SIGEVOlution

newsletter of the ACM Special Interest Group on Genetic and Evolutionary Computation

Volume 5
Issue 4

in this issue

Editorial

GECCO-2012 deadline is less than three months away (January 13, 2012!). It is time to go back to that incredible idea you had for so long but you did not have time to finalize into a paper yet! You know you do not want to miss the next GECCO! You know we would miss you and you also know that you would miss meeting your friends, attending those exciting presentations, those great tutorials and the funny discussions. And, if it helps, I promise this time I won't sing!

This new issue of SIGEVOlution completes the fifth volume and brings you two new exciting articles. The first one, by Jeff Clune and Hod Lipson, presents the algorithm behind EndlessForms.com, their website for the interactive evolution of 3D objects. The scientific purpose of EndlessForms.com is to let researchers explore what complex designs can be produced when evolution is powered by a generative encoding based on developmental biology. Its practical and more fun purpose is to allow people to create unique physical objects easily while seeing artificial evolution in action. And the best part is that the evolved objects can be published and brought to life using 3D printers. Yes, exactly like the ones shown the cover!

The second article by Rebecca Schulman, who gave an extraordinary keynote at GECCO-2011, provides a brief overview of her applications in the new field of structural DNA nanotechnology to modularly design nanoscale components that together can be assembled into a system for self-replicating a new form of chemical information, and thus for evolving a new type of chemical sequence.

As usual, my due thanks go to the people who made this possible: Jeff Clune, Hod Lipson, Rebecca Schulman, Reinaldo Bergamaschi, Cristiana Bolchini, and board members Dave Davis and Martin Pelikan.

The cover shows a set of artifacts evolved with the technology behind the EndlessForms.com website and produced using a 3D printer.

Pier Luca
November 3, 2011

SIGEVOlution Volume 5, Issue 4

Newsletter of the ACM Special Interest Group on Genetic and Evolutionary Computation.

Contributors to this Issue

Contents

ISSN: 1931-8499

Evolving 3D Objects with a Generative Encoding Inspired by Developmental Biology

Jeff Clune & Hod Lipson, Department of Mechanical and Aerospace Engineering, Cornell University

This paper introduces an algorithm for evolving 3D objects with a generative encoding that abstracts how biological morphologies are produced. Evolving interesting 3D objects is useful in many disciplines, including artistic design (e.g. sculpture), engineering (e.g. robotics, architecture, or product design), and biology (e.g. for investigating morphological evolution). A critical element in evolving 3D objects is the representation, which strongly influences the types of objects produced. In 2007 a representation was introduced called Compositional Pattern Producing Networks (CPPN), which abstracts how natural phenotypes are generated. To date, however, the ability of CPPNs to create 3D objects has barely been explored. Here we present a new way to create 3D objects with CPPNs. Experiments with both interactive and target-based evolution demonstrate that CPPNs show potential in generating interesting, complex, 3D objects. We further show that changing the information provided to CPPNs and the functions allowed in their genomes biases the types of objects produced. Finally, we validate that the objects transfer well from simulation to the real-world by printing them with a 3D printer. Overall, this paper shows that evolving objects with encodings based on concepts from biological development can be a powerful way to evolve complex, interesting objects, which should be of use in fields as diverse as art, engineering, and biology.

1 Motivation and Previous Work

The diversity, complexity, and function of natural morphologies is awe-inspiring. Evolution has created bodies that can fly, run, and swim with amazing agility. It would be desirable to harness the power of evolution to create synthetic physical designs and morphologies. Doing so would benefit a variety of fields. For example, artists, architects and engineers could evolve sculptures, buildings, product designs, and sophisticated robots. Evolution should be especially helpful in the design of complex objects with many interacting parts made of non-linear materials. In such challenging problem domains, evolution excels while human intuition is limited. Being able to evolve sophisticated morphologies also furthers biological research because it enables the investigation of how and why certain natural designs were produced. Evolving 3D objects is thus worthwhile both as a basic science and for its innumerable potential applications. This paper describes how 3D shapes can be evolved and then transferred to reality via 3D printing technology (Figure 1).

Previous research in digital morphological evolution has typically involved encodings that were either highly biologically detailed, or highly-abstract with less biological accuracy. The former camp frequently simulates the low-level processes that govern biological development, such as the diffusing *morphogen* chemicals and proteins that determine the identity of embryonic cells [Bongard and Pfeifer, 2001, Eggenberger, 1997, Miller, 2004]. While this approach facilitates studying the mechanisms of developmental biology, the computational cost of simulating chemistry in such detail greatly limits the complexity of the evolved phenotypes.

