Augmenting Artificial Development with Local Fitness

Taras Kowaliw and Wolfgang Banzhaf

Abstract— **In biology, the importance of environmental feedback to the process of embryogenesis is well understood. In this paper we explore the introduction of a local fitness to an artificial developmental system, providing an artificial analogue to the natural phenomenon. First, we define a highly simplified model of vasculogenesis, an environment-based toy problem in which we can evaluate our strategies. Since the use of a global fitness function for local feedback is likely too computationally expensive, we introduce the notion of a neighbourhood-based "local fitness" function. This local fitness serves as an environmental-feedback guide for the developmental system. The result is a developmental analogue of guided hill-climbing, one which significantly improves the performance of an artificial embryogeny in the evolution of a simplified vascular system. We further evaluate our model in a collection of randomly generated two-dimensional geometries, and show that inclusion of local fitness helps allay some of the problem difficulty in irregular environments. In the process, we also introduce a novel and systematic means of generating bounded, connected two-dimensional geometries.**

I. Introduction

Artificial Development (AD, or Artificial Embryogeny, Artificial Ontogeny, etc.) is a field concerned with the simulation and exploitation of a developmental stage in a machine learning task. Through mimicking biological embryogenesis, researchers aim to achieve ever-larger scales of combinatorial organization, and to incorporate features of biological organisms, such as heightened evolvability and self-repair. Previous work in AD has explored the inclusion of environmental information, largely through the same mechanisms as other intercellular communication. This, however, seems to miss the largest benefit of environmental feedback — that of a local functional notion of success — which so successfully guides plant and animal growth.

Here, we introduce the notion of a local fitness function; a function which will take a local neighbourhood of cells and return a valuation of that neighbourhood's functional capacity. Although we do not specify a form for this local fitness in the general case — it is believed too domain specific to do so — we note that it is plausible that such a function be constructible. Specifically, the use of AD generally suggests the gradual organization of discrete components; this suggests that components generally play some functional role, and that local interaction combines to form global fitness. So long as the epistasis of cumulative local interactions is not too high, some meaningful information can be collected, or perhaps, deleterious combinations avoided.

We introduce several novel models: firstly, we introduce a systematic means of generating random, bounded and connected two-dimensional neighbourhoods; secondly, we introduce a novel toy problem in which environmental feedback is expected to play a major role, that of a highly simplified vascular system; finally, we introduce two local fitness functions for this domain, and explore their addition to a standard form of AD, the Cellular Automaton (CA). In this case, we demonstrate that the local fitness can be included without increasing the asymptotic time required.

Through comparison between evolved and pre-programmed strategies, we show that even very simple strategies augmented with local fitness outperform the evolved CA. Further, we show that augmentation of the CA with local fitness improves performance further, both in overall fitness and in computation time. Finally, we show that augmentation of the AD with local fitness guidance helps evolution cope with environmental perturbations associated with increased problem difficulty, i.e., that environmental feedback can help an AD respond to its environment adaptively during development.

Taras Kowaliw and Wolfgang Banzhaf are with the Department of Computer Science, Memorial University of Newfoundland, St. John's, NL, Canada, A1B 3X5 (taras@kowaliw.ca, banzhaf@cs.mun.ca)

II. Review

A. Embryogenesis and Environmental Feedback

A simple example of a means by which environmental feedback might help with 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 neighbourhood will influence the type of leaf growth [1]. 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. Indeed, examples of environmental influence on the morphology of plants are easy to find.

There are examples in animal growth of the molding of morphology by environmental influences as well. For instance, the developing human brain is critically dependent on sight during the third to sixth post-natal months: if deprived of sight, a child will never gain a functioning visual system; Or, if a single eye is deprived, then the second functioning eye will coopt the first's sensory connections permanently. Compare this with the remarkable neural plasticity found in later stages, where people temporarily blinded can regain most function, even from artificial inputs. This suggests a critical role for sensory input in guiding the molding and pruning of neurons [2]. Vasculogenesis is another classic example. Lowe *et alia* showed that the evolution of echinoderms from bilaterals showed a remarkable plasticity in the genetic subsystem governing growth of the vascular system, where not only did existing genomic information adapt to large morphological changes, but also to an alteration in

fundamental role (i.e. transport of water rather than blood) [3]. This last example serves as an exemplar for our work here, and we describe the process in greater detail below.

Vasculogenesis, and the related process angiogenesis, describe the formation of a vascular system in an organism. The process is similar, and often aligned with, the formation of lymphatic and nervous systems. Initially, a collection of blood vessels are created de novo, from which modelling and pruning create the distinct structures which comprise the system. Processes of growth and organization are influenced by several proteins, often stage- and organ-specific. Vasculogenesis naturally promotes many constraints, such as Murray's law, in which the cube of the size of the parent vessel is approximately the sum of the cubes of the child vessels. [4].

