Mechanisms for Complex Systems Engineering through Artificial Development

Taras Kowaliw and Wolfgang Banzhaf

Abstract

In this chapter, we argue that artificial development is an appropriate means of approaching complex systems engineering. Artificial development works via the inclusion of mechanisms that enhance the evolvability of a design space. Two of these mechanisms, *regularities* and *adaptive feedback* with the environment, are discussed. We concentrate on the less explored of the two: adaptive feedback. A concrete example is presented and applied to a simple artificial problem resembling vasculogenesis. It is shown that the use of a local feedback function substantively improves the efficacy of a machine learner on the problem. Further, inclusion of this adaptive feedback eliminates the sensitivity of the machine learner to a system parameter previously shown to correspond to problem hardness.

1 Introduction

Future engineering tasks will require dramatically larger scales of design. To meet these challenges, we need to incorporate techniques from complex systems science into engineering. One such technique, which holds promise for the creation of large and robust designs, is *artificial development* (AD). It concerns the inclusion of a mid-step between the representation and a pattern in an automated design task, inspired by biological embryogenesis. Through the emulation of this biological phenomenon, desirable properties of biological organisms can be included into

T. Kowaliw
Institut des Systèmes Complexes - Paris Île-de-France, Centre national de la recherche scientifique, 57-59 rue Lhomond, 75005, Paris, France. e-mail: `taras@kowaliw.ca`

W. Banzhaf
Department of Computer Science, Memorial University, St. John's, Newfoundland and Labrador, A1B 3X5, Canada. e-mail: `banzhaf@cs.mun.ca`

the machine-learned designs, resulting in increased evolvability in the underlying search space.

In this chapter, we concentrate on two developmental mechanisms by which increased evolvability can be achieved in an evolutionary design task. We examine abstractions of these mechanisms as search strategies, and discuss their function and advantages. The first strategy, *regularities*, is considered to be simple gradient-based functions and is reviewed here. The second, *adaptive feedback*, seen as a means of adding information to a developing phenotype from that organism's environment, is discussed in more detail, as we believe it to be the newer and less explored of the two.

1.1 Complex Systems Engineering

It has been argued that complex systems engineering requires biological development-like conceptualization in order to make progress, that is, the traditional state-based approach in engineering has reached its limits, and the principles underlying complex systems—self-organization, nonlinearity, and adaptation—must be accommodated in new engineering processes [2]. It is our belief that developmental mechanisms will play an important role in the heightened evolvability of these problem spaces.

2 Artificial Development

2.1 Biological Embryogenesis

Organismal development is a means by which complex and robust designs can be produced by natural evolution, and is properly viewed as a means of increasing evolvability in phenotypic space [13]. Paradoxically, it often does so by constraining the space of possible designs searched by evolution. These constraints limit the potential phenotypic configurations achievable from genotypic configurations, ideally creating a more evolvable space of phenotypes. As Gould notes, there exist "positive aspect[s] of constraint as fruitful channeling, along lines of favorable variation that can accelerate or enhance the work of natural selection" [10].

This viewpoint has been further developed by Arthur [1], who refers to any change in ontogeny caused by mutation of developmental genes as developmental reprogramming, as opposed to other forces affecting ontogeny, such as environment and epigenetics. Due to the genotype-to-phenotype map, phenotypic traits have an "ease of variance" associated with developmental reprogramming. For Arthur, traits hard to vary are subject to developmental "constraints", whereas traits easy to vary are subject to developmental "drives". Newman *et al.* [31] list several candidates for these drives, and argue that to understand organismal form, one must understand the "originating physical and otherwise non-programmed determinants of multicellular form". They suggest that these determinants influence much of the adaptivity and morphological plasticity seen in modern organisms, and moreover that the generative entrenchment of these morphological motifs has been the major role of molecular evolution over the past half billion years.

Here we also mention another mechanism believed to increase evolvability in biological embryogenesis: local cellular adaptation to environmental stimuli. A simple example of a means by which environmental feedback might help with the evolvability of an organism can be seen in so-called "amphibious plants" (e.g. *Myriophyllum*). In this case, leaves grow both underwater and in dry air, where the presence of water in a meristem's immediate neighborhood will influence the type of leaf growth [4]. Hence, for the Myriophyllum, it is unnecessary to encode the height of the water in the genome, the growth of the plant will naturally select an appropriate height at which to change leaf types. A lengthy list of examples of organisms exploiting environmental information during development is available in [9].

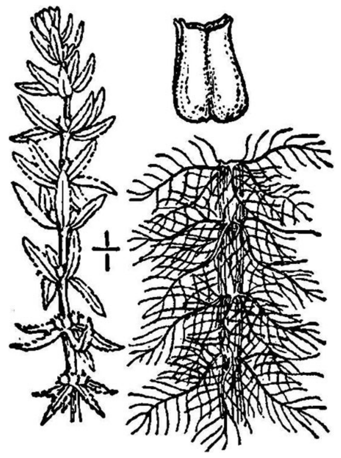

Fig. 1 An illustration of the different leaf types in *Myriophyllum*. Image from the USDA-NRCS PLANTS Database [3].

There are also numerous examples of functional self-adaptation to internal stimuli, ones which seem to use indicators of a developing organism's ultimate fitness. A classic example is the remarkable plasticity found in the visual cortex, where the stimuli (or lack thereof) presented to the brain during early development strongly influence brain architecture [15]. Another example is cellular response to physical stress: some cells are capable of specific differentiation via cues from the stiffness of the extracellular matrix [30]. We note with some interest that this factor, mechanical stress, recovers an interesting and global property of the overall phenotype otherwise unavailable to the locally minded cell.

2.2 *Artificial Development*

There is much interest at present in the use of development in evolutionary computation (EC). Artificial development (sometimes called artificial embryogeny, artificial ontogeny, computational embryogeny, and so on) is used to describe a developmental phase in artificial evolution, that is, an indirect mapping between the genotype (representation) and phenotype (evaluated organism). Usually, this mapping incorporates a notion of time or some other notion of ordered processes. AD is inspired by, but does not necessarily resemble, biological embryogenesis. The "typical" setup for an AD system is shown in Fig. 2.

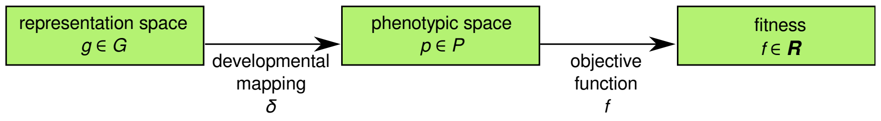

Fig. 2 An illustration of the typical AD process: a representation g (genome) is transformed via a developmental mapping δ into a pattern p (phenotype), which is then, in turn, evaluated by a fitness function f.