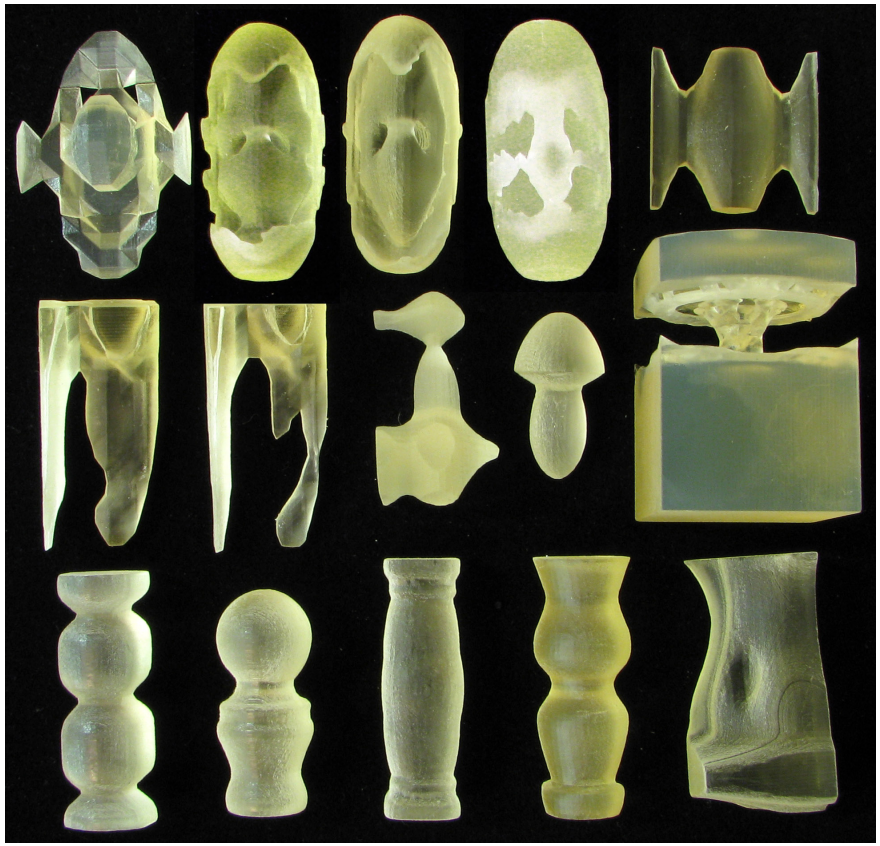


Fig. 1: Examples of evolved objects that were transferred to reality via a 3D printer.

The most complex forms typically evolved in such systems are simple geometric patterns (such as three bands) [Miller, 2004] or groups of shapes resembling the earliest stages of animal development [Eggenberger, 1997].

The second camp employs high-level abstractions that enable the evolution of more elaborate forms with many parts, but these abstractions tend not to reflect the way that organisms actually develop [Wolpert and Tickle, 2010, Bentley, 1996]. An example is Lindenmayer Systems (L-Systems), which iteratively replace symbols in strings with other symbols until a termination criteria is reached [Lindenmayer, 1968, Hornby et al., 2003]. While L-Systems can reproduce a wide variety of organismal shapes, especially those of branching plants, they do not model plant developmental processes [Wolpert and Tickle, 2010]. Another example is the work of Sims (1994), who evolved morphologies that resembled some biological creatures, although with an abstract encoding based on parameterized recursion that does not resemble natural developmental processes [Sims, 1994].

A third option is possible, wherein a high-level abstraction is based on the developmental processes that give rise to natural forms. An example of this approach is Compositional Pattern Producing Networks (CPPNs) [Stanley, 2007], which are used to evolve 3D objects in this paper and are described in Methods. Two groups have previously evolved 3D objects with CPPNs, although neither conducted an open-ended exploration of 3D objects. One group evolved CPPN objects that were composed of variable-sized spheres and were evaluated on two tasks: falling [Auerbach and Bongard, 2010b] or moving rapidly [Auerbach and Bongard, 2010a]. Most of the evolved forms resembled clubs. A second group evolved soft-bodied robots to move quickly [Hiller and Lipson, 2010]. These studies demonstrate that CPPNs can create functional shapes, but leave open the question of what types of 3D objects CPPNs can produce with fewer constraints and without specific objectives.

2D pictures are evolved with CPPNs on picbreeder.org, where humans perform selection [Secretan et al., 2011]. The complexity and natural appearance of the resulting images often support claims regarding the legitimacy of CPPNs as an abstraction of biological development [Stanley, 2007]. A demonstration in 3D would significantly strengthen these claims, however, because the natural world is 3D.

It is possible that CPPNs are unable to frequently make sensible forms with the added difficulty of another dimension, and when objects must be one contiguous unit (which aids in transfers to reality). A recent paper by Bánsági Jr et al. (*Science* 2011) highlights the need to verify that generative encodings that produce complex patterns in 2D also can do so in 3D. By evolving CPPN objects in the natural 3D setting, this paper conducts a critical test of the hypothesis that generative encodings based on *geometric* abstractions of development capture some of the complexity-generating power of natural morphological development. Doing so also provides a visually intuitive testbed for studying how variants of such generative encodings behave. It also reveals the utility of CPPNs as a representation for 3D object design.

2 Methods

2.1 Compositional Pattern Producing Networks

Compositional Pattern Producing Networks (CPPNs) abstract the process of natural development without simulating the low-level chemical dynamics involved in developmental biology [Stanley, 2007]. Cells (and higher-level modules) in natural organisms often differentiate into their possible types (e.g. heart or spleen) as a function of where they are situated in geometric space [Wolpert and Tickle, 2010].

