Evolving phenotypic networks in silico

Paul François
Ernest Rutherford Physics Building, McGill University, 3600 rue University, H3A2T8 Montreal, QC, Canada

ABSTRACT

Evolved gene networks are constrained by natural selection. Their structures and functions are consequently far from being random, as exemplified by the multiple instances of parallel/convergent evolution. One can thus ask if features of actual gene networks can be recovered from evolutionary first principles. I review a method for in silico evolution of small models of gene networks aiming at performing pre-defined biological functions. I summarize the current implementation of the algorithm, insisting on the construction of a proper “fitness” function. I illustrate the approach on three examples: biochemical adaptation, ligand discrimination and vertebrate segmentation (somitogenesis). While the structure of the evolved networks is variable, dynamics of our evolved networks are usually constrained and present many similar features to actual gene networks, including properties that were not explicitly selected for.

In silico evolution can thus be used to predict biological behaviours without a detailed knowledge of the mapping between genotype and phenotype.

© 2014 The Author. Published by Elsevier Ltd. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/3.0/).

1. Introduction

Like any complex emergent process, evolution combines dynamics at different spatial and temporal scales, and for this reason can be challenging to study and model mathematically. Microevolution corresponds to changes of allele frequencies in a population, over relatively “short” time-scales, and population genetics has long been the central mathematical theory to study microevolution [1]. Recent real-time experimental studies have also advanced our understanding of microevolution, e.g. long-term evolutionary experiments in the lab [2,3], artificial selection of complex mechanisms (such as bacterial altruism [4]) or observation of fast evolving systems (like the flu [5]).

Macrōevolution, evolution of high order structures over long time-scales, is more challenging to study. It is of course still impossible to observe experimentally and thus can be studied only indirectly. Most data come from retrospective studies of genomes and fossils, having evolved over 4 billions years. Full access to ancestral phenotypes and measures of ecological pressures are impossible so that macroevolutionary mechanisms for apparition of complex features (such as full-blown organs or signalling pathways) remain speculative. As a consequence, very different views
coexist: for instance, while many biologists (like Stephen Jay Gould [6]) think that evolved structures are historically contingent, others (like Simon Conway-Morris [7]) have used spectacular examples of convergent evolution to argue that solutions found by evolution are much more constrained than usually thought. Modern experimental attempts include the study of evolutionary history in conjunction to development (“evo-devo”), supported by genomic studies [8].

But just like population genetics is the central theory underlying microevolution, a quantitative theoretical framework would be useful for macroevolutionary studies. In particular, one question arising is the nature of constraints on evolvable biological functions: given a complex phenotype, can we use some mathematical theory to predict anything on the underlying gene networks? The issue is that we do not have (yet) a proper formalism to answer such questions: among other problems and despite recent advances (see e.g. [9–11]) the nature of the mapping between genotype and phenotype is still an open question. For this reason, we turn to computational approaches and propose a generic in silico evolution procedure to “predict” what kind of networks can evolve to perform a given biological function [12,13]. In the following, I first describe our method and then discuss three interesting case-studies.

2. Network implementation and algorithm philosophy

In this section, I summarize how we model gene networks and their simulated evolution.

There are two levels in the algorithm: the individual level where genotypes and phenotypes of individuals are defined and computed, and the population level, where evolution is performed.

2.1. Individual level

In our approach, an individual genotype is a mathematical object encoding dynamics of a gene network. Networks consist in bipartite graphs. The first category of nodes is interacting components, typically proteins or DNA sequence. They are themselves connected to the second category of nodes, corresponding to interactions. A grammar of possible interactions is predefined, accounting for various biochemistry, such as transcription, transcriptional regulations, phosphorylations, protein–protein interactions. A network behaviour (i.e. its phenotype) is modelled using ordinary differential equations. We use classical biochemical kinetics to account for the various interaction, e.g. mass-action laws for protein–protein interactions, or Hill functions for transcriptional interactions [14].

To be more specific, let us consider one example, similar to one of the adaptive network evolved in [15]. A full-blown representation of the network is displayed in Fig. 1A, and a simplified representation of the same network in Fig. 1B. There are three proteins (subsequently called $S_0, S_1, S_2$) that we call ‘Species’. regulatory module, DNA are nodes used to model regulatory and coding sequence of gene $S_i$. Finally PPI is an interaction node corresponding here to a complexation between $S_0$ and $S_1$ into $S_2$.