The first computational models of embryogenesis include chemical diffusion by Turing [38] and simple automata by Lindenmayer [25]. Perhaps the best known model, however, is an exceedingly simple proof of concept using recursive systems: Dawkins' "biomorphs", a simple user-guided evolutionary strategy that controls the development of stick-based trees [6]. With biomorphs, the mapping between the genotype and phenotype imposes biases on the space of generated patterns: the recursive structures show repetitions and self-similarities that suggest life-like forms to users, no small part of the popularity of the system.

A primary interest in AD is in the creation of very large designs, ones which cannot be evolved through direct encodings. This is sometimes phrased as the generation of designs through "complexification", and is related to the notion of representations that "scale-up". Indeed, the capacity for AD to generate designs too large for evolution via bijective encodings (and the converse, AD's occasional failure to recreate direct encodings' results) has been demonstrated in different domains [8, 11, 12]. Several other desirable properties have been shown to be plausible in AD models, such as the capacity for self-organization, including self-repair [8, 29], and the generation of not only the final organism, but also a plan for assembly [32].

There are many potential mechanisms, borrowed by analogy from biology, by which AD might achieve these aims. Some of these mechanisms include: cell physics (which has, to some extent, been explored as a means of engineering designs [28]), regularities, and local adaptation to environmental information. Below, we consider in some detail cellular growth models for AD, that is, the use of Cellular Automata (CAs) and CA-like models on a lattice as a means of producing designs (for an introduction to CAs, see [16]). In the next section, we outline an exemplar of cellular growth and describe its application to structural design.

2.3 Truss Design

Structural design is a popular application in both evolutionary computation generally, and in AD. Here we concentrate specifically on truss models. Trusses are simple but realistic models of structural form usually employed in the initial plans for buildings, bridges, and numerous other structures. For our purposes, we assume that they consist of beams and joints, where the beam material and structure is constant. Trusses have two key requirements: to be stable, that is, not to change their shape drastically, and to support external load, that is, to distribute the force resulting from external weight efficiently. Given some truss and some collection of external forces, we may calculate the forces on the beams and the displacement of the joints via a system of linear equations: evaluation of a truss is hence a matter of matrix inversion, which takes time $O(n^3)$ (or slightly less via LU decomposition). The design of topology of trusses (as opposed to, say, the optimization of materials, thickness, member lengths, etc.) is referred to as *topological optimization*. It has been addressed multiple times in evolutionary computation (for a review, see [18]). AD in particular has been applied to trusses or truss-like analysis by several researchers [7, 17, 36, 40].

Kowaliw *et al.* approached this domain of application using *Deva*, a CA-like model of growth. The Deva system begins with a single differentiated cell in the centre of a lattice. Each cell in the lattice utilizes an identical CA-like rule set. At each discrete time step, a cell activates one of its rules, executing a specified action. These actions include cell-based functions, such as division, elongation, and specialization. Overall growth is constrained by a system-specified limit on the amount of resources available: once development fails to produce additional changes (guaranteed to occur due to finite resources), it is considered complete. Through this means, Deva develops an image on the lattice, consisting of a collection of colored cells.

These images are then converted to a truss structure through a custom interpretation. Each cell color corresponds to either a cell type or an elongation of an adjacent cell type. The cell types consist of a joint (or absence thereof) and some collection of beams extending from this central joint. Through this mapping, it is possible to interpret any Deva-produced image as a truss. These trusses are further post-processed to remove obviously redundant components. This mapping process is illustrated in Fig. 3.

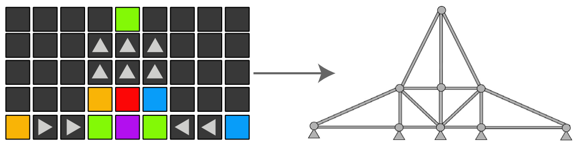
Fig. 3 An illustration of the conversion of a Deva-produced image to a truss.

Kowaliw *et al.* showed that indeed, evolution was capable of developing good designs, and moreover that manipulation of the fitness function allowed for the evolution of several distinct truss designs [24].

3 Regularities

The notion of *regularities* is slippery, and, to the best of our knowledge, has thus far not been rigorously defined[1]. We believe that usage is based loosely on Wolpert's notion of gradient-based, positional information [39]. As a working definition, we take regularity to mean some simple gradient-based function on a physically defined space. Indeed, this description seems to encompass some associated biological phenomena, such as: various symmetries, repetition, and repetition with variation. Regularities in artificial development are well studied and present in many models. Arguably the first AD attempt, Turing's model of chemical morphogenesis relied implicitly on such mechanisms through chemical diffusion [38].

The idea of regularities, at least under our conception, is a primarily geometric phenomenon. It is, however, applicable to areas of inquiry where notions of physical space are typically not used: a common example is in the design of neural network topology or weighting. In traditional neural networks, physical space is unimportant, and training is accomplished strictly via the topology of the network. Yet, there are many new approaches that utilize the proximity of neurons as a key factor in development (perhaps motivated by neurological studies suggesting that physical proximity is still an important concept in the biological brain).

The notion of regularities can, in principle, be easily added to nearly any encoding scheme. Seys and Beer [34], for instance, have explored the explicit use of bilateral symmetry in a neural network topology. They compared two encodings for neural controllers: a direct encoding and a symmetric encoding. This symmetry was achieved by copying a smaller direct encoding of network connections twice. Indeed, not only are the symmetric encodings more evolvable than their asymmetric counterparts, these gains remain even when the latter genomes are made larger in size than the former: Seys and Beer conclude that in their domain, genetic re-use is more valuable to efficient evolution than genotypic size.

Some authors argue that regularities of this form are the sole necessary idea from biology in order to generate scalable designs. For instance, Stanley *et al.* [35] have designed an encoding scheme termed Compositional Pattern Producing Networks (CPPNs). This encoding consists of a composition of primitive functions: simple functions defined on a (potentially continuous) coordinate space. For any point in the coordinate space, the composition of functions returns a value, the collection of which make up the phenotype. Interpreting the results as an image in two-dimensional space, images showing phenomena such as repetition with variation and other life-like phenomena are easily evolved. CPPNs have been further applied to neural network connection weight discovery, where the same set of underlying

[1] One proposed definition from Lipson defines "structural regularities" as an "inverse Kolmogorov complexity", that is, the amount by which a structural description could be compressed by a Universal Turing Machine [26]. Unfortunately, Kolmogorov complexity is generally uncomputable, and simpler forms of compression are often poorly correlated [20]. Further, the attempt to approximate the Kolmogorov complexity of evolved designs does not figure largely in current AD literature. Thus, we believe that most authors are referencing something quite different from Lipson's definition when discussing "regularities".

regularities has been shown to be capable of generating efficiently evolved neural networks, and also has been shown to have potential for scaling-up with problem size.

