On the robustness achievable with stochastic development processes

Shivakumar Viswanathan
DEMO Lab, Computer Science dept.
Brandeis University, Waltham, MA-02454,USA
shiva@cs.brandeis.edu

Jordan B. Pollack
DEMO Lab, Computer Science dept
Brandeis University, Waltham, MA-02454,USA
pollack@cs.brandeis.edu

Abstract

Manufacturing processes are a key source of faults in complex hardware systems. Minimizing this impact of manufacturing uncertainties is one way towards achieving fault tolerant systems. By treating manufacturing as a stochastic development process, we characterize some of the constraints limiting the levels of robustness that can be achieved with evolution. The analysis is by introducing a novel abstraction of development as a strategic decision-making process. Using this abstraction to analyze a toy-system that simulates a process of noisy assembly, we compare the maximum robustness achievable with adaptive and non-adaptive developmental strategies. Even in this highly simplified setup, the optimal adaptive and non-adaptive genotypes reveals a significant empirical difference in their robustness characteristics. This suggests that the choice of developmental strategy and the properties of the setup are major constraints on the robustness achievable, even prior to evolution-related considerations.

1 Motivation

The evolution of fault tolerance and robustness in complex electronic and electro-mechanical systems has been an actively studied issue in Evolvable Hardware research [9, 4]. While the dominant focus has been on the behavior of systems in the presence of faults, a critical concern is the *origin* of faults themselves. A key source of faults in complex hardware systems is the manufacturing process. This is a major factor responsible for the familiar "bath-tub" shape of the plot of hardware failure rates over a system's life-span, in particular, for the high failure rate in the initial burn-in stage (or "infant mortality").

Variability in material properties, components, environmental conditions and processes is ubiquitous during manufacturing. So, two systems manufactured based on the *same design* could still differ significantly in their behavioral properties depending on the characteristics of the manufacturing process used. This is particularly problematic when the behavior of the system is the result of the interaction of a large number of interacting components and features, as with evolved designs. Given the recent interest in interfacing evolutionary design with automated manufacturing [6] and the development of technologies for autonomous construction [5], faults having a manufacturing origin are of critical consequence to the scalability and reliable functioning of automatically evolved and manufactured hardware systems.

This suggests the need for an explicit strategy to minimize (to the extent possible) the impact of manufacturing uncertainties on the reliable functioning of the constructed systems, as a way to achieve fault tolerant systems. While there are several possible approaches to this issue, our broad motivation has been the question of how this process dimension of fault tolerance can be integrated and addressed within the bottom-up, low-inductive bias methodology of evolutionary design. To this end, our approach of choice has been to consider hardware manufacture as being analogous to biological development and therefore a process amenable to evolutionary modification.

In the context of manufacturing, developmental robustness can be considered as a measure of the ability of a process to withstand *persistent stochastic variations* occurring over the duration of the process without resulting a change in the *desired function* of the constructed systems. As processes can vary in their developmental robustness, identifying a robust process defines an evolutionary search problem[10, 7, 11]. While evolution can result in a developmental process that is *relatively* more robust than others in the chosen search space, it leaves open the issue of whether such evolved process can meet levels of robustness required to be acceptable for non-trivial real-world applications. This is an important meta-consideration, prior to the algorithmic decisions about the evolutionary techniques to be used.

In our earlier empirical work, we observed that the *capabilities* provided the manufacturing setup that enabled processes to detect variation by explicit “measurement” made a notable difference in the quality of robustness obtained by evolution[11]. Here, we present a more comprehensive analysis of this initial observation by viewing development as a decision-making process. This abstraction enables a clear conceptual differentiation between adaptive developmental strategies (that explicitly detect variation and change behavior accordingly) and non-adaptive developmental strategies (Section 2). To illustrate the difference in robustness with these strategies, a toy-system that simulates a process of noisy assembly[11] is introduced (Section 3). The optimally robust adaptive genotype for this toy-system is exactly computed by treating development as a Markov Decision Problem (MDP). Even in this highly simplified setup, an empirical comparison of the optimal adaptive and (hand-constructed) non-adaptive genotypes reveals a significant difference in the absolute quality of the processes achievable in each case (Section 4).