To this graph correspond differential equations. Equations are automatically generated by the algorithm to account for the interactions. For node $S_1$, the regulatory module and DNA part here simply encode a default basal transcription rate $\rho$. The PPI interaction adds a non linear forward interaction term $\gamma S_1 S_0$, for complex formation, and a linear backward term $\alpha S_2$ for complex dissociation. Finally, we assume that all species have a linear degradation or dilution rate. For this case, the complete set of differential equations for $S_1$ and $S_2$ thus is

\[ \dot{S}_1 = \rho - \delta_1 S_1 - \gamma S_1 S_0 + \alpha S_2 \]

(1)

\[ \dot{S}_2 = \gamma S_1 S_0 - (\alpha + \delta_2) S_2 \]

(2)

All parameters in these equations are randomly chosen and selected by the algorithm.

In the present case, there is no equation for $S_0$ because it is an external Input, with a prescribed dynamics (here a sequence of steps of random heights). Integration of networks dynamics under control of this Input is performed. Fig. 1 illustrates dynamics of this network for $\delta_1 = 0$. This makes the Output variable $S_2$ adaptive, i.e. after a change of Input value, its values changes before returning to its initial value. This adaptive response can be quantified in various ways, for instance by measuring the deviation from the baseline or by quantifying how the stationary value of the Output depends on the stationary value of the Input. These quantities can be used to define a coarse-grained phenotype. From this phenotype, a fitness or scoring function is computed by the algorithm and is later used for selection (see below for a more detailed description of evolution of adaptive behaviour corresponding to Fig. 1).

2.2. Population level

Our algorithm works very much like actual evolution and other evolutionary algorithms: (1) it takes a population of genotypes; (2) computes their phenotype and fitness as indicated above; (3) selects and mutates networks; and (4) iterates this process over as many generations as desired.

Selection is based on the network fitness. We run the simulations in a very elitist mode. At each generation, the worst half of the networks (based on the fitness) is discarded. Then the best half is ordered, kept, duplicated, and the duplicated half is mutated. To ensure some population mixing even among the best networks, we also systematically add some small random component in the fitness.

Mutations consist in random modifications of the genotype, via changes of either parameters or network topology (addition or removal of nodes). It is important to stress at this stage that individual networks can grow with time, which is different from classical genetic algorithms where genome size is fixed. On the one hand, this prevents any simple implementation of genetic cross-over, but on the other hand, network growth opens up the possibility of dimensionality increase in phase-space and of evolution of new combinatorics that could be crucial to implement new complex dynamics.

Given our pre-defined grammar of interactions, each possible evolutionary move is systematically computed at each generation for each network, and actual mutations are randomly drawn. Individual mutations are assumed to be Poisson processes, with a fixed pre-defined rate. Typically, we choose rates so that the most probable move is a change of kinetics, then second most probable move is removal of nodes, and least probable move is addition of new interactions. This fits the idea that most evolutionary moves are neutral or deleterious, and that addition of new function should be a priori rare.

The time and nature of the next mutation for a given network is chosen using a Gillespie algorithm [16]. An evolutionary time therefore needs to be defined. As networks grow, generation time is dynamically changed so that the average number of mutations per generation per individual is fixed (currently taken to one). This implements an analogue of the “Drake’s rule”, the idea that the mutation rate inversely scales with genome size [17]. This also prevents uncontrolled explosion of network size that would naturally occur given the combinatorial explosion of possible interactions as networks grow.

2.3. Evolution of specific biological functions

A key aspect of any evolutionary computation is the choice of scoring (or fitness) function. By analogy with energy minimization
in physics and contrary to what is usual in biology, we assume in
the following that fitness actually is minimized.

It is clear both from intuition and from (computational) experi-
ence that the choice of fitness crucially influences success of the
evolutionary algorithm. There are first problems related to the
shape of the fitness landscape itself, which are generic for any
optimization problems. If the fitness landscape ressembles a “golf
course”, with only one global minimum but no local slope in
network space, evolutionary computation is no better than simple
random exploration (Fig. 2A). Another problem would be to choose
a fitness giving rise to a “glassy” fitness landscape, with many local
minima where the algorithm can be stuck without converging to
functionally relevant minima (Fig. 2B). Evolutionary computation
will be efficient only if the choice of fitness allows for incremental
evolution in network space, with some (possibly weak) local slope
converging towards an interesting minimum (Fig. 2C).