3.1 Regularities via Developmental Mechanisms

While regularities can be explicitly specified, this is certainly not a necessity, nor the norm. Several regularities are implicitly generated by many popular models of development: for instance, symmetric or repetitive forms are typical for many developmental systems; repetition with variation is commonly seen through exceedingly simple tree-growing L-systems, or, say, in the Sierpinski triangle as generated by simple cellular automata.

Kowaliw *et al.* were concerned with understanding the effect that low-level changes to Deva's developmental mechanisms would have on the final result achieved via evolution. Indeed, it is not clear how the potentially chaotic CA-like developmental stage would interact with the ergodicity of evolutionary computation. To test this, they modified the Deva growth algorithm to support four different sorts of cell division. These division mechanisms were:

- *Orig*, the original model using a *best free location* algorithm for choice of direction
- *Dir*, where directions are explicitly specified directly by the genome
- *Clockwise*, where the direction chosen is a simple, heavily biased function: always select left, or the next free location going clockwise
- *AllWay*, where division occurs in all four (available) directions simultaneously.

The latter strategy produced left-right symmetric Deva images (and hence, symmetric trusses, up to cell type). These developmental mechanisms were contrasted through a series of empirical runs. A simple fitness function (one maximized by two simple pyramids composed of four long members) was used. Some exemplar trusses discovered by these methods are shown in Fig. 4.

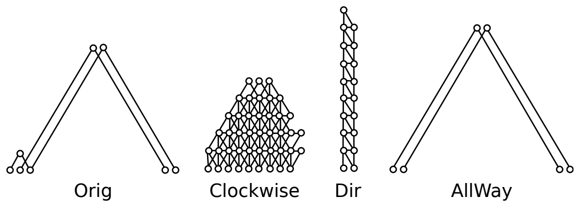

Fig. 4 Best trusses found by strategy. All trusses were stable, and all but the *Dir* truss were capable of supporting the external load.

Indeed, significant differences were discovered between the performance of the four Deva algorithms. Both the *AllWay* and the *Orig* strategies typically found trusses near the suspected global optimum. However, *AllWay* did so with less evolutionary effort, and discovered a better truss overall. The *Dir* version had a strong tendency to become stuck in a local optimum: nearly all discovered trusses consisted of very tall and thin structures, which, while stable, could not support the applied external load. Such trusses seem to maximize aspects of fitness very quickly, but cannot, from this stage, evolve into load-bearing trusses. Hence, the allowance of full genetic control in the division operator seems to be a hindrance, while the tyranny of a pre-specified procedure for cell growth forces a more robust approach. The performance of the *Clockwise* version was unsurprisingly poor. Organisms evolved through the *Clockwise* strategy had a tendency to resemble large groups of identical beams and joints leaning left, of lesser height (due to a tendency to lean left) than other trusses from these experiments.

It seems that low-level mechanisms can indeed be found to match the problem space, and can be utilized to increase the evolvability of solutions through evolutionary computation. In this case, the *AllWay* division strategy successfully increased performance by exploiting the problem space's preference for (or perhaps indifference to) left-right symmetric designs [19].

3.2 Genetic Re-use

An alternative view of regularities is that they are instances of a more general phenomenon, genetic re-use. Indeed, in many cases previously discussed, genetic re-use is inherently included in the implementation of the regularity. For instance, Hornby shows that encouraging regularity and modularity improves the efficacy of evolutionarily designed objects, such as tables [14], and in the process, produces examples which do not seem easily described as a simple gradient function. Indeed, given that geometric regularities are often, possibly always, the result of genetic re-use, it is difficult to distinguish between the two.

However, we note that the genetic re-use conception of increased evolvability is inherently encoding-specific, and hence does not allow us to discuss the possibility that two unrelated mechanisms accomplish the same thing, or operate in the same way.

4 Environmental Feedback

A second, less studied developmental mechanism is that of *local adaptation*. By this, we mean the exploitation of local environmental information by individual components during embryogenesis. "Polymorphism" is a term used by biologists to refer to the capacity of some organisms to achieve any of several *morphs*—distinct phenotypic types—on the basis of environmental cues. A dramatic example is the capacity of some species (e.g., alligators) to select their gender during development based on embryonic environmental conditions (e.g., temperature) [5].

Here, our notion of environmental feedback is highly inclusive, taking into account any change based on external environmental influence. This means that the number of distinct phenotypes for any genotype is potentially infinite. There is some sense in which nearly all AD models exploit environmental feedback, since nearly all include some container in which a phenotype develops and component-to-component interaction is possible. We, however, will restrict our attention to cases where some additional information source is included, one which a developing phenotype can incorporate or adapt to. In short, an external mechanism by which a single genotype can express several reactive phenotypes, and do so reliably. It is our belief that through this local environmental interaction the developing organism can "add information" to its phenotype not explicitly included in its genotype.

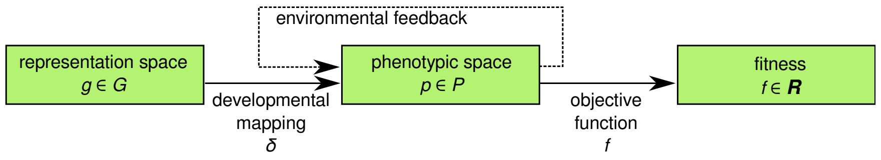

Fig. 5 An illustration of inclusion of environmental feedback into the AD process.

There has been ample work on the incorporation of environmental influence into visual models of plant development. An early example was Open L-systems, where models of plant growth were replaced by a second and distinct environmental process, and morphological plasticity—visually convincing output of plant growth adapted to environmental stimulus—was demonstrated [27]. Indeed, these ideas have been extended in several directions. For instance, works creating visual models of trees include some interaction with external forces or global properties such as gravity or amount of light, geometric effects such as support for climbing plants, and bi-directional effects such as inter-organism competition for resources [33].

While these results are certainly visually impressive, and do indeed seem to capture the interesting causal effects of their biological counterparts, they do little to inform us on the relative evolvability of developmental systems that incorporate such mechanisms. For this, we need some well-defined task and measurable notion of success. Some work on the matter has been conducted in regard to resistance to environmental perturbation in the development of an artificial agent [22], and in the development of distinct circuits from various initial environmental states [37]. We outline below the adaptation of the Deva system to different environmental specifications [23].