We do not, at present, aim for a biologically plausible model; The interested reader may consult the works of Merks *et alia* for examples of simulations of the stages of vasculogenesis based on the Cellular Potts model [5], [6]. The topic interests us as an example of a case in which development is capable of generating an efficient design *without (we suspect) any genetic representation of the overall morphology*. This property, of course, has repercussions on the evolvability of a seemingly complex design. We aim at abstracting this important principle of vasculogenesis, that is, the capacity to create an efficient transport system in a relatively arbitrary environment using only local environmental cues, and to explore the addition of such environmental cues to the artificial case.

B. Developmental Systems and Cellular Automata

Artificial Developmental Systems (ADSs) are systems which include a mid-step between representation and final evaluation inspired by, but not necessarily resembling, biological embryogenesis; It is generally hoped that this mid-step process will bias produced phenotypes in fruitful directions. There are several suspected mechanisms through which desirable properties, especially evolvability, might be achieved: regularities, local adaptation, biophysics, etc. Their relative importance, however, is unknown. Indeed, Stanley has recently introduced a model which relies on only one such mechanism, regularities, doing away with a gradual developmental stage altogether [7]. Here we concentrate on the potential functional contribution of another mechanism, local adaptation, through a cellular automaton-based model.

Cellular Automata (CAs) originate from Ulam and von Neumann, in an attempt to describe a discrete counterpart for continuous dynamical systems. A CA (here) is a discrete space-time diagram, with two dimensions of space. Given some configuration at time t, the next configuration in time will be obtained by applying a transition function to each cell (automaton) and its neighbourhood in parallel, the output of which would determine the new state of the cell at time $t + 1$. Transition functions are a complete specification of each possible neighbourhood configuration associated with an output cell state. For an comprehensive introduction to CAs, see Ilachinski's text [8]. Increasingly, CAs are used as models of biological phenomena, including as models of pattern formation [9] [10].

The concept of environmental feedback in an ADS has been explored previously. Mech and Prusinkiewicz, in their Open L-Systems, explored the addition of a communication channel between an environment and a developing visual plant model [11]. The environment was modelled as a separate program to the organism, to which the (non-evolved) genome would react, producing visually convincing models of phenomena such as root growth and morphological plasticity. Tufte *et alia* have explored the matter with regards to the development of cellular automaton-based hardware design. In their work, environmental fluctuations are expressed as state changes in the cellular configuration, and are examined with regards to the capacity of a genome to generate several phenotypes [12], [13]. The notion of phenotypic plasticity through environmental factors has also been explored in a functional context by Kowaliw *et alia*, via the re-generation of genomes at several different sizes [14], [15], and in the generation of different phenotypes through different environmental spatial geometries [15]. Our approach here incorporates the latter ideas, the spatial geometry of the environment. However, our other form of environmental guidance, local fitness, is a digression from the current state of the art.

C. Local Fitness

The concept of a local fitness function is not new to Evolutionary Computation. For instance, it has been used in Potter and De Jong's framework for cooperative coevolution; In this architecture, solutions are composed of smaller subcomponents, drawn from separated evolving populations. The local fitness of a subcomponent is the expected increase in fitness in the merged individual [16]. Further, there have been applications of local fitness functions to standard genetic algorithms as well, typically in helping to guide more intelligent genetic operators (e.g. attempting to use local fitness to help prevent deleterious mutations). Recently, this concept has been extended by Aichour and Lutton to the design of intelligent operators for genetic programming [17].

Much like these previous attempts, the local fitness function here is an attempt to capture some of the difficulty of solving the global problem in a local sense. However, in this work, the local fitness is used in the developmental stage, meaning that it is applied pre-evaluation. Hence, development becomes a hill-climbing process, where the guided growth undertakes only steps which increase expected fitness.

III. Computational Complexity of Local Fitness-Enhanced Developmental Systems

We will describe an AD system as one which includes a developmental stage between representation (genome) and fitness value:

$$G \times \mathcal{E} \xrightarrow{d} \mathcal{E} \xrightarrow{f} \mathbb{R} \tag{1}$$

where $g \in G$ is the space of genomes, and $E \in \mathcal{E}$ is the space of phenotypes (environmental configurations), and f

is the fitness function. We will say that $d(g, E) \rightarrow E'$ is the developmental step. The running time of d and f will be written $\delta(n)$ and $\varphi(n)$, respectively, where $n = |E|$ is the size of the environment. Evaluation of such a developmental genome typically requires time $O(\delta(n) + \varphi(n))$. Since evaluation is repeated often during a machine learning optimization, ensuring fast evaluation is critical.

The obvious means of augmenting the developmental step with environmental feedback would be to re-compute the fitness at each step, using the change in values to guide growth. This sort of approach (a *global fitness-enhanced AD*, in our parlance), would require full evaluation at each developmental step, or $O(\delta(n) \cdot \varphi(n))$. Given that, in a real-world application, evaluation f often dominates an evolutionary design process, such an approach is likely to be computationally expensive. Thus, more efficient alternatives are desireable.

Imagine instead the use of a local fitness function - that is, some function that evaluates the efficacy of a design in a local neighbourhood. If we specify that the local fitness function, independent of global fitness, be computable in constant time, then augmentation of the AD (a *blind local fitness-enhanced AD*), has an unchanged worst-case evaluation time: $O(\delta(n) + \varphi(n))$.