Second, considering a specific biological function, it is not nec-
essarily obvious how to mathematically encode what we want to
select for in a generic and almost parameter free way. For instance,
image we want to select for a network “patternning” an embryo. In
some obvious cases, we might intuitively see that a network creates
a richer pattern than another network, but it might not be obvious
which quantity to compute to rank such networks.

Finally, it happens very often that biological functions actually
optimize some different constraints, we might need to combine
these into a single number, and we might be concerned about the
trade-offs between them.

To circumvent these difficulties, our fitnesses are coarse-grained
and optimize generic abstract properties. Next section provides
specific examples. They allow for easy neutral evolution, and incre-
mentally change even far from the abstract behaviour we select for.
In some instances we also studied evolutionary trade-off between
different fitnesses. Our algorithm can also keep several fitnesses to
perform Pareto evolution [18].

2.4. Network simplification

Once interesting behaviours have been selected, correspond-
ing network structures can be quite convoluted and potentially
intractable. This is a known difficulty in evolutionary simulations
called “code-bloat”. To understand the “core” working network,
we usually run new evolutionary simulations initialized with the
network of interest, in a mode where the only possible mutations
are nodes removals. The final network of this simplification step
thus performs the same function but is usually much simplified
compared to the initial network. This helps to get a better under-
standing of the dynamics: with less variables and less interactions,
the crucial components of the network are more easily identified
and mathematical analysis is therefore easier.

3. Examples

In the following, I present three interesting examples of suc-
cessful evolutionary computations. We have used computational
evolution for many other problems, including multi stability and
oscillators [12], embryonic timing and patterning [19], tempera-
ture compensation for circadian clocks [20], Pareto evolution of
networks with asymmetric response time [18].

3.1. Biochemical adaptation

Many biological pathways are adaptive: when a constant Input
changes suddenly to a different value, an Output transiently devi-
ates from its baseline before coming back. Examples include
bacterial chemotaxis [21], light adaptation in retinal rods [22], or
signalling pathways such as ERK [23]. Fig. 1B illustrates such a typi-
cal behaviour, with the Input being Species 0 (green) and the Output
being Species 2 (blue).

From a computational evolution standpoint, a proper adaptive
system can be characterized by two main coarse-grained features:
its core should come back to its baseline while there should be a signifi-
cant deviation after an Input change. Thus it is natural to define two
quantities: \( \Delta O_{\text{max}} \), which is the final deviation from baseline after an
Input change, and \( \Delta O_{\text{max}} \), the corresponding maximum deviation
from the baseline (Fig. 3A). We can then use these two quantities to
select for an adaptive behaviour: any combination minimizing
\( \Delta O \) and maximizing \( \Delta O_{\text{max}} \) should give rise to a bona fide adaptive
system.

These quantities are completely agnostic to the precise quan-
titative properties of adaptation, such as time-scale and direction
of deviation. This is a desirable aspect of fitness functions to select
for qualitative behaviours, because it does not constrain the precise
dynamics of the response and therefore can potentially select for
many different solutions.

Practically, to get unbiased results, we generated randomly
many different Input profiles to compute averaged values of
\( \Delta O_{\text{max}}, \Delta O_{\text{max}} \). In [15], we defined several fitnesses combining
these two quantities, and used them to evolve adaptive sys-
tem. Examples include minimization of \( F_1 = \Delta O_{\text{max}} \Delta O_{\text{max}} \) or of

Fig. 1. Example of a network definition and its numerical integration. (A) Full graphical representation of the network, with all nodes and interactions implemented in the code. (B) Simplified representation of the same network, with corresponding dynamics. Inverted triangle is Inputs, triangle is Outputs, ‘ppi’ indicates protein–protein interaction. Output Species 2 is adaptive with respect to changes of Input Species 1.
Examples of good and bad fitnesses. (A) Fitnesses rewarding only some "perfect" network give no clue about direction of evolution. (B) If fitnesses are too complex, the evolutionary simulation might get stuck in some local optimum and never reach a satisfying solution. (C) Only smooth incremental fitnesses will converge efficiently.

Reconstruction of evolutionary trajectories shows indeed that evolution optimized these two functions at the same time (Fig. 3C), by tuning this single parameter within the network, thus converging very rapidly.