4.1 Environment as Geometric Constraint

Kowaliw *et al.* used spatial geometry as a means to guide morphological growth. They did so by specifying a collection of alternative geometric environments for growth, and evolving good genotypes in each. The conjunction of spatial geometry and objective function indeed could be used to control evolutionary output. The same objective function, utilizing differing spatial geometries, produced wildly different, but functioning, designs. Next, a series of experiments compared phenotypic expression of evolved genotypes to new phenotypes grown at sizes and in environments new to the genome. It was shown that genomes re-grown at different sizes usually retained much of their fitness, and that some genomes evolved in a particular spatial geometry could perform well when regrown in a different environment. Moreover, genomes that performed well in some previously unseen environment tended to perform well in *all* the environments. The phenotypes of one such genome are illustrated in Fig. 6 [23].

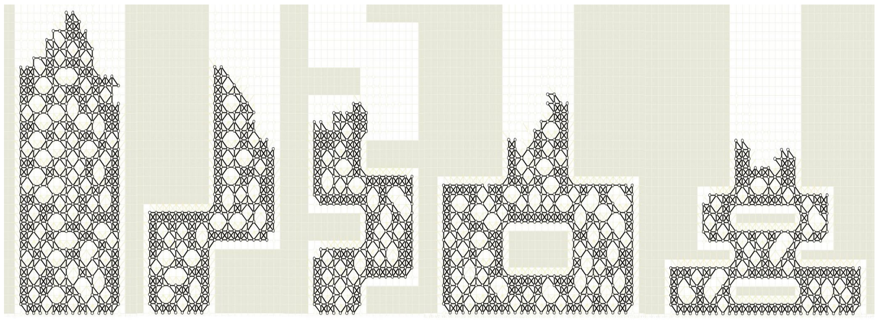
Fig. 6 Phenotypic regrowth of a single genome in multiple environments.

This work has potential repercussions on the use of AD for engineering design. First, it shows that one can potentially discover genomes having some resistance to variations in the problem domain. This, in turn, suggests that a genetic-toolkit approach might be valuable. Second, it shows that the inclusion of an explicit developmental mapping between representation and pattern includes an alternative context for the imposition of constraints. This suggests that alternatives to tuning the fitness function or multi-objective methods exist, and that geometric information can indeed be added to the developing phenotype.

5 Fitness Feedback

There is a sense in which the previously described work does not include the full benefits of environmental feedback, since the environment was explicitly specified by the programmers. One would prefer the more general sense in which nature's exploitation of environmental information serves to reinforce "good" phenotypic developments, and penalize "bad" ones, where "good" is linked to the ultimate success of the organism. In this case, amongst other mechanisms, there is a "viability constraint" at play, one in which growth can simply not occur in the absence of sufficient resources, and sufficient resources will not exist without a pre-existing "good" design.

As a simple illustration, consider the hypothetical example in Fig. 7. Let us assume that the left image is a viable tree, adept at distributing resources between the base and the leaves. Further, let us assume that the right image is not a viable design, for the evident reason that the thickness of the upper branches significantly outweigh the thickness of the base. Most evolutionary representations make both designs equally likely, a consequence of a generally unbiased combinatorics of parameters. In nature, however, the necessity of resource transport during growth would surely make the right design impossible, that is, the thickness of the upper branches would be naturally constrained by the amount of resources which could be transported through the trunk, and such an imbalanced tree could not grow.

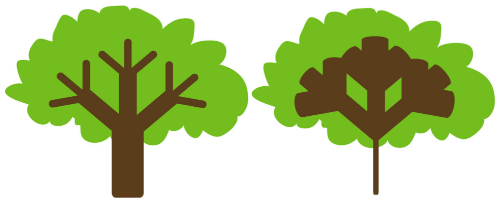

Fig. 7 The left image is (assumably) a viable tree design, and the right image a silly one. In nature, the necessity of resource transport during growth would likely make the right design impossible.

One possible model for this fitness-feedback is shown in Fig. 8. A significant problem with this model is that it requires repeated computation of the fitness function. In many design problems, however, the cost of fitness calculation dominates the developmental process. Hence, computational complexity will be a significant concern when using this model. We will refer to the computation time of the developmental stage as $\delta(n)$, and the computation time of the fitness as $\varphi(n)$. We further note that for many AD models, including CAs, L-systems, and matrix re-writers, the development can be decomposed into two levels: $\delta(n) = \delta_{\text{time}}(n) \cdot \delta_N(n)$, where δ_{time} refers to the number of discrete time steps, and $\delta_N(n)$ the time taken by each of the components during a single time step.

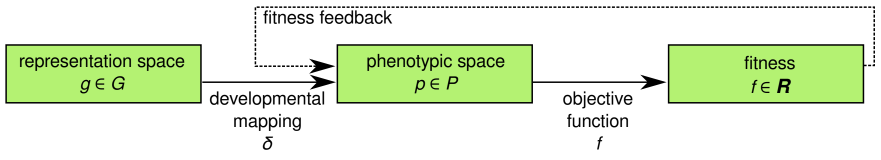

Fig. 8 An illustration of the inclusion of fitness feedback in the AD process. Unlike most applications in AD, here fitness is computed as a component of the developmental stage.

To test these theories, we have explored a model of fitness-feedback in artificial development. Due to computational constraints, we have chosen to operate on a toy problem, since the comparison of several strategies required *many* evaluations to be undertaken. This work is summarized below (the interested reader can find full explication in [21]).

5.1 A Highly Simplified Model of Vasculogenesis

The problem is as follows: Given some two-dimensional developmental environment, including a "start" square, design a network of cells capable of distributing a resource to as many cells as possible from a single source cell. Our task consists of "normal", "transport", and "barrier" cells. Beginning with the start cell, fluid is distributed to all transport cells in the immediate von Neumann neighborhood recursively. Finally, any normal cells which are bordered by a fluid-carrying transport cell are considered "served". The *global fitness* is the number of "served" normal cells:

$$f_{\text{glob}}(E) = \frac{1}{|E|} \sum_{c \in E} \begin{cases} 1; & \text{if color}(c) = \text{ "normal" } \wedge c \text{ is "served"} \\ 0; & \text{otherwise.} \end{cases} \tag{1}$$

where $|E|$ is the total number of cells of color "normal" and "transport" in E. Hence, we aim to design a transport system that covers as much of the non-barrier environment as possible in a connected fashion, but one which does not take up more space than is necessary. Since the process is governed by a flood-fill extending from the start cell, it can be accomplished in time $O(|E|)$.