Better still, assume further that our developmental stage is divided into steps (for instance, a discrete time which governs cell actions), and can be decomposed into a time and a cell function: $\delta(n) = \delta_{time}(n) \cdot \delta_{cells}(n)$. There are many systems which could be decomposed in this manner, such as L-Systems, parallel cellular growth systems, and iterative graph re-writers. Then, we can consider the computation of global fitness at the end of each time step, accompanied by a local fitness function which utilizes that information at the cellular level. If we require that again local fitness is constant time, our evaluation time becomes: $O(\delta_{time}(n) \cdot \varphi(n) + \delta(n))$.

In the following paper, we will use a highly simplified model of vasculogenesis as a domain of application, and two-dimensional CAs as an ADS to explore these concepts. In so doing, we will illustrate the above concepts.

IV. The Model

A. A Highly Simplified Model of a Vascular System

Our problem is as follows: Given some two-dimensional developmental environment, $E \subset \mathbb{Z}^2$, including a "start" square, we aim to 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" cells and "transport" cells. Beginning with the start cell, fluid is distributed to all transport cells in the immediate von Neumann neighbourhood 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:

$$f_{glob}(E) = \frac{1}{|E|} \sum_{c \in E} \begin{cases} 1; & \text{if } colour(c) = \text{ "normal"} \\ & \wedge\, c \text{ is "served"} \\ 0; & \text{otherwise.} \end{cases} \quad (2)$$

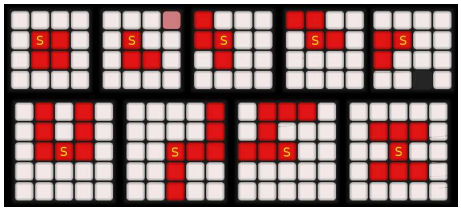

Fig. 1. Examples of optimal configurations for toroidal environments of size 4×4 and 5×5. Start cells are labelled with a yellow "S". Transport cells are bright red if connected to the start cell, very light red if not. Normal cells are white if "served", and dark grey otherwise.

where $|E|$ is the total number of cells of colour "normal" and "transport" in E. Hence, we aim to design a transport system which covers as much of the 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 $O(|E|)$.

In smaller cases, we can find the optima for f_{glob} via brute force. For example: there are 44 optimal configurations if E is a torus of dimension 4×4; 126 at dimension 5×5; and 544 at dimension 5×7. Some examples are shown in Figure 1. Note the presence of many distinct global optima, a property we believe shared at higher cardinalities.

Global optimum for any large environment is likely to be approximately $f_{glob} = \frac{2}{3}$. This is evident from considering such an optimum as a collection of Moore neighbourhoods, of which nearly all would contain three transport cells: although one or two transport cells could provide fluids to all remaining cells in a neighbourhood, it is not possible to simultaneously provide connectivity between surrounding neighbourhoods with less than three transport cells.

B. Representation of Environments

Here we define a means of representing[1] a general two-dimensional connected subset of $\mathbb{Z}^2$; Our goal is a systematic way of describing environments which can be utilized in further experiments without introducing significant bias. Further, we desire environments: which are contiguous; which envelop a specified "start" cell; in which all regions are connected via a path of minimum width; which can be described via a complexity measure (one which correlates with problem difficulty); which can be generated with a specified overall size. Finally, we desire a representation which provides a good coverage of the space of all such environments.

We will do so by introducing "barrier" cells into rectangular environments, where barriers are understood to be inactive space-fillers. The remaining region of the rectangular environment will be the environment proper. Further, the environment will be completely surrounded by barrier cells. We define the size of the environment, $|E|$, to be the number of non-barrier, or "normal" cells.

Our approach will be based on the construction of Voronoi regions in our rectangle surrounding randomly placed dots.

[1] Our thanks to Simon Harding for suggesting the use of Voronoi diagrams.

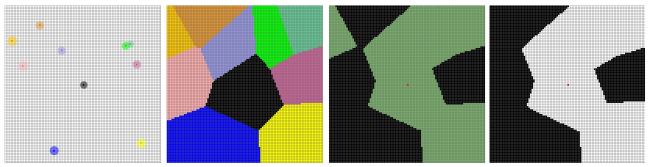

Fig. 2. An illustration of the generation of a random environment: *(from left to right)* random selection of ten dots; construction of a Voronoi diagram; labelling of regions as positive or barrier; flood-fill and fattening.

Fig. 3. Environments with dot complexities *(from left to right)* $dC = 5, 20, 50, 100, 200$.

The number of dots, dC, will control the number of Voronoi regions. These regions will be labelled either "positive" or not, where positive cells might become normal (i.e. elements of the set), and non-positive cells are labelled barriers. The probability that a region is labelled positive is controlled through a system parameter pP. The start cell is used as a guarantee of some non-empty positive central region; The positive regions surrounding the start cell will be flood-filled, and all connected cells will be labelled "normal". This normal region is then "fattened" to ensure that a path of width at least three connects the whole positive area. The process is described in more detail in Appendix A, and illustrated in Figure 2.