2 Decision making in development

2.1 Ontogenic trajectories

Evolutionary search is typically defined on a phenotype set $\mathcal{P}$ with an associated fitness function $\mathbf{e} : \mathcal{P} \rightarrow \mathbb{R}$. In a design setting, $\mathcal{P}$ is treated is the set of possible designs with the fitness function providing a measure of suitability for a chosen task. So, the search problem is to find the phenotypes in $\mathcal{P}$ that maximize the value of $\mathbf{e}$, given the available resources.

A developmental phase separates the entities on which the search problem is defined (i.e. $\mathcal{P}$) from the entities on which search occurs (i.e. the set of genotypes $\mathcal{G}$)[2]. In a deterministic setting, these entities are related via the genotype-phenotype map $\psi : \mathcal{G} \rightarrow \mathcal{P}$. A genotype here can be treated as being a "recipe" for the construction of the phenotype. It is in this sense that we will speak of the relationship of the genotype to the construction process.

Let an ontogenic step take the form $\phi_{t+1} = \mu_{g,t}(\phi_t)$ where $\mu_{g,t}$ is the *ontogenic function* for a given $g \in \mathcal{G}$. So, a developmental process can be seen as being an iterative application of the ontogenic function to the phenotype produced at each time step till some stopping condition is reached. This temporally ordered sequence of phenotypic states $\phi_0 \rightarrow ...\phi_i \rightarrow \phi_j ... \rightarrow \phi_{final}$ is referred to as an *ontogenic trajectory*, where $t = 0, ...t_{final}$[1]. Functionally, the only change to the genotype-phenotype relation ψ would be in taking the expanded form $\psi' : \mathcal{G} \rightarrow (\mathcal{P}_0 \rightarrow \mathcal{P})$, where $\mathcal{P}_0 \subset \mathcal{P}$ is the set of initial phenotype states[1] and every triple $(g, \phi_0, \phi) \in \psi'$ is associated with a single ontogenic trajectory T_g.

[1]To maintain consistency with the published work on development, $\mathcal{P}_0$ is assumed to have exactly one element $\mathcal{P}_0 = \{\phi_0\}$. With $\mathcal{P}_0$ being fixed, the equivalence to ψ is straightforward as $\psi'_{\phi_0} : \mathcal{G} \rightarrow \mathcal{P}$.

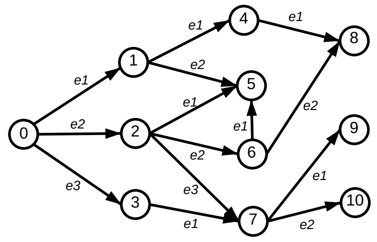

Figure 1. Ontogenic structure induced on $\mathcal{P}$

Each ontogenic trajectory can be represented as a directed graph where $T_g = (P, E)$ where $P \subset \mathcal{P}$ are the vertices, and E is the set of directed edges where an edge exists between ϕ_a and ϕ_b ($\phi_a, \phi_b \in P$) iff ϕ_a immediately precedes ϕ_b in the ontogenic process. For simplicity, we will assume that these trajectories are acyclic.

As can be seen, the cumulative effect of all the possible ontogenic processes is to induce significant structure on the phenotype set as shown in Figure 1, where an edge exists between two phenotypes ϕ_a and ϕ_b (shown by their labels a and b) iff ϕ_a immediately precedes ϕ_b for some trajectory T_g ($g \in \mathcal{G}$). The edges have a labeling relative to the vertices. This additional structure allows us to introduce a decision-making abstraction based on the notion of a *graph game*.

2.2 Ontogenic graph game

Consider a simple 1-player graph game defined as follows. We are given a finite connected directed graph $G = (V, E)$ with a special vertex designated as the *start vertex* v_0 and where there is a path from the v_0 to every vertex in V. There is a payoff function $\mathbf{e} : V \rightarrow \mathbb{R}$ that assigns a real valued payoff to each vertex.