3.2. Ligand discrimination: adaptive sorting for immune recognition

In another instance [24], we evolved biochemical adaptation in combination with a 'proofreading' process [25,26] as a sub-function of a more complex behaviour. This is an illustration of how simple biological features can simultaneously evolve and combine via selection of complex phenotype.

Immune cells have to detect efficiently infected cells: very schematically this is done through interaction of T cell receptors (R) with pMHC ligands (L) at the surface of Antigen Presenting Cells. These ligands can be characterized by their binding time $\tau$ with T cell receptors. One hypothesis is that the nature of the response of T cells is then completely controlled by the relative position of $\tau$ with
Fig. 4. Adaptive sorting. (A) Representation of adaptive sorting network. LRI stands for Ligand–Receptor Interaction. "P" is phosphorylations interactions. For phosphorylations, substrates correspond to blue arrows and kinase to red arrows. Ligands are associated to receptors with binding time \( \tau \). When ligands dissociate, \( C_i \) is assumed to be immediately dephosphorylated. (B) \( C_i \) concentration as a function of \( L \) for different values of \( \tau = 5s \) (red) and \( \tau = 3s \) (blue). Note how flattened the response is with respect to ligand concentration. Dashed line indicate threshold for response used for panel (C): for \( \tau = 3s \), there is no response irrespective of ligand concentration, while for \( \tau = 5s \) only a couple of ligands trigger response. (C) Response in the Ligand–\( \tau \) space.

respect to a threshold \( \tau_c \), independently from ligand \( L \) concentration (the so-called “Lifetime dogma” [27]). If \( \tau < \tau_c \), T cells do not trigger immune response, even in presence of saturating ligands. On the contrary, if \( \tau > \tau_c \), T cells do trigger immune response even when only a couple of ligands are present. This is actually puzzling: a priori, from thermodynamics, we would naively expect that more abundant ligands would “compensate” for smaller binding time \( \tau \). So how can the \( \tau \) dependency vanish?

Altman-Bonnet and Germain [28] were the first to propose and validate experimentally a model based on interlocked feedbacks combined to a kinetic proofreading backbone to explain this process. While working perfectly, their model is quite complex and mathematically intractable. In [24] we exhibited a simpler tractable model of this system, showing in particular how a negative feedback loop acting on a kinetic proofreading cascade could give an output of the network \( O(L, \tau) \) with a flattened, non-monotonic dependency of \( O \) as a function of \( L \). But this is only thanks to computational evolution that we could formulate and solve this problem in its simplest form [29].

Let us assume that some output variable \( O(L, \tau) \) is responsible for detection of the immune response. Then, irrespective of \( L \), the steady state value of \( O \) for different \( \tau \) should be different enough so that T cells can sort out between ligands with \( \tau > \tau_c \) and ligands with \( \tau < \tau_c \). We recast this as an information theory problem [30], assuming that the system optimizes a classical quantity called the mutual information \( I(O, \tau) \) between \( O \) concentration and value of \( \tau \). Intuitively, mutual information between two variables quantifies how one of the variable can “predict” the state of the other.

We then ran evolutionary simulations using directly (minus) mutual information as a fitness [29]. The algorithm quickly converged to a rather simple solution (at least compared to other models) that we called “Adaptive sorting”, displayed in Fig. 4A.

Schematically, ligand and receptor interact to give a complex \( C_0 \). Then \( C_0 \) gets modified by a kinase \( K \) to give the output \( C_1 \). If kinase \( K \) is unregulated, then \( C_1 \) would be a rather simple function, linear in \( L \) and increasing with \( \tau \), as expected from biochemistry. But the key feature of the adaptive sorting network is that \( C_0 \) actually deactivates its own kinase \( K \) (e.g. by phosphorylating it into an inactive form \( K' \)), so that \( K \) concentration is inversely proportional to both \( L \) and \( C_0 \). As a consequence, the product \( K \). \( C_0 \) is itself independent from \( L \), effectively implementing biochemical adaptation for high enough ligand concentration \( L \).