It is clear that, barring "fattening", any environment E can be generated through such a method, if only through setting $dC = |R|$ and specifying every cell, where R is the minimal rectangle including every non-barrier cell in E. The dot-complexity dC of the environment will be explored as a measure of environment difficulty; while not a perfect measure, as trivial environments sometimes result from large numbers of dots, informal visual inspection suggests that this measure will work in statistical questions (see Figure 3 for examples).

An important point in the generation of such environments is ensuring a consistent environmental size, which we define as $\epsilon = \frac{|E|}{|R|}$. To that end, we performed a regression of ϵ relative to various system parameters. Using the approach described above, 10 000 environments were generated using parameters chosen randomly and uniformly in the following ranges $height,\ width \in \{10, ..., 200\}$; $dC \in \{5, 2\max\{height, width\}\}$, and $pP \in [0.1, 0.9]$. Further, a normalized dot complexity was also calculated, $\frac{dC}{2\max\{height,width\}}$. For each randomly generated set of parameters, the specified environment was computed, and the proportion of normal cells ϵ was computed. It was shown that ϵ was relatively independent of all parameters (correlation coefficient $|\rho| < 0.2$) save pP ($\rho = 0.8656$, $p < 0.00001$). We calculated a linear regression with inverse $pP' = \frac{\epsilon' + 0.1694}{1.2486}$, and a 95% confidence interval of 0.1659.

Henceforth, we will write $\epsilon(E) \sim X$ to mean that with probability greater than 0.99, $\epsilon(E) \in [X - 0.1659, X + 0.1659]$. We can achieve this by making at most four attempts to generate an E in that range using our interpolated pP' value; in the unlikely cases that we do not succeed, we will substitute the closest generated E.

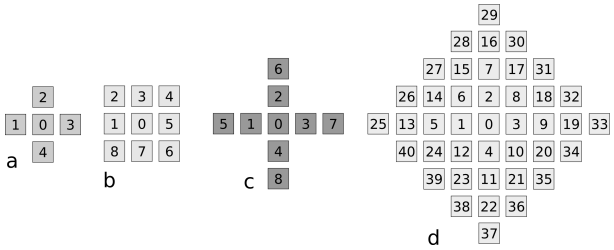

Fig. 4. Neighbourhoods: *(a)* VN-5; *(b)* Moore-9; *(c)* Cross-9; *(d)* VN-41.

C. A Highly Simplified Model of Vasculogenesis

Let our developmental environment, $E \subset \mathbb{Z}^2$, be a two-dimensional rectangular region, our alphabet be $\Sigma = \{0, 1, 2\} = \{$"normal", "transport", "barrier"$\}$, and consider our system governed by a discrete time. At initialization, our environment begins with all states of colour "0" or "2", save the unique start cell, of colour "1". At every time step, we compute f_{glob}. Time runs from $t = 0$ until the "end of time", defined either as the first t such that $f_{glob}(E_{t+1}) < f_{glob}(E_t)$, or when f_{glob} has been static for five time steps[2].

The development of the space between time 0 and the "end of time" will be referred to as vasculogenesis, or simply as growth. Below, we discuss developmental systems which control the form of vasculogenesis, interpreting the points in E as "cells", and specifying a transition function as a genome; This is quite similar to cellular automata.

V. Strategies

In this setion we outline a number of strategies for solving our vasculogenesis problem, some pre-programmed and some evolved. The first strategy will be described in some detail (Section V-A.1), and further strategies will follow the general pattern. Note that in the introduction of fitness-based strategies, we will lose the perfect parallelization typically associated with Cellular Automaton-like systems. As a result, strategies will output different results on different executions.

In outlining these strategies, we will discuss some two-dimensional neighbourhoods about a central point. These neighbourhood types, along with indexing, are illustrated in Figure 4, where the central point is indexed 0.

We will define an *non-null state* cell as one which has at least one "transport" cell in its Moore-9 neighbourhood.

Further, we define *blind local fitness* of any given neighbourhood N to be the number of cells of colour "normal" with at least one cell of colour "transport" in the immediate

[2]Originally, we intended to require strict increases in coverage, but due to use of stochastic algorithms we risked premature termination.

Moore-9 neighbourhood, divided by the size of the neighbourhood.

$$f_{blind}(N) = \sum_{c_i \in N} \begin{cases} 1 & ; \text{ if } colour(c_i) = \text{“normal”} \wedge \\ & \quad c \text{ has transport 9-neighbour} \\ 0 & ; \text{ otherwise.} \end{cases} \tag{3}$$

It measures the number of “served” cells, making the (possibly false) assumption that all transport cells in the neighbourhood are “served”.