The rules of the game are defined as follows, a token is placed at v_0 and the player picks an outgoing edge along which the token is to be moved. The token is then deterministically moved along this edge to the next vertex, and once there the player can pick another edge to move and so on. The game ends when the player cannot make any further moves or declares a halt. The value $\mathbf{e}(v)$ at the stopping vertex v is the payoff obtained by the player. The objective of the game is for the player to move the token to a vertex v_{max} associated with the highest payoff.

Now, the game defines a sequence of decision points at each vertex, where the player is faced with a set of alternatives which are the outgoing edges from that vertex, and a particular choice needs to be made. So this leads to the issue of what *strategy* the player is implementing. A *strategy* is a prescription of the choices that a player makes given the available information. This is critical as the net effect of implementing a strategy manifests itself as a play.

Here we will restrict our discussion to two representative types of strategies. A *stationary strategy* exhaustively specifies the choice to be taken at each vertex and is defined as $\pi : V \rightarrow E \cup \{halt\}$. So, if the player moves to a particular vertex v_x where $(v_x, e_x) \in \pi$ and e_x is an outgoing edge of v_x, then the player always makes the same choice e_x.[2] As a play has to halt for the player to collect a payoff atleast one of the vertices is necessarily associated with a choice to halt. The other strategy of interest is a *ballistic strategy* $\pi_b : \mathbb{N} \rightarrow E \cup \{halt\}$ where the choices made at each vertex are entirely based on the time-stamp irrespective of the specific vertex that the token is at.

This game provides an abstract interpretive framework for development. The set of vertices V can be seen as equivalent to the set of phenotypes $\mathcal{P}$, with each phenotype having a fitness value associated with it. The sequence of transitions made through $\mathcal{P}$ under the control of a particular genotype $g \in \mathcal{G}$ resulting in an ontogenic trajectory can be described as a play. So, a genotype $g' \in \mathcal{G}$ that halts at v_{max} (i.e. the phenotype having the highest fitness) is the optimal genotype as it effects a "winning play".

The notion of a strategy as used here provides a high-level description of the ontogenic behavior. However, at the mechanistic level, there is no notion of a global choice as the genotypes only specify local *actions*. When the actions deterministically result in a particular outcome, this split between *actions* and the *choices* is effectively indistinguishable. However, this is not true in real-world construction where the relation between an action and the corresponding outcome can be non-deterministic.

This non-deterministic element of the graph game can be thought of as a player having a "trembling hand" [8]. Consider the example in Figure 1 where the edges have a labeling relative to the vertices. Consider a ballistic strategy $g_1 = \{(1, e_3), (2, e_1), (3, e_1)\}$. Even though the choice prescribed by the strategy is given, with a player having a "trembling hand" there is a chance that the token is moved along a different edge. At vertex v_0 even though the choice is e_1, the probability of the token being moved along e_1 could be 0.33 (rather than 1) and the same along e_2 and e_3 (rather than 0). Suppose that this is true at all the vertices. As a result, rather than the vertex v_9 being the deterministic outcome of the play with g_1, the outcome on a given instance could be either one of v_8, v_5, v_9 or v_{10}, each possibly having very different payoffs and occurring with different probabilities. Furthermore the ballistic strategy $g_2 = \{(1, e_1), (2, e_1), (3, e_1)\}$, which results in v_8 in the deterministic case, can now produce the same outcomes as g_1 namely v_8, v_5, v_9 and v_{10}.

The refinement of the notion of a strategy for a non-deterministic development process can be described as follows. Let $\mathcal{A}$ be the finite set of actual actions. The *actual strategy* can be considered to be $\tilde{\pi} : V \rightarrow \mathcal{A} \cup \{halt\}$ in the stationary case and along similar lines $\tilde{\pi}_b : \mathbb{N} \rightarrow \mathcal{A} \cup \{halt\}$. The key difference as compared to the deterministic case is the probabilistic nature of the *state-transition function*. The state transition function is defined as $\mathcal{T} : V \times \mathcal{A} \rightarrow \Pi(V)$ giving for each phenotype/vertex and action a probability distribution over the possible choices ($\mathcal{T}(v, a, v')$ is the probability that the outcome of action a when in state v results in the state v').