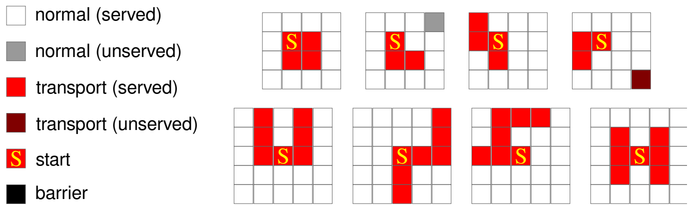

Fig. 9 Examples of optimal configurations in toroidal environments of size 4×4 and 5×5.

In smaller cases, via brute force, we can find the global optima: there are 44 optimal configurations if E is a torus of dimension 4×4, and 126 at dimension 5×5. Some examples are shown in Fig. 9. The global optimum lies around $f_{\text{glob}} = 2/3$, and it is likely that any particular environment will support many distinct global optima.

The starting environment consists of "normal" and "barrier" cells everywhere except at the start square, which is of type "transport". We will refer to any development of this environment over time toward some final network as *vasculogenesis*, or simply as growth. Our general goal is to find developers that optimize f_{glob}. Via the "barrier" symbol in our alphabet, it is possible to generate a collection of random starting environments, ones that divide a space into "normal" and "barrier" regions, thus give rise to a very large number of variations on our vasculogenesis problem, each with a different geometry for vascular growth. These environments are produced by stochastic placement of dots in a containing rectangle, and next a division of space into Voronoi diagrams (see Fig. 10 for an example). Steps are taken to ensure that all "normal" regions are connected by a corridor of width at least 3. The number of dots used in the construction, dC, seems—at least by visual estimation—to form a measure of complexity on the space of all possible environments. This is illustrated in Fig. 11.

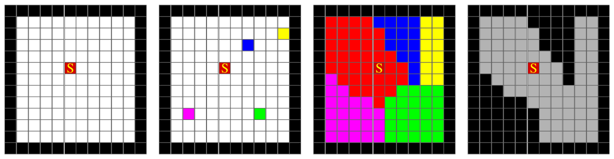

Fig. 10 Illustration of the creation of a random environment: *(from left to right)* The start cell is placed in the centre; four random points are placed; Voronoi regions are computed; the regions are assigned to either "barrier" or "normal".

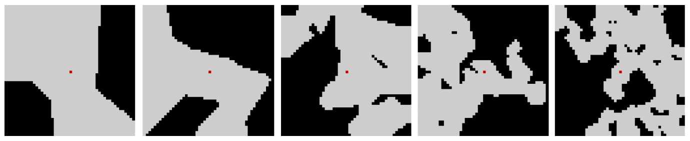

Fig. 11 Environments with dot complexities *(from left to right)* $dC = 5, 20, 50, 100, 200$.

5.2 *Vasculogenesis Strategies*

In this section we outline a number of strategies for solving our vasculogenesis problem, some pre-programmed and some evolved.

5.2.1 Cellular Automata (CA)

The CA strategy is a straight-forward implementation in two-dimensional CAs. Representation consists of a cell state for each possible neighborhood, where a cell state is either "normal", "transport" or "barrier". This leads to a ruleset representation of size $2 \cdot 3^{|N|-1}$, where the neighborhood is a cross extending a distance of two cells in each of the four primary directions (i.e., $N = 9$). At each time step, in parallel, each cell collects a description of its local neighborhood, queries the ruleset for an action, then changes its cell state. An illustration of a possible collection of rules are shown in Fig. 12. Some exemplars of evolved CAs solving the vasculogenesis problem can be seen in Fig. 13.

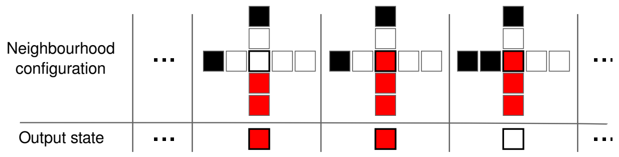

Fig. 12 Three of the $2 \cdot 3^{|N|-1}$ rules making up a particular cellular automaton. At each developmental time step, every cell in the environment will find the matching neighborhood configuration and change color to the associated output state.

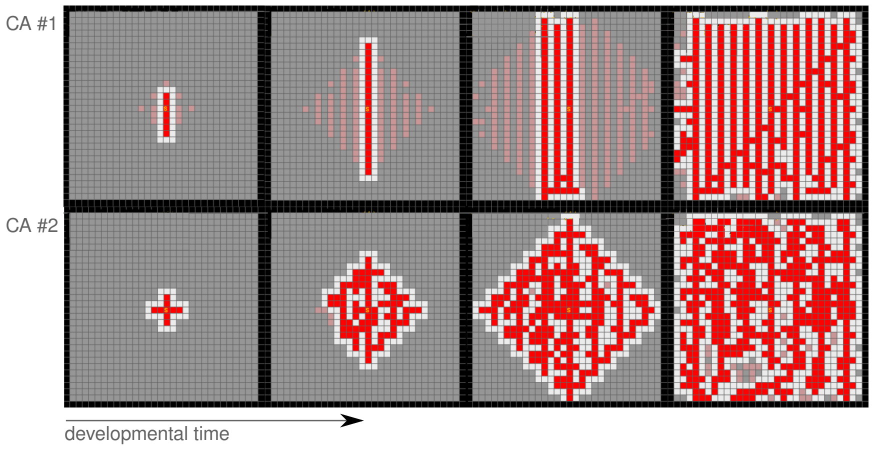

Fig. 13 Two evolved exemplars of the CA strategy. The configurations are far from optimal: transport cells are placed too close together (ideally, they should be separated by two normal cells rather than one), sometimes transport cells are placed together in clusters, and there is unnecessary redundancy in the connection network. We attribute this poor performance to the difficulties of evolving cellular automata generally.

5.2.2 Local Fitness

Two local functions were designed to capture important aspects of the problem. The goal is to generate simple functions capable of approximating the fitness function in constant time, in an attempt to optimize the global design via many simple local optimizations: in this respect, it resembles energy minimization.

A "blind" local fitness function, f_{blind} was defined on a local neighborhood surrounding a point. In this function, the (often false) assumption was made that any present "transport" cells were served (i.e., the explicit assumption was made that this neighborhood was connected to the central start cell); f_{blind} then returned the number of served "normal" cells.

A "sighted" local fitness function, f_{sight} was designed under the assumption that global fitness would be computed with each discrete time step, so that any particular cell would have access to information regarding its connectivity to the central cell in the previous time step. Hence, a lesser assumption was made: that any "transport" cells in the current neighborhood had the same connectivity in this time step as they did in the last time step. The sighted local fitness function then computed the number of served “normal” cells in the local neighborhood, and output that number.