The definition of *sighted local fitness* is slightly more involved. We begin by assuming that fitness has already been computed in the environment, and that small changes have been made to the neighbourhood since. Given such a neighbourhood N, we first mark any transport cells “served” in the original global fitness computation as “local served”. Next, we flood-fill any connected transport cells in N, marking them as also “local served”. Finally, we mark any normal cells neighbouring a “local served” transport cell as also “local served”, and compute the number:

$$f_{sighted}(N) = \sum_{c_i \in N} \begin{cases} 1 & ; \text{ if } colour(c_i) = \text{“normal”} \\ & \quad \wedge\ c \text{ has local-served} \\ & \quad \text{transport 9-neighbour} \\ 0 & ; \text{ otherwise.} \end{cases} \tag{4}$$

A. Pre-programmed Strategies

1) Random Greedy Growth Strategies (RLG and RGG): The randomized local greedy growth strategy (RLG) grows a network from a single starting cell. At each time step, all normal non-null state cells are chosen in random order. With probability $1/3$ we attempt to change the cell's state to type “transport”. State change is carried out only if local fitness increases. Note that the order of processing of cells matters, since the change of state will effect future computations of f_{blind}. Since f_{blind} is computed using a static size of neighbourhood (Moore-9), development time is not increased in the limit, leading to a time of $O(|E|^2)$.

The RLG algorithm can be written:

1: initialize E_0 with start cell
2: **for** time $t = 1$ to t_{final} **do**
3: Environment $E_t := E_{t-1}$
4: List C of all non-null state cells c in E_t
5: $randomize(C)$
6: **for all** $c \in C$ **do**
7: **if** $rand < 1/3$ **then**
8: $c' := c$
9: $colour(c') := 1$
10: Neighbourhood n_{orig} := Moore-9 neighbourhood about c
11: Neighbourhood $n_{changed}$:= Moore-9 neighbourhood about c'
12: **if** $f_{blind}(n_{changed}) > f_{blind}(n_{orig})$ **then**
13: $colour(c) := 1$
14: **end if**
15: **end if**
16: **end for**
17: **end for**
18: **return** $E_{t_{final}}$

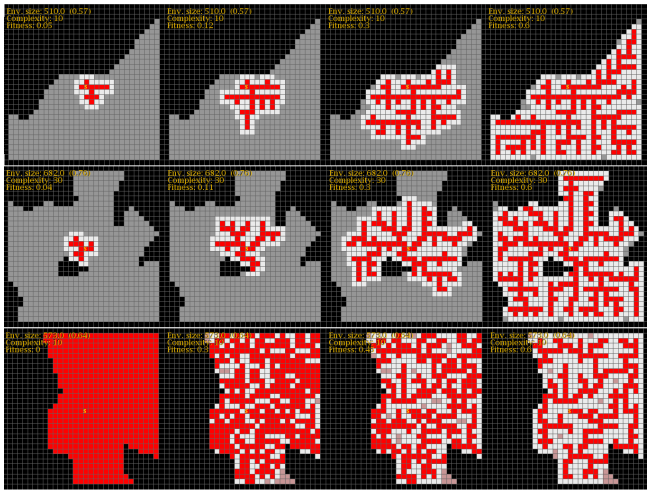

Fig. 5. Exemplars of the (*top to bottom*): local greedy growth (RLG); global greedy growth (RGG); and global knockout (RGN) strategies.

The randomized global greedy growth strategy (RGG) is identical to the randomized local greedy strategy, save that global fitness f_{glob} is used instead of local fitness f_{blind}. This increases development time to $O(|E|^3)$.

2) Randomized Knockout Strategies (RLN and RGN): The randomized local knockout (RLN) strategy begins with an environment filled with transport cells. At each time step, each transport cell (in randomized order), is considered with probability $1/3$. If local fitness f_{blind} about the point is increased by changing the cell's colour to normal, then the action is undertaken. Its running time is $O(|E|^2)$.

The RLN strategy performed very poorly, almost immediately devolving to a single served point. We've included discussion of the strategy to highlight that the f_{blind} local fitness function has a limitation: it cannot see connectivity, and hence cannot preserve it outside of a purely constructive context.

The randomized global knockout (RGN) strategy is identical to the RLN strategy except that global fitness f_{glob} is used instead of local fitness. This increases running time to $O(|E|^3)$.

Instances of the random strategies RLG, RGG, and RGN[3] are shown in figure 5.

B. Evolved Strategies

1) Cellular Automata (CA): The CA strategy is a straightforward implementation of two-dimensional CAs. Representation consists of a cell state for each possible neighbourhood, where a cell state is either “normal” or “transport”. This leads to a transition function representation of size $2 \cdot 3^{|N|-1}$, where N is the neighbourhood type used. At each time step, in parallel, each cell collects a description of its

[3]The RLN strategy is essentially a trivial version of the RGN strategy, and hence is not illustrated.

local neighbourhood, queries the transition function for an action, then changes its cell state.

2) Blind-Local Fitness Enhanced Constructive CA (BLF-CCA): The BLF-CCA strategy includes the use of the f_{blind} fitness function as a guide for cell actions. That is, with each considered cell action, the local fitness difference in the local neighbourhood of executing or not executing the action is computed, and the action is undertaken only if f_{blind} increases. Since the f_{blind} function can be computed in constant time, our running time is unchanged between the CA and the BLF-CCA strategies: $O(|E|^2)$.