Here, we can see that the stationary strategy possesses an *adaptive capability* in that it can respond differently depending on the specific phenotypic state at each point in time. However, the ballistic strategy is non-adaptive in this sense as it has a hardwired set of actions that are performed independent of the situation. *What is the impact of this difference on the robustness characteristics achievable*? An experimental setup to evaluate this question is described next.

3 Experimental setup

3.1 System description

The toy system described in this section instantiates a version of the "trembling hand" conception of development as described in the previous section.

The system is a tiling machine modeled as a gantry robot that is restricted to movement along the X-axis as shown in Figure 2. The machine has a head that can be moved under programmable control to locations in the workspace identified by it's X coordinate. The atomic operations (i.e. the genetic "actions") performed at the head are a **release** operation to release tiles individually. The workspace is a square partition of a two dimensional plane such that it can perfectly accommodate $M \times M$ identical square tiles without any gaps. The machine has an explicit spatial existence so it cannot move through tiles. A finite number of tiles are available to the machine which can be released individually at the head under programmable control.

Here, the set of actions $\mathcal{A} = \{1, ...M\}$, where each $x \in \mathcal{A}$ implies the action of releasing a tile at coordinate x. The set of all legal tile configurations in the workspace is treated as being the phenotype set $\mathcal{P}$. So the configuration of tiles in the workspace at time t, is the equivalent of the state of the embryo ϕ_t. With the execution of the strategy g, the tile configurations also change resulting in a walk through the ontogenic fitness landscape.

The physics governing the behavior of a tile on being released at the tile-head is similar to the Tetris game. There is

[2]A *non-stationary strategy* is one where the choices are indexed by a time stamp $\pi_t : V \times \mathbb{N} \rightarrow E \cup \{halt\}$ i.e. the choice made at a particular vertex depends both on the specific vertex as well as the point in time during the play at which the vertex is reached.

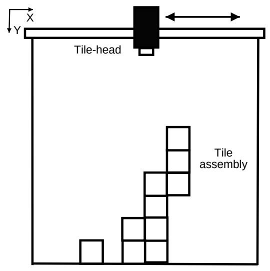

Figure 2. Tiling machine in workspace

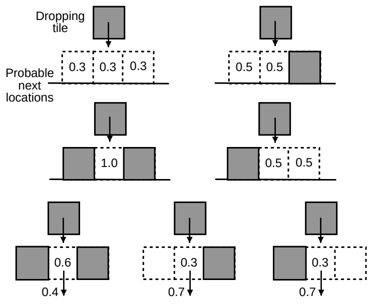

Figure 3. Tile interference physics

a constant velocity "diffusion" acting downward along the $Y+$ direction such that a tile released at a particular location moves at the rate of one tile length per time step. The stochasticity arises from the interference when a released tile comes in contact with other tiles or with the edges of the workspace and continues till the tile comes to rest.

The interference is modeled as occurring as defined in Figure 3. The possible positions of a descending tile at the next time step are shown as squares with dotted outlines and the numbers indicate the probability of moving to that position. The particular values chosen are arbitrary. The dark line indicates a tile or a wall at that periphery and the tile cannot move through it.

3.2 The fitness function

A behavior evaluation function $\mathbf{e} : \mathcal{P} \to \mathbb{R}$ is defined as $\mathbf{e}(\phi) = \sum_{i=1}^{n}(d(T_i, A) + (d(T_i, B))^2$, where $\phi \in \mathcal{P}$, T_i is the location of the i^{th} tile in ϕ, and $d(p, q)$ is the non-linearized version of the Manhattan distance between two points p and q in the workspace where $d(p, q) = |x_p - x_q| + |y_p - y_q|$ if $|x_p - x_q| \leq M/2$; and $d(p, q) = M/3 + |y_p - y_q|$ otherwise.