Components of natural organisms cannot directly determine their geometric location, so developmental processes have evolved to create gradients of chemicals and proteins called morphogens that organismal components use to figure out *where* they are and, thus, *what* to become [Wolpert and Tickle, 2010]. For example, in many animals the anterior-posterior and dorsal-ventral axes are specified by maternally provided morphogen gradients. Embryonic genes then construct more complicated geometric patterns of morphogens as a function of these simpler gradients. Downstream genes can construct additional pattern as a function of any of the patterns already created, enabling the production of patterns of arbitrary complexity [Wolpert and Tickle, 2010].

CPPNs abstract this process by allowing similar geometric patterns to be composed of other geometric patterns, but represent the patterns mathematically instead of via diffusing morphogens. To replace maternally-provided gradients, the experimenter provides the initial gradients. Final patterns output by the CPPN determine the attributes of the phenotypic components at different geometric locations.

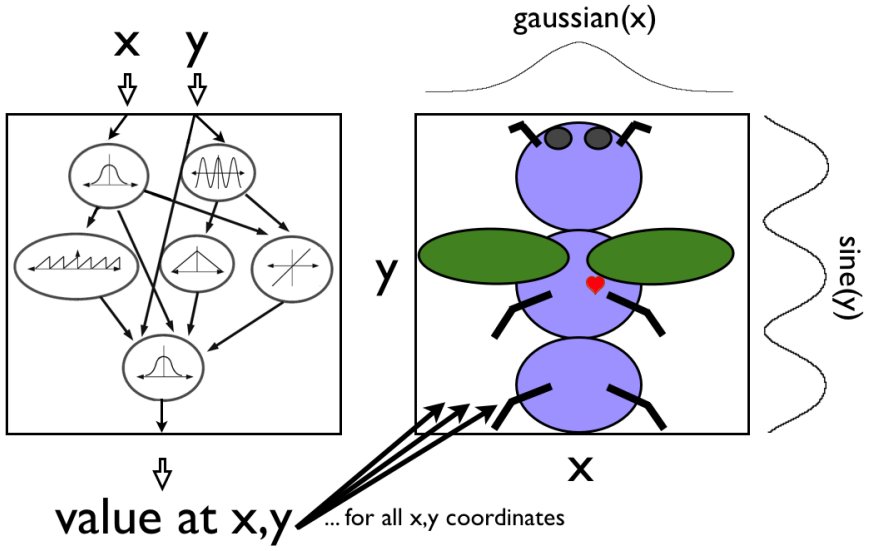


Fig. 2: CPPNs combine mathematical functions to create regularities, such as symmetries and repeated modules, with and without variation. Adapted from Stanley (2007).

For example, two-dimensional pictures could be encoded by iteratively passing the coordinates of each pixel on a canvas (e.g. $x = 2$, $y = 4$) to a CPPN genome and having the output specify the color or shade of each pixel (Figure 2).

Each CPPN is a directed graph in which every node is itself a single function, such as sine or Gaussian. The nature of the functions can create a wide variety of desirable properties, such as symmetry (e.g. a Gaussian function) and repetition (e.g. a sine function) that evolution can exploit. Because the genome allows functions to be made of other functions, coordinate frames can be combined. For instance, a sine function early in the network can create a repeating theme that, when passed into the symmetrical Gaussian function, creates a repeating series of symmetrical motifs (Figure 2). This process abstracts the natural developmental processes described above [Wolpert and Tickle, 2010].

The links that connect and allow information to flow between nodes in a CPPN have a weight that can magnify or diminish the values that pass along them. Mutations that change these weights may, for example, give a stronger influence to a symmetry-generating part of a network while diminishing the contribution from another part.

Variation is produced by mutating or crossing CPPNs. Mutations can add a node or change weights. The default set of allowable functions for CPPNs in this paper are sine, sigmoid, Gaussian, and linear, although we also experimented with additional functions (see Results). The evolution of the population of CPPN networks occurs according to the principles of the NeuroEvolution of Augmenting Topologies (NEAT) algorithm [Stanley and Miikkulainen, 2002].

The NEAT algorithm contains three major components [Stanley and Miikkulainen, 2002]. (1) It starts with small genomes that encode simple networks and complexifies them via mutations that add nodes and links to the network. This complexification enables the algorithm to evolve the network topology in addition to its weights. (2) NEAT preserves diversity via a fitness-sharing mechanism that allows new innovations time to be tuned by evolution before competing them against more optimized rivals. (3) crossover utilizes historical information in a way that is effective, yet avoids the need for expensive topological analysis.

2.2 Encoding 3D Objects with CPPNs

To evolve 3D objects, inputs for the x, y, and z dimensions are provided to a CPPN. Additional gradients can be provided, which may bias the types of objects produced (see Results). A *workspace* (maximum object size) is defined with a *resolution*, which determines the number of voxels in each dimension. In this paper there are 10 voxels in the x and z dimensions and 20 in the y (vertical) dimension. The x, y, and z value of each voxel are iteratively input to a CPPN, and voxels are considered full if the CPPN output is greater than a threshold (here set to 0.1), otherwise the voxel is considered empty. The 3D voxel array is then processed by the surface-smoothing Marching Cubes algorithm [Lorensen and Cline, 1987]. A normal is provided for each vertex when visualizing the objects in OpenGL, a graphics technique that further smooths the surface. These two smoothing steps enable high-resolution CPPN objects to be visualized without prohibitive computational costs.