In designing the BLF-CCA strategy, we chose to use a constructive CA, i.e., one in which a cell could change state from "normal" to "transport", but not vice versa. The reason for doing so involves the failure of the development of the pre-programmed RLN strategy, that is, the failure of f_{blind} strategy to adapt to the removal of cells due to its inability to recognize connectivity. Note that the use of a constructive CA removes most of the computational power of CAs generally, by ensuring that a system always ends in a stable point.

3) Sighted-Local Fitness Enhanced CA (SLF-CA): The SLF-CA strategy includes the use of the $f_{sighted}$ local fitness function as a guide. With every developmental step, global fitness is computed. Then, for every cell action, the difference in $f_{sighted}$ is computed for the original and new neighbourhoods, and the action is undertaken only if $f_{sighted}$ increases. Since we compute global fitness at each time step, and since $f_{sighted}$ is constant time, we have a running time of $O(\delta_{time}(|E|) \cdot \varphi(|E|) + \delta(|E|)) = O(|E|^2)$.

Unlike the BLF-CCA strategy, any CA could be utilized (i.e., the CAs are not constructive), widening the range of possible growth.

4) Global Fitness Enhanced CA (GF-CA): The GF-CA strategy is identical to the SLF-CA strategy, save that f_{glob} is computed 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 Figure 6.

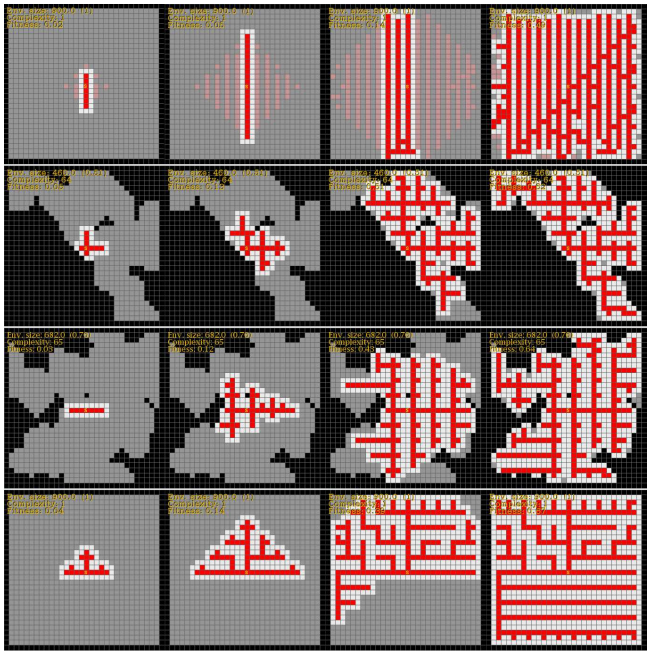

Fig. 6. Exemplars of the *(top to bottom)*: CA strategy; BLF-CCA strategy; SLF-CA strategy; GLF-CA strategy.

VI. Experiments

A. Experimental Setup

We use a typical Evolutionary Algorithm for evolution. The genome consists simply of a description of the CA transition function, that is, a list of bits corresponding to cell states associated with neighbourhood descriptions. Operators include a simple point-wise flip mutation, and an all-point recombination operator. An informal parameter search was carried out, and EA parameters maximizing the fitness of the CA strategy were used for all other strategies. These parameters were:

initial pop. size	400	*pop. size*	200
prob. mutation	0.01	*prob. crossover*	0.5
prop. elite	0.02	*tran. func. nbhd*	Cross-9
f_{blind} *nbhd*	Moore-9	$f_{sighted}$ *nbhd*	VN-41

The function $o = f_{glob}$ was used as an objective function. In the case of several of the evolved strategies, BLF-CCA, SLF-CA, and GF-CA, growth was a stochastic process. Indeed, much variance was often seen using different random seeds. Hence, for these cases, we used an altered optimization function: the genome was expressed in the same environment five times, and the optimization function o was defined to be

$$o = \sqrt[5]{\prod_{1 \le i \le 5} f^i_{glob}} \tag{5}$$

where f^i_{glob} was the global fitness of the i-th expression of the genome. A product was chosen so as to encourage consistent developmental results. Below, we report the average global fitness, defined as

$$\overline{f_{glob}} = \frac{1}{5} \sum_{1 \le i \le 5} f^i_{glob} \tag{6}$$

B. Empty Environment Experiments

A series of experiments were undertaken using the various strategies. Using an empty rectangular environment of size 30×30, each strategy was run 50 times, for 200 generations. Below, we report the mean best f_{glob} for each. Note that for BLF-CCA, SLF-CA, and GF-CA we report the mean best $\overline{f_{glob}}$.

	strategy	mean	s.d.
pre-prog.	*RLG*	0.600	0.005
	RGG	0.606	0.005
	RLN	0.041	0.031
	RGN	0.595	0.008
evolved	*CA*	0.448	0.039
	BLF-CCA	0.658	0.004
	SLF-CA	0.663	0.007
	GF-CA	0.667	0.003

Evolutionary results over time for typical runs (runs in which final fitness resembled mean) are illustrated in Figure 7.