In words, the value $\mathbf{e}(\phi)$ is the sum of the distances (as defined by d) of every tile in a tile-configuration ϕ from two pre-chosen points A and B in the workspace. The points chosen here are $A = (1, M)$ and $B = (M, M)$ where M is the length of each side of the workspace. These points correspond to the lower left and lower right corners of the workspace in Figure 2.

The fitness of a configuration is related to the presence of specific features of the configuration. Configurations that have tiles concentrated close to $X = M/2$ axis, and away from the $Y = M$ edge (i.e. having a "T" shape) would tend to have higher values as compared to one where the tiles are randomly distributed around the workspace.

4 Comparison of strategy properties

4.1 Results

In this system, a ballistic strategy is equivalent to a list of X-coordinates that are executed in sequence by releasing a tile at those locations. A stationary strategy takes the form $\tilde{\pi} : \mathcal{P} \to \mathcal{A} \cup \{halt\}$, where there is an action specified for every phenotype state that can occur.

Each strategy g, whether ballistic or stationary, is associated with $\Pi(P)$, where $P \subset \mathcal{P}$ is the set of final phenotypes produced by the strategy and $\Pi(P)$ is the probability distribution over P obtained as a result of the strategy. In order to compare their robustness properties, the *expected fitness*[3] of the outcomes is used as a combined measure of robustness and fitness. The comparison is of the optimal stationary strategy and the optimal ballistic strategy that can produce the highest fitness phenotype in $\mathcal{P}$.

The problem of finding the optimal stationary strategy for this formulation is similar to that of finding the optimal stationary policy for a *Markov Decision Problem (MDP)*. An MDP can be described as a tuple $\langle \mathcal{P}, \mathcal{A}, \mathcal{T}, R \rangle$ where $\mathcal{T}$ is the state transition function, and R is the *value function* $R : \mathcal{S} \times \mathcal{A} \cup \{halt\} \to \mathbb{R}$ which specifies the expected immediate "reward" gained by taking each action in each state ($R(s, a)$ is the expected reward for taking action a in state s)[3].

Here we use the *value-function iteration algorithm* to obtain the optimal policy for an MDP. Value function iteration determines the optimal value function for the problem by repeatedly iterating through the phenotype states and updating the value function till it converges. As it involves an enumeration of the entire set $\mathcal{P}$, we focus on a tractable case where $M = 5$ and the number of tiles available $N = 10$. With these settings, $|\mathcal{P}| = 80915$.

On running the value iteration algorithm, the optimal stationary strategy $\tilde{\pi}_{optimal}$ (i.e. policy in the context of MDPs) was obtained by identifying the actions at each step that corresponds to the highest expected reward as specified by the value function. Due to the small size

[3]The expected fi tness of a strategy g is given by $E_g = \sum_{i=0}^{N} p_i \mathbf{e}(\phi_i)$, where $\phi_i \in P$ and p_i is the probability with which ϕ_i is produced as the outcome.

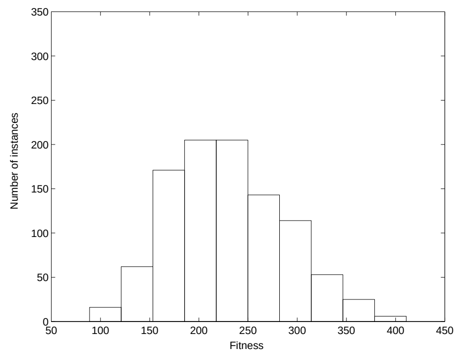

Figure 4. Fitness of phenotypes constructed with $\tilde{\pi}_{optimal}$ (1000 runs)