This algorithm for encoding 3D objects is a more straightforward extension of how CPPNs encode 2D pictures [Stanley, 2007, Secretan et al., 2011] than another algorithm for evolving 3D objects with CPPNs, which included growth over time and limited shapes to collections of attached spheres of different sizes [Auerbach and Bongard, 2010b, Auerbach and Bongard, 2010a].

2.3 Selection Mechanisms (Fitness Assignment)

We evolve images with interactive evolution and target-based evolution. During interactive evolution the user (here, the first author) views N rotating objects (here, 15) and selects a champion, which receives a fitness of 1000. The user can also reward additional organisms that receive a fitness of 500. To avoid uninteresting objects, those that are not chosen, yet have voxel counts between 10% and 90% of the maximum number possible, are given a fitness of 100. The remaining objects are given a fitness of 1. For target evolution, the fitness is the percent of voxels that matched the target object. To magnify differences in fitness values, all fitness scores serve as an exponent to a large constant $c = 2000$ to produce the final fitness value. The parameters are identical to a previous work [Clune et al., 2011], except mutations were allowed to be larger (MutationPower = 2.5).

3 Results and Discussion

3.1 Interactive Evolution

We study interactive evolution because it allows an open-ended exploration of the design space of objects CPPNs can produce. Additionally, interactive evolution avoids the greedy nature of target-based evolution, potentially allowing it to access more interesting objects [Secretan et al., 2011, Lehman and Stanley, 2008]. A drawback of interactive evolution is that it is subjective, but science should not abandon such a useful tool simply because it is subjective. While user preferences bias the types of objects selected, the encoding has to be able to produce such objects in the first place in order for them to be selected. Different encodings will bias the types of patterns evolved [Clune et al., 2011], meaning that interactive evolution can inform us about the biases and expressive power of the encoding.

Figure 3 shows example objects from different generations during a run of interactive evolution. The geometric patterns become more complex over generations, which reflects the property of complexification built into NEAT [Stanley and Miikkulainen, 2002].

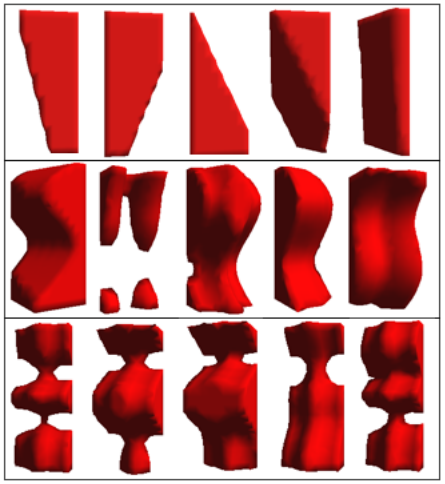


Fig. 3: Representative objects from different generations of a single run of interactive evolution. From top to bottom, rows display individuals from generations 1, 15, and 33.

Figure 4 displays a few of the interesting objects discovered in different runs, some of which had different inputs and parameters (described below). It is important to note that these objects were chosen from a small number of runs performed by one person, each of which was limited to tens or perhaps a few hundred generations. It is noteworthy that such recognizable 3D forms emerge in such a small sample size. These 3D objects should not be held to the same standard as pictures from picbreeder.org, where hundreds of users have published thousands of images after performing over 150,000 evaluations across hundreds of generations [Secretan et al., 2011].

The objects in Figure 4 exhibit many properties that are desirable both for studying morphological evolution and harnessing it for engineering or artistic purposes. The objects are frequently regular, a property which is important in engineering and for evolvability [Lipson, 2007, Clune et al., 2011]. An important regularity is symmetry, which is evident with respect to different dimensions in many of the objects. For example, all of the objects in generation 33 of Figure 3 are highly left-right symmetric, and objects b7 and b8 in Figure 4 exhibit left-right and top-bottom symmetries. Another useful regularity is repetition, which occurs frequently in the evolved objects (e.g. the top-right object in Figure 3).

A further beneficial property is exhibiting regularity with variation [Stanley and Miikkulainen, 2003, Lipson, 2007, Clune et al., 2011]. For example, Figure 4b1 has a motif that appears like an animal head, but is repeated in different sizes and with other subtle variations. Symmetric patterns with asymmetric variations can also be observed, such as in Figure 4a8 and Figure 4b6.

It is important to note that humans often select regular, symmetrical shapes, which increases their frequency in interactive evolution. That said, biology and engineering also often reward regularity. Additionally, it has been shown that when CPPNs generate artificial neural networks that control robots in target-based evolution, the neural wiring patterns are often regular, including symmetries and repeated themes [Clune et al., 2011], demonstrating that CPPNs produce regularities even without humans performing selection.

Most importantly, the evolved objects often look similar to natural forms or engineered designs, revealing that CPPNs can produce the types of objects we are interested in designing and studying with synthetic morphological evolution. Humans can only select such such familiar forms if an encoding tends to produce such designs, which has not been the case for most previous generative encodings. People often describe Figure 4a2 and 4a3 as faces, 4a4 as a Jack-o'-lantern face, 4a5 as an animal figurine, 4a6 as an African statue of a human, 4a7 as a human female stomach, 4a8 as a human female torso, 4b1 and 4b4 as animals, 4b2 and 4b3 as elephants, 4b5 as a human head and shoulders, 4b6 as a horned mask, and 4b7 and 4b8 as spaceships. Some also describe 4b7 as a butterfly. People describe other objects as interesting art, even though they do not resemble any specific natural or human design (e.g. Figure 4a1). Such objects can potentially spark artistic ideas for new forms. The fact that the shapes consistently evoke human and natural designs demonstrates the expressive power of the CPPN encoding to produce interesting 3D objects.