However, there still is a remaining \( \tau \) dependency coming from the dissociation of the ligand from the receptor. Steady state value of \( C_1 \) as a function of \( L \) and \( \tau \) is illustrated in Fig. 4B: for a fixed \( \tau \), \( C_1 \) is an almost flat function of \( L \), just like an adaptive system, but the adaptive value of \( C_1 \) clearly depends on \( \tau \). Thus, by choosing a threshold on \( C_1 \), the cell is able to sort out ligands based on their relative position to \( \tau_c \), effectively solving the detection problem, as shown in Fig. 4C. One interesting aspect of this model is that, because of the action of \( C_0 \) on the variable responsible for adaptation \( K \), it naturally displays the well-known biological effect of ligand antagonism: in presence of high concentrations of ligands with small \( \tau < \tau_c \), kinase \( K \) is inhibited by \( C_0 \) so that low concentration of ligands with \( \tau > \tau_c \), no longer triggers response. Antagonism in the basic adaptive sorting network is actually much stronger than in immune detection, to the point that immune detection would be essentially impossible as soon as weaker ligands are present. We therefore ran new simulations in presence of high concentrations of ligands with small \( \tau \) to solve this strong antagonism problem [29]. More elaborate networks evolve from an adaptive sorting backbone, adding cascades of proofreading interactions, and giving interaction networks with close resemblance to actual ones [24]. So computational evolution here not only is able to select for simple generic models, but also is able to complexify them after addition of evolutionary constraints, to give networks closer to reality.

3.3. Striped pattern in a growing embryo

Previous examples illustrate how in silico evolution works well to select for small networks performing simple biochemical computations, by direct optimization of functionals defining the fitness. In this section, we show a completely different example where a full complex “developmental” pathway evolve under a rather simple evolutionary constraint.

Virtually all multicellular animals have a well-defined body plan, with specialized cells of different types in different parts of the body. In particular, many metazoan body plans are segmented [31]: metameric structures first form at the embryonic stage (segments in insects, somites in vertebrates) and can later specialize to form more complex structures (such as vertebrae).

Segment formation is first characterized by the striped expression of specialized segmentation genes in embryos along the antero-posterior axis. We used computational evolution to ask the following question: what kind of gene networks can give rise to stripe formation [13]? While there were already attempts to use an evolutionary approach with reaction–diffusion based models [32], we were interested in figuring out what happens in a cell-autonomous context, where, presumably, stripe formation is coupled to embryonic growth as observed in most animals.

Mathematically, we reduced the problem to the following question: consider different cells indexed by \( n \)-corresponding to the position along the antero-posterior axis-, and assume cell \( n \) is exposed to an Input signal \( \text{Species 0} \) in Fig. 5) for a time linear in \( n \) (cells with higher position are exposed for a longer time). Assume there is an Output gene called \( E \), and consider the steady state concentration \( E(n) \). Intuitively, a “striped” pattern for \( E \) means that \( E(n) \) essentially “goes” up and down as a function of \( n \) (see Fig. 5, second
Fig. 5. Typical steps in the simulated evolution of a patterning system (A) generation 7, (B) generation 140, (C) generation 360, and (D) generation 490. From [13,33]. First column is the topology of the network, interactions are purely transcriptional, green arrows indicate activations, red arrows repressions. Second column is the steady-state profile of such a network as a function of position (higher positions are exposed to the input for a longer time). Fitness is the number of ups and downs of the output (Species 1) profile, so that respective fitnesses are: (A) 1, (B) 2, (C) 3, and (D) 19. Third column is behaviour of the network as a function of time in a cell exposed to the output for 100 units of time in panels (A)–(C) and 270 in panel (D), illustrating the increase of complexity in the dynamics.
column). This is the most generic way to define a striped pattern, and we used it as a fitness function. More precisely, we defined two threshold concentrations $E_{\text{min}}$ and $E_{\text{max}}$, and we simply computed an integer $F$, corresponding to the number of times $E(n)$ crosses each of these thresholds as a function of $n$. We use $-F$ as a fitness to minimize.

This simple coarse-grained fitness gives rise to an extremely elegant evolutionary dynamics, which is highly reproducible in different evolutionary simulations.

First, evolution finds a simple way to define two regions of output expression: a bistable system evolves, so that the Output can stabilize at two different values once the Input gene is gone. This gives rise to one single transition from down to up ($F = 1$), where the Output is fully activated only after some exposure to the input (Fig. 5A). Then, the system evolves a “stripe” module: another gene (Species 2 in Fig. 5B) appears and represses the Output shortly after it has been activated in most of the cells. This creates two transitions at steady state: up, and then down, so that $F = 2$. Then, the very same evolutionary process is iterated: another repressor evolves, repressing Species 2 so that the repression on the Output is relieved and the fitness would be 3 (Fig. 5C). Essentially, the system incrementally increases $F$ by 1 by adding a new repressor. Strikingly however, after a couple of repressors (or even after the first one if there is some transcriptional delay), evolution finds a much more parsimonious way to make suddenly many stripes: it uses the chain or evolved repressors to close a feedback loop and create an oscillation (Fig. 5D). This oscillation ensures that the Output goes up and down many times temporally and spatially at steady state, increasing considerably the fitness (see also online video [34] for another example from [13]).

