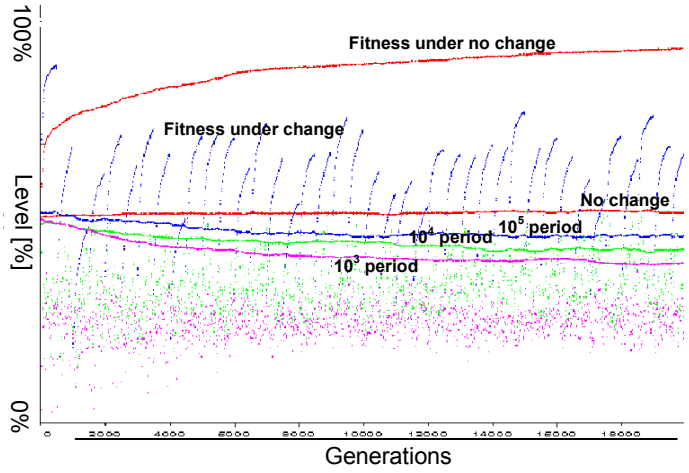


Figure 5. Four curves for different environmental change rates. (a) No change: (b) change every 500 generations, (c) change every 50 generations, and (d) change every 5 generations. Each curve shows average over 100 experiments with real-valued matrices. Other curves show fitnesses: Top curve shows fitness of population (a), and fitnesses of other three populations sawtooths up and down too rapidly to create a continuous curve.

in which the vector **B** was fixed (no change), to the other extreme where at least one building block flipped sign almost every generation.

RESULTS

In the above experiments, we measured the average amount of coupling and the average fitness of the population of candidate designs over the evolutionary time. Experiments were left running for 4×10^6 iterations where progress appears to have stabilized. This is approximately equivalent to 20,000 generations of a population of 200 individuals. We show below some typical runs, as well as statistical analysis across 100 experiments

SINGLE EXPERIMENT OF BINARY MATRICES

Figure 3 shows two typical evolutionary runs, each subject to different environmental changing rates. Here the population consisted of 500 binary individuals. One population is subject to change introduced every 20 generations, whereas another population is subjected to change every 80 generations. The top dashed line shows the mutation bias of 66% non-zeros, because this is a binary matrix with each element either –1, 0, or +1. The lower dashed line shows the optimal coupling of a diagonal matrix.

For each of the two cases there are two curves: the top two curves show fitness of the two populations, and the lower curves show coupling of the two populations over time. Under slow changing environment solutions have higher average fitness but higher coupling, whereas under faster changing environment solutions have lower average fitness but lower coupling. Also note that when environment changes are introduces, the population with the higher coupling suffers a harder blow.

Figure 4 shows four curves for different environmental change rates. (a) No change: (b) change every 500 generations, (c) change every 50 generations, and (d) change every 5 generations. Each curve shows average over 10 experiments with binary matrices, each with population of 500 matrices over 2,000 generations. The matrices were all initialized to all-zero state, and so the solutions process represent a bottom-up composition rather than top-down decomposition. Separation of the four coupling curves according to change rates is already visible after 1,000 iterations. All populations reach maximum fitness

Copyright © 2001 by ASME

Averaged experiments with real valued matrices

We repeated the experiments above with real-valued matrices for four different change rates: every 5, 50, and 500 generations, and no change (control). This time we ran each experiment 100 times, so that average and variance could be estimated more precisely, and let the experiment continue for 20,000 generations.

Figure 5 shows four horizontal curves, representing average coupling over time of these four populations. The top horizontal curve corresponds to non-changing (control) environment, and indeed has a coupling level corresponding roughly to the 50% mutation bias of a real-valued matrix. The three curves below it correspond to increasingly higher change rates, respectively. Variances of these curves are ±4%. The curves appear to reach asymptotes after approximately 10,000 generations.

The other curves in the background show fitnesses: The topmost curve shows fitness of control population, and fitnesses of other three populations sawtooths up and down too rapidly to create a continuous curve at this scale.

Statistical analysis

In order to support the hypothesis that internal coupling decreases as rate of change of the environment increases, we summarized these results in a plot of internal coupling versus environment change rate, shown in Figure 6. This plot shows a clear linear-log correlation. Each point in the plot is averaged for 100 experiments, each of 20,000 generations of a population of 200 real-valued matrices.