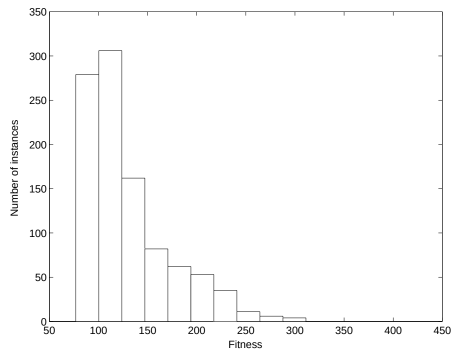

Figure 5. Fitness of phenotypes constructed with $\tilde{\pi}_{b-optimal}$ (1000 runs)

of the problem,the ballistic strategy $\tilde{\pi}_{b-optimal}$ was constructed by hand and the sequential list of actions is $[3, 3, 3, 3, 4, 3, 2, 1, 4, 5, halt]$.

Figures 4 and 5 show histograms of the fitness values of the final phenotypes produced over 1000 instances of executing strategies $\tilde{\pi}_{optimal}$ and $\tilde{\pi}_{b-optimal}$ respectively. Over these 1000 instances, the range of fitness values obtained with $\tilde{\pi}_{optimal}$ was $[88.440, 410.889]$ and $[77.000, 311.222]$ with $\tilde{\pi}_{b-optimal}$, with the expected fitness being 225.870 and 129.450 respectively.

While the ballistic sequence performs the same set of actions independent of the intermediate phenotypes obtained, the stationary strategy is sensitive to these differences. As compared to the single fixed action sequence of the ballistic strategy $\tilde{\pi}_{b-optimal}$, over these 1000 runs there were a total of 469 different sequences of actions with $\tilde{\pi}_{optimal}$. This suggests that a key reason explaining this observed difference in robustness characteristics is related to the adaptive capability of the stationary strategy.

4.2 Discussion

So in this simplistic example, we can see that the *results* of evolution through a space $\mathcal{G}$ where every strategy is strictly ballistic (as in [7]) could be both qualitatively and quantitatively different in their robustness characteristics as compared to evolution through a space of adaptive strategies for the same problem. In general, it provides us with the intuition that an *adaptive* capability i.e. (a) the ability to extract relevant information from the current state, and (b) to allow this informative to inform the choice of action, can play a fundamental role in obtaining construction processes that are acceptably robust. However, implementing an adaptive strategy comes with several challenges.

The stationary strategy is idealized with respect to a key issue, namely, that the player has *full information* about the current state. In the context of the graph-game, *full information* means that the player knows exactly which vertex the token is on at every step. In contrast, a scenario of *partial information* would arise if the player cannot unambiguously distinguish whether the token is on vertex v_x or vertex v_y. If the graph was such that each vertex is colored with either one of two colors in C where $C = \{black, white\}$ then when a token is moved to a particular vertex, rather than it's unique label, the only information available to the player is the color of the vertex. In this case, a stationary policy is drastically reduced in size taking the form $\tilde{\pi}_{partial} : C \rightarrow \mathcal{A} \cup \{halt\}$. Rather than being able to uniquely identify each vertex, the granularity of the information is now limited to identifying whether a vertex belongs to the partition V_{black} or V_{white} ($V = V_{black} \cup V_{white}$ and $V_{black} \cap V_{white} = \emptyset$), and hence it can perform a total of only two possible actions independent of the number of possible actions available.

One of the most critical constraints in real-world construction is that the current phenotypic state is only partially observable due to limits on the sensory capabilities available to uniquely identify the phenotype state at a given time. In the tiling machine, we assumed that each of the 80915 possible states could be uniquely identified enabling the strategy to perform a unique action in each case. If we were to consider how this identification of state could be achieved mechanistically say with a sensor attached to the tile-head, it would involve being able to determine whether a tile exists at each of the $10 \times 10 = 100$ positions in the entire workspace at each step as it is not known *a priori* which positions could be occupied and which may never be occupied. Furthermore, from an information theoretic standpoint, being able to achieve this distinction using the bare minimum of $log_2(|\mathcal{P}|) \approx 17$ measurements would involve substantial knowledge – both about the structure of the workspace and the structural properties of the possible phenotypes in order to make only relevant measurements. Added to this problem of state identification, the constraints on whether the tiling head could move would greatly limit the measurements that could be performed, making a unique identification of every phenotype physically impossible in this system.