There are many lessons from this evolutionary simulation. It is first a striking example where selection for a developmental pattern in a very broad sense gives rise to evolution of complex sub-dynamics (bistability, oscillation) that were not selected explicitly, but, instead, spontaneously appear and eventually combine to optimize the scoring function. Second, evolution of each of these complex sub-dynamics is incremental: each bifurcation (in the dynamical system sense) is associated to an increase of $F$ [33]. As a consequence, the solutions found are actually modular, both in phase space and in network space, with an oscillator built on top of a bistable system. Finally, some properties come “for free”: oscillation naturally gives rise to stripes of equal size, as observed in nature.

Such a segmentation clock in vertebrate has been observed for a long time [35,36], and has been recently discovered in some insects [37]. Our simulations predict a specific bifurcation diagram from an oscillatory dynamics to a bistable system [38], which has not been checked experimentally yet.

4. Discussion

Computational evolution is a very efficient tool to select for networks performing specific behaviours. We propose that the fitness function should be coarse-grained for evolution to be efficient. We can actually relate this to macroevolution itself: our goal is not to select for characters specific to a species – which would lead to a glassy landscape –, but rather for general features characteristic of entire branches of the tree of life (such as segment formation in metazoans). Our evolved networks are often modular, a generic feature of many complex networks [39]. Here modularity either comes from the addition of multiple evolutionary constraints, or simply from the incremental property of evolution. Addition of proofreading steps to fight against antagonism is an example of the former [24], while evolution of a clock upstream of a bistable system for segmentation is an example of the later.

The fact that we could identify actual networks similar to the one we evolved in the computer validate the approach, but still, one could fear that coarse-grained fitness might neglect important biological details. One should first point out that, selecting for more detailed behaviour (such as precise positioning of stripes in the segmentation case) might actually impinge evolution. Relating to our previous macroevolution analogy: one should not numerically evolve very specific features without evolving some simpler ancestral feature to be later specialized. Indeed, we can speculate that if our approach of coarse-grained fitness is correct, we should be able to recapitulate actual evolutionary pathways [38] (in a similar way to the pioneering work of Nilsson and Pelger for eye evolution [40]). Second, for a given simulation, we typically obtain many possible networks, but in all cases, it is possible to coarse-grain the results themselves to isolate some core working behaviour. These models we obtain via evolution are often simple, with a few number of variables, but still nicely recapitulate observed phenotypes we select for.

In particular, in most instances of evolved networks, we can recover or predict some non trivial observed features that were not specifically selected for. Adaptive network of Fig. 1 verifies the ubiquitous Weber’s law in adaptive systems [41]: the maximum value $O_{\text{max}}$ is proportional to the ratio between two consecutive values of Inputs. Evolved networks for immune response display antagonism in a very similar way to actual networks [24]. Segments from the example of Fig. 5 are roughly of equal size like as a consequence of the clock control, as observed in nature. In another developmental context, we also recovered the “posterior dominance” property of Hox genes from simple mutual information optimization [19]. This raises interesting theoretical questions: in some cases, such as immune antagonism, these features might be absolutely necessary for the realization of the biological function. This means that computational evolution, by defining properly phenotype to select for, is able to recover hard-wired design principles of biological networks.

In other cases, such as segment formation, we can actually imagine alternative networks without these features, which suggests that these constraints come from the evolutionary process itself, similar to externalities characterizing the phenomenon of “path dependence” in economy or history [42]. Such constraints on more complex phenotype might not be easy to figure out and computational evolution could help uncover them. In both cases, this finally suggests that, despite their relative simplicity, the class of “phenotypic models” we evolved might have a very good predictive power, potentially short-cutting the recurring problem of genotype to phenotype mapping in theoretical biology.

Acknowledgments

The study was supported by Natural Science and Engineering Research Council of Canada (NSERC), Fond de Recherche Québecois Nature et Technologie (FRQNT), and Human Frontier Science Program (HFSP).

References


[34] Segmentation model - YouTube. http://www.youtube.com/watch?v=p90q88JW8c