Statistical significance of differences in coupling among pairs of populations subject to difference change rates was measured using the standard Wilcoxon-Mann-Whitney test for distribution independent comparison of populations (Weerahandi, 1994). This test was used since the populations in questions cannot be assumed to have a normal distribution, and so a distribution-free test is required. Table I compares pairs of the four populations against the null hypothesis that coupling is not reversely correlated to change rate. Note that period of change is the reciprocate of rate of change.

CONCLUSIONS

One of the keys to the scalability of evolutionary design systems to more complex problems is the ability of the process to identify and re-use knowledge in the form of functional modules. In this paper we proposed a new mechanism that can enhance modularization, based on the observation that designs that exhibit modularity have higher adaptability and consequently have better survival rates under changing requirements. We have demonstrated this principle on an abstract general design problem based on dependency-matrix representation common in system engineering. In our simulated experiments, a clear correlation between rate of change of the environment and amount of coupling in the dependency matrix is observable. This result was also observed in many other variants of these experiments which we did not report here

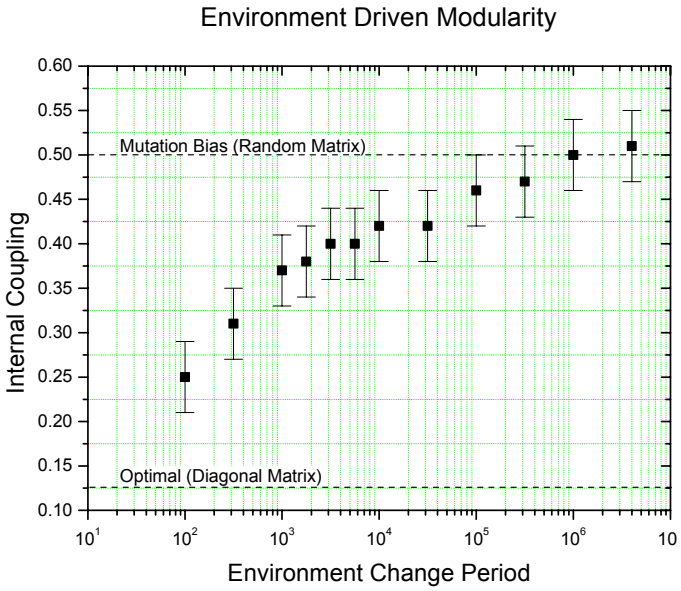


Figure 6. Environment driven modularity: Internal coupling versus environment change rate shows a linear-log correlation. Each point in the plot is averaged for 100 experiments, each of 20,000 generations of a population of 200 real-valued matrices. Internal coupling is defined by Eq. (2).

Table I

Statistical significance* of differences in coupling among pairs of populations subject to difference change rates after **10,000** generations

Population A		Population B		H_0	p	Conf.
Period	Coupling**	Period	Coupling**			
10^6	μ=0.51 σ=0.04	10^5	μ=0.46 σ=0.04	$C_A>C_B$	8.11	>99.99%
10^6	μ=0.51 σ=0.04	10^4	μ=0.41 σ=0.04	$C_A>C_B$	11.58	>99.99%
10^6	μ=0.51 σ=0.04	10^3	μ=0.37 σ=0.04	$C_A>C_B$	12.03	>99.99%
10^5	μ=46 σ=0.04	10^4	μ=0.41 σ=0.04	$C_A>C_B$	7.97	>99.99%
10^5	μ=0.46 σ=0.04	10^3	μ=0.37 σ=0.038	$C_A>C_B$	10.76	>99.99%
10^4	μ=0.41 σ=0.04	10^3	μ=0.37 σ=0.04	$C_A>C_B$	6.19	>99.99%

* Using distribution free Wilcoxon test (Weerahandi, 1994)
** Figures based on 100 experiments for each population.

(different matrix sizes, selection pressures and mutation rates), and we anticipate that additional experiments with other genetic operators, other induced changes, and on more complex substrates would enhance these results.

Contrary to other techniques for modularization in evolution, this is a weak (indirect) formulation that does not require representation of partial solutions or definition of a genotype from which a design is developed. However, while this method promotes modularity, an additional mechanism is necessary to actually re-use the revealed modules in a hierarchy, such as a grammar (Hornby *et al*, 2001) or tree structure (Koza, 1994). Further more, an additional mechanism is required to allow the abstraction of the behavior of a module,

Copyright © 2001 by ASME

so that simulation complexity will scale and remain tractable too.