An additional important property is that the offspring of the 3D CPPN objects are similar to their parents, but are varied in interesting ways. Some encodings lack this property in that mutations have dramatic effects, rendering most offspring very different from their parents, which hinders evolvability [Stanley and Miikkulainen, 2003].

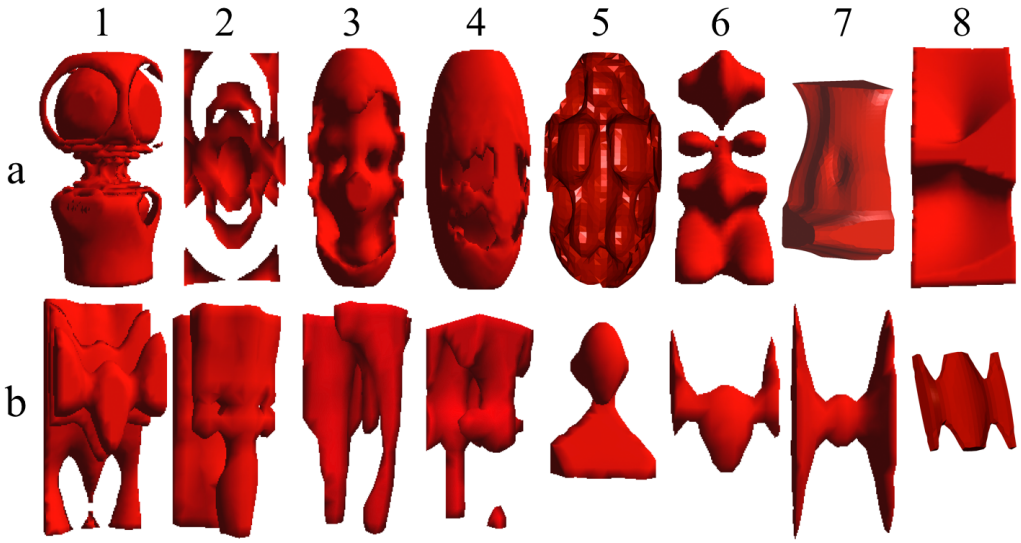


Fig. 4: Example objects evolved with CPPNs via interactive evolution.

For example, Figure 4b4 is the child of Figure 4b3, and Figure 4b2 is their close relative. All three are consistently described as animals, yet are interesting variations on the animal theme. For example, only a single generation of genetic changes between Figure 4b3 and Figure 4b4 transformed what appears like an elephant with a trunk into something resembling an elephant with warthog tusks. A different variant of Figure 4b3 that thickened the trunk can be seen in Figure 1 (center row, left), which is next to a printed copy of Figure 4b3. Moreover, Figure 4b3, its relative in Figure 1, and Figure 4b2 all evoke elephants, but they are quite different objects, suggesting that the CPPN has captured some fundamental aspects of the elephant concept that it expresses in different ways.

Some of the geometric complexity in the genome is not visible in these 3D phenotypes because a threshold determines the presence or absence of a voxel. In contrast, picbreeder pictures have a continuum of outputs in grayscale and color, which adds to their complexity. Pre-thresholded geometric information could be useful, however, to make colored 3D objects, or to have objects with multiple materials (e.g. the soft-robot equivalent of muscle and bone).

To test whether the types of objects produced could be biased by the CPPN inputs and parameters, we performed multiple runs of interactive evolution with varying conditions. We initially provided only x, y, and z values for each voxel. Even with this minimal information, regularities such as symmetries and repeating themes were common (Figure 3), which is expected in a generative encoding with symmetric and repeating genomic functions. The objects in this setup seemed to require more generations before they became interesting, and usually did not appear like objects floating in space, but instead bordered the workspace wall.

We then added the distance from center as an input to the CPPN, which picbreeder also has (in 2D) [Secretan et al., 2011]. This information more frequently created rounded objects centered in space. Because the distance-from-center function took the normalized values in each dimension, and the y (height) dimension was longer, an egg-shaped motif was common (Figure 5, left three). All of the objects in Figure 4 have this input. Preliminary experiments with other inputs also revealed interesting biases in the resulting objects (not shown), suggesting a rich area of research regarding how best to bias CPPNs with seed gradients.

To date, no published results explore how patterns differ when recurrence is allowed in CPPN genomes. We enabled recurrence and discovered that the resulting patterns are qualitatively different in that they tend to include fractal patterns. For example, branching patterns emerged, such as an object resembling a tree (Figure 6, left) and another evoking the vascular system (Figure 6, center). Like with fractals, the complexity is often concentrated at the surface boundary, producing a jagged surface effect (e.g. Figure 6, right). Objects with recurrent genomes were much more likely to have small, separated pieces floating in space.