5.2.3 Fitness-Enhanced Random Growth

The first attempted strategies were not evolved at all: the effects of fitness-enhancement were investigated on random growth instead, that is, a scheme was designed by which any “transport” cell in the developing phenotype had a probability of being copied in a random direction. However, for each proposed change, the current f_{glob} value and the f_{glob} value that would result if the change were made were computed. If the expected fitness was higher than the current fitness, the change was made. This strategy is called the Random Global fitness-enhanced greedy Growth (RGG). Since f_{glob} is computed at each time step, for a maximum of $|E|$ time steps, we have complexity $O(|E|^3)$. A local version was also designed, one which utilized f_{blind} rather than f_{glob}. This strategy was named the Random Local fitness-enhanced greedy Growth (RLG) strategy, taking time $O(|E|^2)$.

Knock-out versions of these strategies were also considered: that is, development began with an environment filled entirely with “transport” cells, and randomly chosen cells were changed to type “normal”, if local or global fitness improved. The Random Local fitness-enhanced greedy kNockout (RLN) strategy has running time $O(|E|^2)$, while the Random Global fitness-enhanced greedy kNockout (RGN) strategy has running time $O(|E|^3)$.

The RLN strategy performed very poorly, almost immediately devolving to a single served point. We have included discussion of the strategy to highlight that the f_{blind} local fitness function has a limitation: it cannot see global connectivity, and hence cannot preserve it outside of a purely constructive context. Instances of the other random strategies, RLG, RGG, and RGN, are shown in Fig. 14.

5.2.4 Fitness-Enhanced Cellular Automata (F-CA)

Finally, the global and local fitness functions were used to enhance CA rules, instead of random growth. By a Constructive Cellular Automaton (CCA), we mean a CA in which it is possible for a “normal” cell to change state to a “transport” cell, but not vice-versa. Thus, a CCA is a much simpler machine than a CA, evidenced by the recognition that a CCA must eventually reach a stable point. Since CCAs cannot eliminate global connectivity once it has been created, however, they can be used with the f_{blind} fitness function, in much the same way as the RLG strategy.

The Blind Local Fitness-enhanced CCA (BLF-CCA) strategy is a constructive CA guided via the f_{blind} fitness function, that is, for each possible cell action suggested by the rule set, the value of f_{blind} is computed first for the current neighborhood, and next for the neighborhood with the action undertaken. If the proposed action increases f_{blind} then the action is undertaken; otherwise, no action is taken.

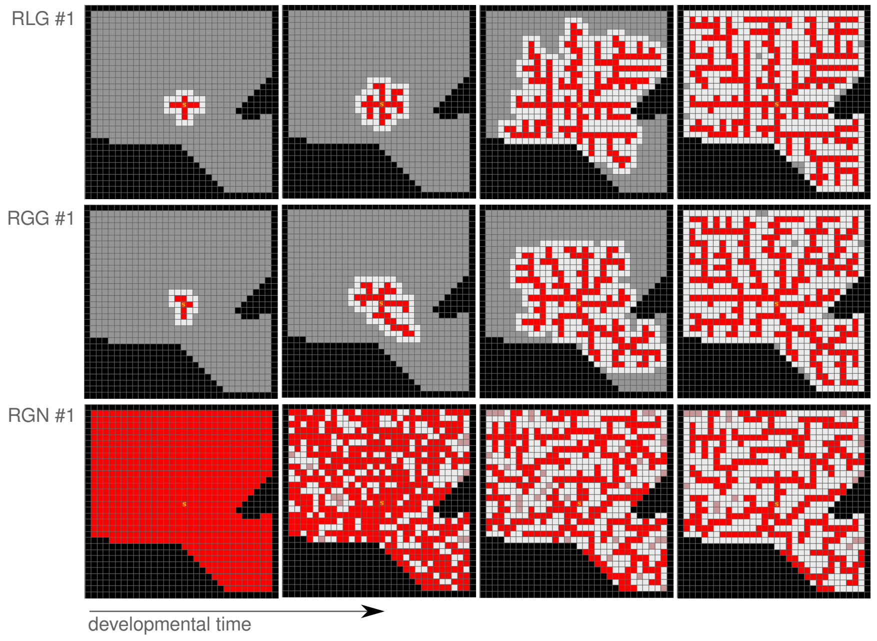

Fig. 14 Exemplars of the RLG, RGG, and RGN strategies. The greedy nature of random growth leads to sub-optimal configuration here: for instance, transport cells are placed too close together (separated by one rather than two normal cells), and, for the RGN strategy, some transport cells are located directly adjacent to barriers.

Since the f_{blind} function can be computed in constant time, running time is unchanged between the CA and the BLF-CCA strategies: $O(|E|^2)$.

For more robust means of computing fitness—such as f_{sight} and f_{glob}—the possibility exists that we can use a full CA, and hence potentially access a wider breadth of growth types. The Sighted Local Fitness-enhanced CCA (SLF-CCA) strategy includes the use of the f_{sight} local fitness function as a guide. With every developmental step, global fitness is computed. Then, for every cell action, the difference in f_{sight} is computed for the original and new neighborhoods, and the action is undertaken only if f_{sight} increases. Since we compute global fitness at each time step, and since f_{sight} is constant time, we have a running time of $O(\delta_{\text{time}}(|E|) \cdot \varphi(|E|) + \delta(|E|)) = O(|E|^2)$.

Finally, the Global Fitness-enhanced CCA (GF-CA) strategy computes f_{glob} for every cell action, and cell actions are only executed if there is an expected increase. This leads to a running time of $O(\delta(|E|) \cdot \varphi(|E|)) = O(|E|^3)$. Even for this relatively small domain of application, using a global fitness-enhanced ADS is computationally expensive, with individual evolutionary runs requiring several hours to complete. An illustration of highly evolved individuals' growth can be seen in Fig. 15.