The pre-programmed 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, while the local fitness-enhanced knockout strategy performed dismally. The latter's poor performance illustrates that the f_{blind} function works in constructive scenarios, where previous connectivity is guaranteed, while it does not when removing cells, as a small gain in local fitness can exclude a large chunk of connectivity. It should be further noted that all the pre-programmed strategies do 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, being unable to catch up to the pre-programmed strategies despite much evolution. It seems that the chaotic growth of CAs, and the associated issues with evolvability, prove too daunting to overcome in this context.

In contrast, the fitness-enhanced strategies BLF-CCA, SL-CA, and GF-CA clearly all outperformed both the standard CA strategy, and the pre-programmed 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.

C. Dependency on Problem Complexity

The above experiments were repeated, this time allowing for randomly generated environments. That is, at the beginning of each run, a particular starting environment was generated, with $\epsilon \sim 0.6$, and $dC \in [10, 75]$ chosen randomly and uniformly. For each strategy, we also computed the Spearman coefficient (ρ) between the variables dC and f_{glob}. As before, we substituted $\overline{f_{glob}}$ for the fitness-enhanced CAs. Each strategy was evaluated through fifty runs, with results summarized below:

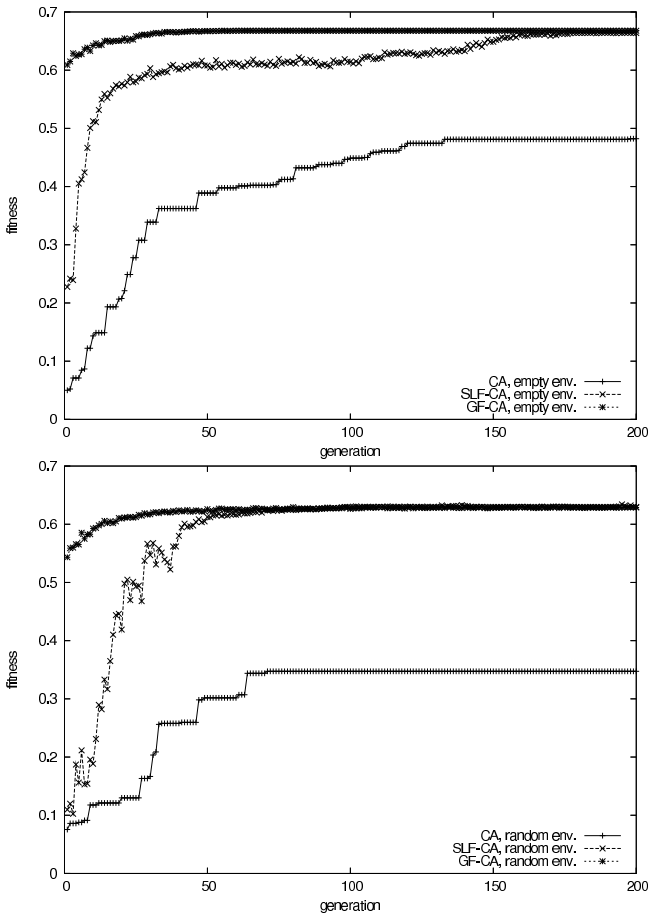

Fig. 7. Comparative plot of evolutionary success over time for *(top)* the empty environment, and *(bottom)* a random environment.

strategy	mean	s.d.	ρ
RLG	0.578	0.027	$\lvert\rho\rvert < 0.3$
RGG	0.587	0.008	$\lvert\rho\rvert < 0.3$
RGN	0.573	0.011	$\lvert\rho\rvert < 0.3$
CA	0.316	0.079	-0.608 ($p < 0.001$)
BLF-CCA	0.635	0.010	-0.603 ($p < 0.001$)
SLF-CA	0.611	0.024	$\lvert\rho\rvert < 0.3$
GF-CA	0.640	0.007	$\lvert\rho\rvert < 0.3$

The course of evolution for typical runs is illustrated in Figure 7.

All strategies performed more poorly than in the empty environment, as would be expected in a more difficult domain. The general trend amongst the evolved strategies was maintained. The CA runs indicate that the non-fitness enhanced strategy was, in fact, subject to problem complexity as measured by dC. That is, with high probability, as dC increased, the efficacy of the strategy decreased. This effect was also seen with the BLF-CCA strategy.

It is interesting to note that there is little to no recognizable dependency on problem complexity ($|\rho| < 0.2$) for all pre-programmed 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 overcome problem complexity in this context.

In these experiments, the BLF-CCA strategy outperformed the SLF-CA strategy. At first glance, it may appear that the former is simply superior to the latter, and indeed, in this simple domain of application, it likely is. However, we re-iterate that the constructive cellular automaton is a much simpler machine than the general CA, and hence, could not be expected to extend its success to more difficult problem domains.