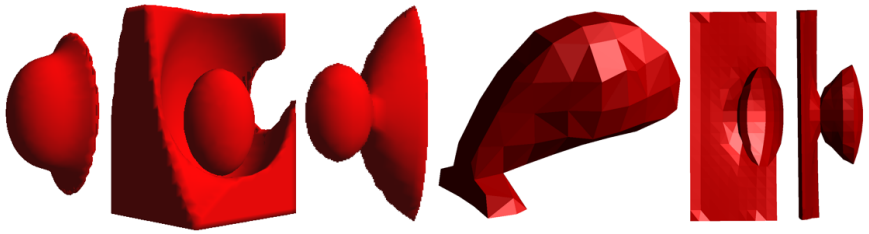


Fig. 5: Objects evolved with a distance-from-center input (left three), which frequently featured egg-shape motifs, and objects evolved with an expanded set of genome functions (right three). The rightmost two images show different angles of the same object. Facets in the right three objects result from a close zoom and because, for illumination, normals are provided for facets instead of vertices.

Another interesting parameter of CPPNs is the set of possible genomic node functions. No research published to date has tested different function sets on the same problem to understand how CPPN patterns are affected by this parameter. Visual domains such as 3D objects are a helpful place to start such explorations because of the intuition they provide. We added a square, cosine, and sign-preserving square root function and performed additional runs. Objects in these runs tend to be more complex in earlier generations, and seem to involve both rounded and sharp edges. Figure 4b7 and the rightmost three in Figure 5 are example objects evolved with this expanded genomic node function set.

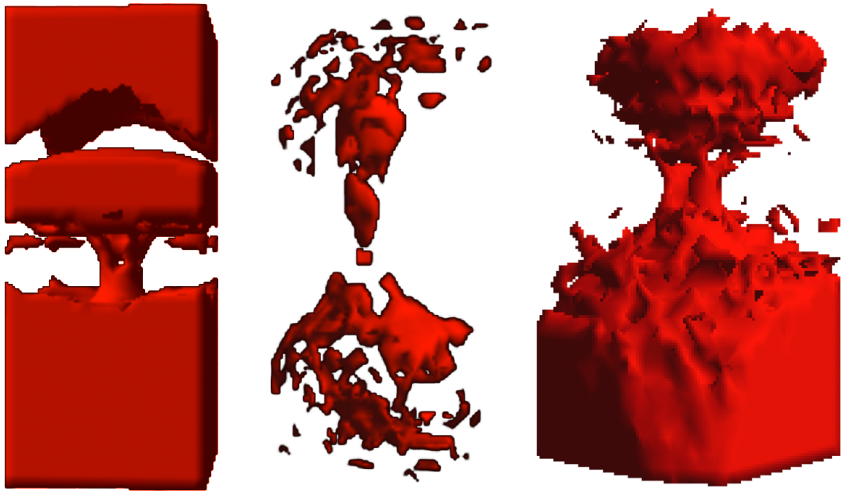


Fig. 6: Example objects with recurrent genomes.

3.2 Target-based Evolution

A second way to explore the capabilities of CPPNs is to challenge them to produce a target object. Knowing how CPPNs perform in 3D in target-based evolution is helpful for numerous reasons. Initially, it serves as a preliminary test of how CPPNs might perform on more open-ended, yet still target-based problems, such as evolving robot morphologies to perform certain tasks (e.g. locomotion). Additionally, biologists would benefit if they could repeatedly evolve various morphologies to study whether certain developmental strategies for constructing 3D geometric patterns arise frequently. Finally, target evolution allows an artist or engineer to explore objects that are similar to a target object, yet differ in interesting ways (similar to how Figure 4b4 and Figure 4b2 result from slight permutations to the genome of Figure 4b3). Finally, target-based evolution is much faster, enabling an exploration the effects of different parameter settings, which can inform interactive evolution.

The target object for this paper is shown in Figure 8a. It consists of four partially-overlapping spheres, with the outer two halved by workspace bounding box. This target has round shapes that are different from the egg-shaped motif facilitated by the distance-from-center input, providing a test of whether such a related input improves performance. Each treatment has 20 runs with a population of 150 for 1000 generations, unless otherwise specified.

The baseline treatment featured only x, y, and z inputs and the default set of genome functions. The best performing object in each run captures the long cylindrical shape of the target, but most attempts at rounded edges are imperfect combinations of straight-line functions. All runs except one failed to carve much material away between the spheres. An average of 90.8% ($\pm$ 0.003 SE) of voxels are matched (Figure 7), but the target object is not identifiable until about $\geq$ 93% of voxels are matched. As such, the small differences in fitness between the treatments in Figure 7 represent substantial differences in whether the target object is recognizable. Interestingly, one outlier run in this treatment performed much better than the rest (with 94.6% of voxels correct). It features rectangular approximations of spheres (Figure 8b). The lack of round shapes in this treatment corroborates the previous subjective observation from interactive evolution that CPPNs can struggle to evolve and exploit round gradients when they are not provided as inputs.

To test if seeding CPPNs with spherical gradients makes it easier to match this rounded target, we added distance to the center as an input. The CPPNs in the previous treatment could have evolved to calculate this same information, but that may have been difficult. Surprisingly, this information significantly lowered performance to 90.0% ($\pm$ 0.002 SE, $p = 0.013$, Mann-Whitney test, Figure 7). However, the evolved objects all have smooth, round forms (Figure 8c-d), confirming that providing different seed gradients can bias the types of evolved objects. While this might be expected in early generations, it is interesting that the gradients provided have noticeable effects after a thousand generations. This result is in line with a previous paper that found that the information input into CPPNs can bias the resulting phenotypes [Clune et al., 2009]. We include this input in the remaining treatments in this paper because it facilitates round surfaces, even though it hurt performance in this experiment.