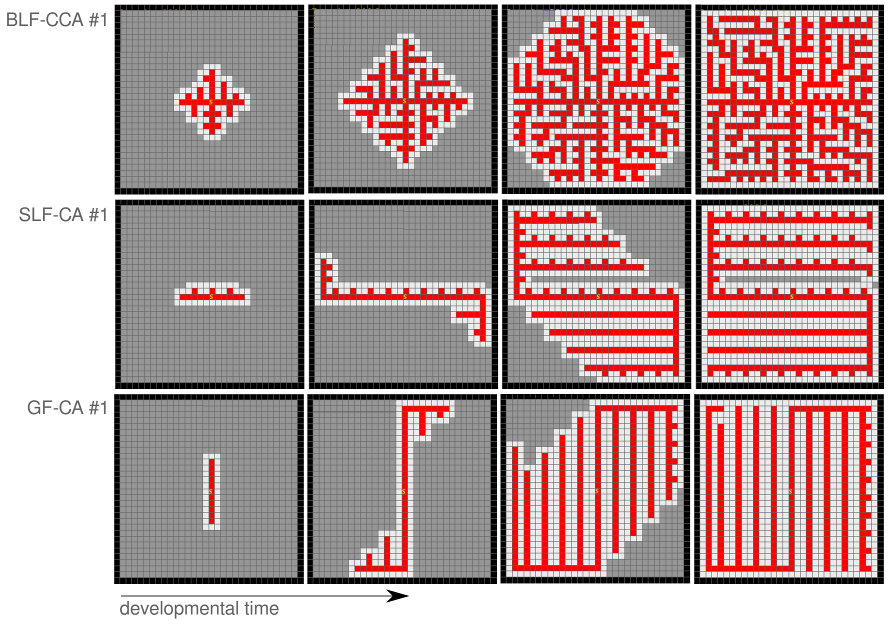

Fig. 15 Exemplars of the BLF-CCA, SLF-CA, and GF-CA strategies. Much like the random growth strategies, the BLF-CCA strategy places transport cells too close together for optimal distribution. The SLF-CA and GF-CA strategies do better in this regard.

5.3 *Comparative Experiments*

As previously alluded to, the strategies were evolved through a standard evolutionary algorithm 50 times and contrasted. The random growth strategies were also run 50 times. The first experiments were conducted in an empty environment, that is, a simple square of "normal" cells of size 30×30. The average best global fitness is shown in Fig. 16.

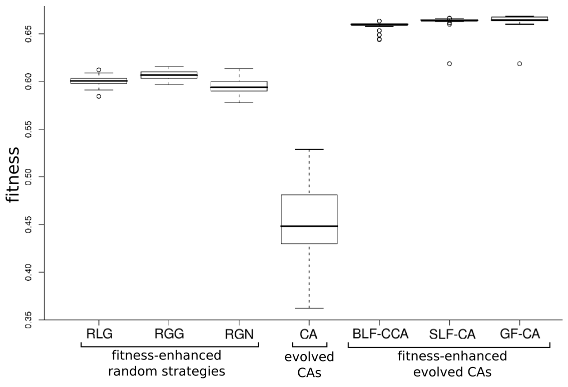

Fig. 16 Boxplots of the fitness values of vasculogenesis strategies in the trivial environment.

The random strategies, despite being quite simple, performed admirably well. Interestingly, the local fitness-enhanced strategy RLG performed with the same efficacy as the global fitness-enhanced strategy RGG, making the excess computational time spent on computing global fitness unnecessary. The global knockout strategy performed approximately as well as RLG and RGG. It should be further noted that the random strategies did not optimize the layout of the transport-veins, where the more efficient network (with two spaces between veins) is sacrificed for greedy immediate fitness gain (one space between veins).

The simple CA strategy performed very poorly. It seems that the chaotic growth of CAs, and the associated issues with evolvability, proved too daunting to overcome in this context. In contrast, the fitness-enhanced strategies all outperformed both the standard CA strategy, and the random strategies. This in terms of both overall fitness and speed of convergence. It is clear that the combination of fitness-enhancement and artificial development is a strong boon in this context.

5.4 Dependency on Problem Complexity

The above experiments were repeated, this time using randomly generated environments. At the beginning of each run, a particular starting environment was generated with dC between 10 and 75. For each strategy, the Spearman's rank correlation coefficient (ρ) between the variables dC and fitness was computed. Each strategy was also evaluated through 50 runs.

Indeed, as expected with a more difficult problem, fitness was generally lower for all strategies. However, the general trend remained the same, with the CA strategy performing the worst, followed by the random strategies, and the F-CA strategies all clustered near the expected global optimum. The correlation between environmental complexity dC and fitness scored were highly negative for both the CA and BLF-CCA strategies, with values of -0.608 ($p < 0.001$) for the CAs and -0.603 ($p < 0.001$) for the BLF-CCAs. This is not surprising: the more complex the environment, the lower the performance. Indeed, our intuition that dC was a measure of problem difficulty was confirmed with these findings.

It is interesting to note that there was little to no recognizable dependency on problem complexity ($|\rho| < 0.2$) for all random strategies. Arguably, the pre-programmed strategies are the simplest, and are likely the most adaptive as a result. Similarly, the GF-CA and the SLF-CA strategies were also resistant to problem complexity ($|\rho| < 0.3$) suggesting that the combination of fitness-enhancement and a robust developmental programs can sometimes overcome problem complexities.

5.5 Summary of Vasculogenesis Results

The use of global fitness as a feedback mechanism, at least in the explored context, made immediate and substantial improvements to the efficacy of the evolved solutions. Indeed, even the random-growth strategies exceeded the capacity of the evolved cellular automata! And augmentation of an evolved cellular automaton with global fitness quickly achieved values we believe to be optimal.

In nearly any real-world scenario, however, global fitness is likely an intensive process, and computing its value for each incremental change during development is unlikely to be feasible. The use of a local fitness function, as demonstrated here, can recover much of the efficiency gains with little to no change to asymptotic running time.

Finally, in the vasculogenesis problem, there existed a notion of problem complexity, associated with the complexity of the environment in which the development took place. The use of global fitness, and one local fitness strategy, eliminated this dependency, suggesting that fitness-feedback is capable of promoting self-adaptation during development to environmental conditions.

6 Conclusions

In this chapter, we have argued that artificial development is an appropriate means of approaching complex systems engineering. Artificial development works via the inclusion of mechanisms that enhance the evolvability of a design space. Two of these mechanisms, regularities and adaptive feedback with the environment, have been reviewed.

The explicit use of biologically inspired mechanisms of development, including local adaptation, have been shown to potentially improve the evolvability of complex problem spaces. It is our belief that more such developmental mechanisms exist—possibly including mechanisms such as cell physics, or cell chemistry-induced biases—and form a means of approaching complex problems.

Acknowledgements TK and WB would like to thank Nature for billions of years of tireless effort. Smashing stuff! In addition NSERC of Canada supported WB by discovery grant RGPIN 283304-07.

References

1. W. Arthur. The effect of development on the direction of evolution: Toward a twenty-first century consensus. *Evolution & Development*, 6(4):282–288, 2004.
2. W. Banzhaf and N. Pillay. Why complex systems engineering needs biological development. *Complexity*, 13(2):12–21, 2007.