VII. Conclusions

In this paper, we have motivated the use of a local fitness function during development, and provided a concrete example of its use in a developmental task. This re-formulation of AD allows for the use of development as a form of guided hill-climbing. This is a contrast to the usual means of including environmental information in the developmental stage, as it includes a notion of local success, rather than simply an addition of new inter-cellular signals. To the best of our knowledge, this is the first time that a concept of local fitness has been applied in the developmental stage.

In the process, we have introduced a novel means of systematically describing bounded, connected two-dimensional geometric environments, and a novel toy problem, our simplified vasculogenesis problem.

It was shown that relative to the use of evolved cellular automata, simple pre-programmed random strategies augmented with local fitness were a more effective method of solution design. Additionally, that local-fitness enhanced cellular automata could approach expected global fitness, far outperforming the non-enhanced CAs and the simple pre-programmed strategies. These local fitness-enhanced strategies could be executed without increase in asymptotic computational complexity.

Further, it was shown that, unlike CAs, (global or local) fitness-enhanced ADSs were more resistant to a measure of problem difficulty associated with our geometric environments.

References

[1] E. Coen, *The Art of Genes: How Organisms Make Themselves*. Oxford University Press, 1999.

[2] P. R. Huttenlocher, *Neural plasticity : the effects of environment on the development of the cerebral cortex*. Harvard University Press, 2002.

[3] C. J. Lowe and G. A. Wray, "Radical alterations in the roles of homeobox genes during echinoderm evolution," *Nature*, vol. 389, pp. 718–721, 1997.

[4] S. F. Gilbert, *Developmental Biology*, 8th ed. Sinauer Associates Inc., 2006.

[5] R. M. H. Merks and J. A. Glazier, "Dynamic mechanisms of blood vessel growth," *Nonlinearity*, vol. 19, pp. C1–C10, 2006.

[6] R. M. H. Merks, S. V. Brodsky, M. S. Goligorsky, S. A. Newman, and J. A. Glazier, "Cell elongation is key to in silico replication of in vitro vasculogenesis and subsequent remodeling," *Developmental Biology*, vol. 289, no. 44-45, 2005.

[7] K. Stanley, "Compositional pattern producing networks: A novel abstraction of development," *Genetic Programming and Evolvable Machines*, vol. 8, pp. 131–162, 2007.

[8] A. Ilachinski, *Cellular Automata: A Discrete Universe*. World Scientific, 2001.

[9] A. Deutsch and S. Dormann, *Cellular Automaton Modelling of Biological Pattern Formation: Characterization, Applications and Analysis*. Birkhauser, 2005.

[10] S. Camazine, J. Deneubourg, N. Franks, J. Sneyd, G. Theraulaz, and E. Bonabeau, *Self-Organization in Biological Systems*. Princeton U. Press, 2001.

[11] R. Mech and P. Prusinkiewicz, "Visual models of plants interacting with their environment," in *SIGGRAPH '96 Proceedings*, vol. 30, 1996, pp. 397–410.

[12] 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.

[13] G. Tufte, "Evolution, development and environment toward adaptation through phenotypic plasticity and exploitation of external information," in *Artificial Life XI (ALIFE XI)*, 2008.

[14] T. Kowaliw, P. Grogono, and N. Kharma, "Bluenome: A novel developmental model of artificial morphogenesis," in *Genetic and Evolutionary Computation - GECCO '04*, K. D. et al., Ed. Springer-Verlag, 2004.

[15] ——, "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*. New York, NY, USA: ACM, 2007, pp. 1037–1044.

[16] M. A. Potter and K. A. D. Jong, "Cooperative coevolution: An architecture for evolving coadapted subcomponents," *Evolutionary Computation*, vol. 8, no. 1, pp. 1–29, 2001.

[17] M. Aichour and E. Lutton, "Cooperative co-evolution inspired operators for classical GP schemes," in *Nature Inspired Cooperative Strategies for Optimization (NICSO 2007)*, 2007, pp. 169–178.

Appendix

A. Generation of a random environment

Require: Rectangular environment E, start cell s, dot-complexity dC, probability of positive pP

initialize List D of dC dots
$D[0] := s$
$D[0].positive := true$
for $dotIndex = 1$ to $dC - 1$ **do**
  $D[dotIndex]$ is a random point in E
  $D[dotIndex].positive := (rand < pP \ ? \ true \ : \ false)$
end for
compute Voronoi diagram about dots
for all Cell $c \in E$ **do**
  if $closestDot(c, D).positive = true$ **then**
    $c.positive := true$
  end if
end for
List pE is all positive $c \in E$ connected to s through flood fill
for all Cell $c \in E$ **do**
  if $c \in pE$ **then**
    $colour(c) :=$ "normal"
  else
    $colour(c) :=$ "barrier"
  end if
end for
$E' := E$
for all Cell $c \in E'$ **do**
  if c has a neighbour of colour "normal" in its Moore 9-neighbourhood in E **then**
    $colour(c) :=$ "normal"
  end if
end for
return E'

Java source code for environment generation is freely available at http://kowaliw.ca/envs.html