Implications for evolutionary design

Our results indicate that evolution of designs to meet a fixed fitness criterion in a fixed environment will tend to yield ad-hoc solutions, which are perhaps satisfactory in their performance, but do not have a modular, regular, or "elegant" structure. However, if the fitness criterion and/or environment would change over time, then the modularity and regularity of the solution would be enhanced.

Consequently, we predict that modularity in evolved designs can be promoted by introduction of variable environment and/or variable functional requirements.

These results may also shed some light on the evolution of modularity in nature.

REFERENCES

Akao Y., Ed. (1990). *Quality Function Deployment*, Productivity Press, Cambridge, MA, USA.

Angeline, P. J. and Pollack, J. B. (1993). "Evolutionary Module Acquisition". Second Evolutionary Programming Meeting, 154-163

Bentley P. Ed. (1999) *Evolutionary Design by Computers*. Morgan Kaufmann, San Francisco.

Goldberg, DE, Korb, B, & Deb K, 1989 "Messy Genetic algorithms: Motivation, Analysis and First Results", Complex Systems 1989, 3, 493-530.

Gruau F., Quatramaran K., (1997), "Cellular Encoding for interactive evolutionary robotics", in Husbands and Harvey (Eds.), 4th Euro. Conf. on artificial life, MIT Press

Hartwell, L.H., Hopfield, J.J., Leibler, S. & Murray, A.W. (1999) From molecular to modular cell biology. Nature 402, C47-52

Hornby G., Lipson H., Pollack J. B., (2000) "Generative design system for modular physical robots", IEEE Conference on Robotics and Automation, 2001, Seoul, Korea (to appear)

Huang C.C. and Kusiak A., (1998) Modularity in Design of Products and Systems, *IEEE Transactions on Systems, Man, and Cybernetics*, Part A, Vol. 28, No. 1, 66-77

Juillé, H. (1999). *Methods for Statistical Inference: Extending the Evolutionary Computation Paradigm.* Doctoral Dissertation, Brandeis University, Department of Computer Science, May 1999

Koza J. R., (1994) *Genetic Programming II: Automatic Discovery of Reusable Programs* (Complex Adaptive Systems), MIT Press

Langlois R. N. (2001), "Modularity in Technology and Organization," *Journal of Economic Behavior and Organization*, in press

Lipson H., Pollack J. B., (2000), "Automatic design and Manufacture of Robotic Lifeforms", Nature 406, pp. 974-978

Mitchell M. (1996) *An Introduction to Genetic Algorithms*, MIT Press, Cambridge MA

Rogers, J. L. and Bloebaum, C., (1994) Ordering Design Tasks Based on Coupling Strengths, 5th AIAA/USAF/NASA/ ISSMO Symposium on Multidisciplinary Analysis and Optimization, Panama City, Florida, September 7-9

Smith J.M., Szathmary E. (1999), The Origins of Life, Oxford Univ. Press.

Steward, D. V., (1981) "The Design Structure System: A Method for Managing the Design of Complex Systems" IEEE Transactions on Engineering Management, vol. 28, pp. 71-74

Suh N.P. (1990) *The Principles of Design*, Oxford University Press, Oxford UK.

Ulrich K.T. and Eppinger S.D., (2000) *Product Design and Development*, 2nd Ed., McGraw-Hill, New York

Ulrich K.T. and Tung K., (1991) "Fundamentals of product modularity", *Issues in Design/Manufacture Integration,* Sharon A. (Ed.), ASME: NY., pp 73-79

Warfield, J. N., (1973) "Binary Matrices in System Modeling" IEEE Transactions on Systems, Man, and Cybernetics, vol. 3, pp. 441-449

Watson, R.A. and Pollack, J.B. (1999). "Incremental Commitment in Genetic Algorithms". Proceedings of 1999 Genetic and Evolutionary Computation Conference (GECCO 99). Banzhaf, Daida, Eiben, Garzon, Honavar, Jakiela, Smith, eds., Morgan Kauffmann, pp.710-717.

Watson, R.A., (2001) "Analysis of Recombinative Algorithms on a Non-Separable Building-Block Problem", *Foundations of Genetic Algorithms*, Volume 6, proceedings of FOGA VI, Charlottesville, VA, July 21-23, 2000, Edited by Worthy N. Martin and William M. Spears, Morgan Kaufmann, to appear.

Weerahandi S., (1994) *Exact Statistical Methods for Data Analysis*, Springer Verlag, Berlin

Copyright © 2001 by ASME