Furthermore, even though adaptive developmental strategies hold the promise of achieving increased levels of robustness, in order that this can scale with the increasing complexity of the systems to be constructed firstly requires a manufacturing setup that can enable such an adaptive capability to be achieved. As the dimensions of variability that are relevant to the functioning of a system can change from one system to another, the problems of measurements are not fixed. In the absence of a manufacturing system that could possibly itself be reconfigured and retrofitted to meet these changing requirements, could limit the scalability of robustness with increasing design complexity achievable in a fixed setup.

5 Conclusions

We have presented a conceptual framework to engage a class of engineering issues associated with the evolution of reliable construction processes. The conception of development as a process of strategic interactions involves a number of simplifying assumptions neglecting issues such as the impact of simulation fidelity, and also does not address how such strategies may be implemented and evolved. However, the contribution is in providing a principled body of engineering intuitions to integrate the manufacturing issues into the evolutionary design of functional, fault tolerant systems. By framing the issue in terms of decision making under uncertainty, we have identified a number of issues that need to be addressed in developing the system capability for reliable construction. A key aspect of this capability is the ability to extract state information to reduce the uncertainty in the outcomes of construction. This raises a number of challenging questions that need to be addressed toward achieving a fully autonomous design and construction capability.

References

[1] P. Alberch, S.J. Gould, G. F. Oster, and D. B. Wake. Size and shape in ontogeny and phylogeny. *Paleobiology*, 5(3):296 – 317, 1979.

[2] P.J. Angeline. Morphogenic evolutionary computations: Introduction, issues and example. In John R. McDonnell, Robert G. Reynolds, and David B. Fogel, editors, *Evolutionary Programming IV: Proceedings of the Fourth Annual Conference on Evolutionary Programming*, pages 387–401. MIT Press, March 1995.

[3] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. Planning and acting in partially observable stochastic domains. *Artifcial Intelligence*, 101:1–45, 1998.

[4] D. Keymeulen, A. Stoica, R. Zebulum, and V. Duong. Results on the fitness and population based fault tolerant approaches using a reconfigurable electronic device. In *Proc. of the 2000 Congress on Evolutionary Computation*, pages 537–544, Piscataway, NJ, 2000. IEEE Service Center.

[5] E. Malone and H. Lipson. Functional freeform fabrication for physical artificial life. In *Proceedings of the Ninth Int. Conference on Artificial Life (ALIFE IX)*, pages 100–105, 2004.

[6] J. B. Pollack, H. Lipson, G. Hornby, and P. Funes. Three generations of automatically designed robots. *Artificial Life*, 7(3):215–223, 2001.

[7] J. Rieffel and J. Pollack. The emergence of ontogenic scaffolding in a stochastic development environment. In *Proceedings of the 2004 Genetic and Evolutionary Computation Conference*. Springer Verlag, 2004.

[8] R. Selten. Re-examination of the perfectness concept for equilibrium points in extensive games. *International Journal of Game Theory*, 4:22–55, 1975.

[9] A. Thompson. Evolutionary techniques for fault tolerance. In *Proc. UKACC Int. Conf. on Control 1996 (CONTROL'96)*, pages 693–698. IEE Conference Publication No. 427, 1996.

[10] P. Van Remortel, T. Lenaerts, and B. Manderick. The robustness of small developed SBlock circuits using different clocking schemes. In *The Third NASA/DoD Conference on Evolvable Hardware*, pages 26–35. Jet Propulsion Laboratory, California Institute of Technology, IEEE Computer Society, 15-18 July 2002.

[11] S. Viswanathan and J. Pollack. Towards an evolutionary-developmental approach for real-world substrates. In *Proceedings of the ninth international conference on the simulation and synthesis of living systems (Artificial Life IX)*, pages 45–50. MIT Press, 2004.