3. N.L. Britton and A. Brown. *An illustrated ora of the northern United States, Canada and the British Possessions, vol. 2*. C. Scribner's Sons, 1913.
4. E. Coen. *The Art of Genes: How Organisms Make Themselves*. Oxford University Press, 1999.
5. D. Crews and J.J. Bull. Sex determination: Some like it hot (and some don't). *Nature*, 451:527–528, 2008.
6. R. Dawkins. *Climbing Mount Improbable*. W.W. Norton & Company, 1996.
7. A. Devert, N. Bredeche, and M. Schoenauer. Artificial ontogeny for truss structure design. *Self-Adaptive and Self-Organizing Systems Workshops, IEEE International Conference on*, 0:298–305, 2008.
8. D. Federici and K. Downing. Evolution and development of a multicellular organism: Scalability, resilience, and neutral complexification. *Artificial Life*, 12(3):381–409, 2006.
9. S. F. Gilbert and D. Epel. *Ecological developmental biology : integrating epigenetics, medicine, and evolution*. Sinauer Associates Inc., 2009.
10. S. J. Gould. *The Structure of Evolutionary Theory*. The Belknap Press of Harvard University Press, 2002.
11. F. Gruau, D. Whitley, and L. Pyeatt. A comparison between cellular encoding and direct encoding for genetic neural networks. In *GECCO '96: Proceedings of the First Annual Conference on Genetic Programming*, pages 81–89, Cambridge, MA, USA, 1996. MIT Press.
12. S. Harding and J. Miller. The dead state: A comparison between direct and developmental encodings. In *GECCO '06: Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*, 2006.
13. J. L. Hendrikse, T.E. Parsons, and B. Hallgrmsson. Evolvability as the proper focus of evolutionary developmental biology. *Evolution & Development*, 9(4):393–401, 2007.
14. G. Hornby. Measuring, enabling and comparing modularity, regularity and hierarchy in evolutionary design. In *GECCO '05: Proceedings of the 2005 conference on Genetic and evolutionary computation*, pages 1729–1736, New York, NY, USA, 2005. ACM.
15. P. R. Huttenlocher. *Neural plasticity : the effects of environment on the development of the cerebral cortex*. Harvard University Press, 2002.
16. A. Ilachinski. *Cellular Automata: A Discrete Universe*. World Scientific, 2001.
17. R. Kicinger, T. Arciszewski, and K. A. De Jong. Morphogenic evolutionary design: cellular automata representations in topological structural design. In I. C. Parmee, editor, *Adaptive Computing in Design and Manufacture VI*. Springer-Verlag, 2004.
18. R. Kicinger, T. Arciszewski, and K. A. De Jong. Evolutionary computation and structural design: a survey of the state of the art. *Computers & Structures*, 83(23-24):1943–1978, 2005.
19. T. Kowaliw. *A Good Number of Forms Fairly Beautiful*. PhD thesis, Concordia University, Montréal, QC, Canada, 2007.
20. T. Kowaliw. Measures of complexity for artificial embryogeny. In *GECCO '08: Proceedings of the 10th annual conference on Genetic and evolutionary computation*, pages 843–850, New York, NY, USA, 2008. ACM.
21. T. Kowaliw and W. Banzhaf. Augmenting artificial development with local fitness. In A. Tyrrell, editor, *Evolutionary Computation (CEC), 2009 IEEE Congress on*, pages 316–323, 2009.
22. T. Kowaliw, P. Grogono, and N. Kharma. Bluenome: A novel developmental model of artificial morphogenesis. In Kalyanmoy Deb et al., editor, *GECCO '04: Proceedings of the 6th Annual Conference on Genetic and Evolutionary Computation*. Springer-Verlag, 2004.
23. T. Kowaliw, P. Grogono, and N. Kharma. Environment as a spatial constraint on the growth of structural form. In *GECCO '07: Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pages 1037–1044, New York, NY, USA, 2007. ACM.
24. T. Kowaliw, P. Grogono, and N. Kharma. The evolution of structural form through artificial embryogeny. In *IEEE Symposium on Artificial Life, ALIFE '07*, pages 425–432. IEEE, 2007.
25. A. Lindenmayer. Mathematical models for cellular interaction in development. *Journal of Theoretical Biology*, 18:280–315, 1968.

26. H. Lipson. Principles of modularity, regularity, and hierarchy for scalable systems. *Journal of Biological Physics and Chemisty*, 7:125–128, 2007.
27. R. Mech and P. Prusinkiewicz. Visual models of plants interacting with their environment. In *SIGGRAPH '96 Proceedings*, volume 30, pages 397–410, 1996.
28. R.M.H. Merks, M.H. Roeland, A.G. Hoekstra, J.A. Kaandorp, P.M.A. Sloot, and P. Hogeweg. Problem-solving environments for biological morphogenesis. *Computing in Science and Engg.*, 8(1):61–72, 2006.
29. J. Miller. Evolving a self-repairing, self-regulating, french flag organism. In Kalyanmoy Deb et al., editor, *GECCO '04: Proceedings of the 6th Annual Conference on Genetic and Evolutionary Computation*, pages 129–139. Springer-Verlag, 2004.
30. B. Mulder. On growth and force. *Science*, 322:1643–1644, 2008.
31. S. A. Newman, G. Forgacs, and G. B. Müller. Before programs: The physical origination of multicellular forms. *International Journal of Developmental Biology*, 50:289–299, 2006.
32. J. Rieffel and J. Pollack. The emergence of ontogenic scaffolding in a stochastic development environment. In Kalyanmoy Deb et al., editor, *GECCO '04: Proceedings of the 6th Annual Conference on Genetic and Evolutionary Computation*. Springer-Verlag, 2004.
33. S.I. Sen and A.M. Day. Modelling trees and their interaction with the environment: A survey. *Computers & Graphics*, 29(5):805 – 817, 2005.
34. C.W. Seys and R.D. Beer. Effect of encoding on the evolvability of an embodied neural network. In *GECCO '06: Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*, 2006.
35. K.O. Stanley, D.B. D'Ambrosio, and J. Gauci. A hypercube-based encoding for evolving large-scale neural networks. *Artificial Life*, 15(2):185–212, 2009.
36. T. Steiner, Y. Jin, and B. Sendhoff. A cellular model for the evolutionary development of lightweight material with an inner structure. In *GECCO '08: Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation*, pages 851–858. ACM, 2008.
37. G. Tufte and P. C. Haddow. Extending artificial development: Exploiting environmental information for the achievement of phenotypic plasticity. In *Evolvable Systems: From Biology to Hardware*, 2007.
38. A. Turing. The chemical basis of morphogenesis. *Philosophical Transactions of the Royal Society B*, 237:37–72, 1952.
39. L. Wolpert and G. Dover. Positional information and pattern formation. *Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences*, 295(1078):441–450, 1981.
40. O. Yogev, A.A. Shapiro, and E.K. Antonsson. Modularity and symmetry in computational embryogeny. In *GECCO '08: Proceedings of the 10th annual conference on Genetic and evolutionary computation*, pages 1151–1152, New York, NY, USA, 2008. ACM.