Because interactive evolution features smaller population sizes, it is worthwhile to study how this difference affects the search for 3D objects. Additionally, since NEAT complexifies genomes over evolutionary time, having more generations may improve the search by accessing genomes with more hidden nodes. We investigate these issues by decreasing the population size from 150 to 15 and increasing the number of generations tenfold to 10^4, which keeps the number of evaluated objects the same. This change significantly improves performance to 91.8% ($\pm$ 0.003 SE, $p < 0.001$, Mann-Whitney test, Figure 7), suggesting that the small population sizes in interactive evolution do not hurt, and may actually benefit, morphological evolution with NEAT-based encodings. The evolved objects tend to have more space carved out between the spheres (Figure 8e-f).

A fundamental evolutionary parameter that can greatly affect evolvability is the mutation rate.We varied the major sources of mutation in NEAT by altering the rate at which genomic links are added, removed, and mutated, as well as the rate at which genomic nodes are added. Increasing the node addition rate significantly boosted performance ($p < 0.001$, Mann-Whitney test, Figure 7) to 91.5% ($\pm$ 0.003 SE). Changing the other mutation rate parameters did not improve performance (data not shown).

Because a smaller population with more generations was beneficial, and because a higher mutation rate was beneficial, we tested whether both changes together would outperform either alone. The combination did improve performance to 92.0% (Figure 7), but the difference was not significant ($p > 0.05$, Mann-Whitney test). We also found that the expanded genome function set (described previously) improved performed to 93.0%, which was significant ($p = 0.022$, Mann-Whitney test). As before, the objects in this treatment seemed to combine rounded surfaces with sharper edges: while most were smooth (e.g. Figure 8g-h), a few had rough patches on their surface, including Figure 8i. Adding recurrent genomic connections to this treatment did not significantly affect performance (93.3%, $p > 0.05$).

Overall, the target-based evolution experiments reveal that evolving CPPNs can roughy match a target object. While a high percentage of voxels were matched, the degree to which the evolved objects qualitatively resemble the target is subjective and debatable. The most important contribution of these experiments is to better understand the way in which target-based evolution is biased by different parameters. These results are preliminary, however, until more tests can be conducted with additional targets.

It is also interesting that many of the evolved objects look designed for a purpose. For example, many of the objects in Figure 8 seem like functional and aesthetically attractive objects carved on a lathe, such as legs from tables and chairs or posts from banisters and railings. One reason this is surprising is because it could have been the case that the greedy nature of target-based evolution would have gained improvements by iteratively adding small patches of voxels that match a subset of the overall space. Such a patchwork solution would not look as regular and smooth as the objects that actually evolved, suggesting that CPPNs are biased away from such a piecemeal strategy. Previous work has shown that CPPNs have difficulty making exceptions to regular patterns when evolving neural networks [Clune et al., 2011], which could explain why the target object in this study was not matched one patch at a time. Such a bias toward regularity may simultaneously explain the smoothness of the evolved objects and why matching the final few percent of voxels is so difficult.

Artists and engineers may actually benefit from the fact that the evolved objects share some properties of the target, but are different in interesting ways. This means that a designer can provide a seed object as a target, and a series of objects can automatically be generated that are aesthetically interesting variations on that seed concept (Figure 8).

3.3 Transferring Objects to the Physical World

Advances in 3D printing technologies make it possible to transfer evolved objects into the physical world, which may help artists and engineers benefit from this technology. To test whether CPPN objects maintained their appearance and structural integrity in reality we printed them on a Connex500 3D printer. The objects look similar to their simulated counterparts and are structurally sound (Figure 1). One difference is that non-contiguous pieces (e.g. the top of Figure6, left) are not held in place in the physical world without additional scaffolding. By printing in a semi-transparent material, we also discovered that none of the objects have visible hollow areas embedded within them, although CPPNs can create such negative spaces. While the gap between simulated and physical objects was not expected to be large for static objects, it is helpful to have verified the fidelity of the transfer.

4 Conclusions and Future Work

This paper introduces an algorithm for evolving 3D objects with the CPPN generative encoding, which is a computationally efficient abstraction of biological development. We conducted both interactive and target-based evolution to explore the ability of CPPNs to create complex objects, especially those that resemble natural and engineered designs.

A small, preliminary exploration of the design space of 3D CPPN objects unearthed a diversity of objects that evoke natural and engineered forms. Many of the objects featured regularities such as symmetry and repetition, with and without variation. Such properties are important for engineering and evolvability [Lipson, 2007, Clune et al., 2011], and suggest that CPPNs are a promising encoding for evolving useful and aesthetically pleasing objects. To extend this research we are creating a website like picbreeder.org [Secretan et al., 2011] where users can collaboratively evolve 3D objects online, which will provide a much larger exploration of the potential of this technology. It will also overcome the need for any individual to perform all of the evaluations in a lineage and thus allow more complex objects to evolve.

Experiments with target-based evolution on one target revealed how the inputs and parameters of CPPNs can influence the types of objects they evolve. The evolved objects roughly resemble the target, but do not match it precisely. While the evolved objects share some properties of the target, they also differ from it in interesting ways. This property could help artists and engineers by providing 3D designs that are variations on a seed concept. All of these conclusions are tentative, however, since experiments were only conducted with one target. Future work is necessary to determine whether these observations generalize.

While there are many useful applications for evolving static, single-material 3D objects, this technology is also a stepping stone to evolving objects that can move and that have multiple materials. In future work we will evolve such soft-bodied robots in simulation and transfer them to the physical world. Doing so will enable us to harness the power of evolution and developmental biology to begin to create synthetic creatures that have some of the exciting properties of their natural counterparts.

Acknowledgments

Thanks to Jon Hiller, an NSF Postdoctoral Research Fellowship in Biology to JC (DBI-1003220), NSF CDI Grant ECCS 0941561, and NSF Creative-IT Grant IIS 0757478.

References

[Auerbach and Bongard, 2010a] Auerbach, J. and Bongard, J. (2010a). Dynamic resolution in the co-evolution of morphology and control. In *Proceedings of Artificial Life XII*.

[Auerbach and Bongard, 2010b] Auerbach, J. and Bongard, J. (2010b). Evolving CPPNs to grow three-dimensional physical structures. In *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, pages 627–634. ACM.

[Bentley, 1996] Bentley, P. J. (1996). *Generic Evolutionary Design of Solid Objects using a Genetic Algorithm*. PhD thesis, University of Huddersfield.

[Bongard and Pfeifer, 2001] Bongard, J. and Pfeifer, R. (2001). Repeated structure and dissociation of genotypic and phenotypic complexity in artificial ontogeny. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 829–836.

[Clune et al., 2009] Clune, J., Ofria, C., and Pennock, R. (2009). The sensitivity of HyperNEAT to different geometric representations of a problem. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 675–682.

[Clune et al., 2011] Clune, J., Stanley, K., Pennock, R., and Ofria, C. (2011). On the performance of indirect encoding across the continuum of regularity. *IEEE Transactions on Evolutionary Computation*, To appear.

[Eggenberger, 1997] Eggenberger, P. (1997). Evolving morphologies of simulated 3D organisms based on differential gene expression. In *Fourth European Conference on Artificial Life*, pages 205–213. The MIT Press.

[Hiller and Lipson, 2010] Hiller, J. and Lipson, H. (2010). Morphological evolution of freeform robots. In *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, pages 151–152. ACM.

[Hornby et al., 2003] Hornby, G., Lipson, H., and Pollack, J. (2003). Generative representations for the automated design of modular physical robots. *IEEE Transactions on Robotics and Automation*, 19(4):703–719.

[Lehman and Stanley, 2008] Lehman, J. and Stanley, K. (2008). Exploiting open-endedness to solve problems through the search for novelty. *Artificial Life*, 11:329.

[Lindenmayer, 1968] Lindenmayer, A. (1968). Mathematical models for cellular interactions in development I. Filaments with one-sided inputs. *Journal of Theoretical Biology*, 18(3):280–299.

[Lipson, 2007] Lipson, H. (2007). Principles of modularity, regularity, and hierarchy for scalable systems. *Journal of Biological Physics and Chemistry*, 7(4):125.

[Lorensen and Cline, 1987] Lorensen, W. and Cline, H. (1987). Marching cubes: A high resolution 3D surface construction algorithm. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 163–169.

[Miller, 2004] Miller, J. (2004). Evolving a self-repairing, self-regulating, French flag organism. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 129–139. Springer.

[Secretan et al., 2011] Secretan, J., Beato, N., D'Ambrosio, D., Rodriguez, A., Campbell, A., Folsom-Kovarik, J., and Stanley, K. (2011). Picbreeder: A Case Study in Collaborative Evolutionary Exploration of Design Space. *Evolutionary Computation*, To appear.

[Sims, 1994] Sims, K. (1994). Evolving 3D morphology and behavior by competition. *Artificial Life*, 1(4):353–372.

[Stanley, 2007] Stanley, K. (2007). Compositional pattern producing networks: A novel abstraction of development. *Genetic Programming and Evolvable Machines*, 8(2):131–162.

[Stanley and Miikkulainen, 2002] Stanley, K. and Miikkulainen, R. (2002). Evolving neural networks through augmenting topologies. *Evolutionary Computation*, 10(2):99–127.

[Stanley and Miikkulainen, 2003] Stanley, K. and Miikkulainen, R. (2003). A taxonomy for artificial embryogeny. *Artificial Life*, 9(2):93–130.

[Wolpert and Tickle, 2010] Wolpert, L. and Tickle, C. (2010). *Principles of Development*. Oxford University Press, 4th edition.

About the authors

Jeff Clune is at Cornell University funded by an NSF Postdoctoral Research Fellowship. He studies generative encodings, which enhance evolutionary algorithms by augmenting them with concepts from developmental biology. Such concepts enable the assembly of complex forms from compact genomes, including the evolution of large-scale, structurally organized neural networks. Jeff also investigates open questions in evolutionary biology, and has published work on the evolution of altruism, phenotypic plasticity, and evolvability. Jeff was the co-chair of the Generative and Developmental Systems track at GECCO (2010 & 2011) and won the best paper award in that track in 2009. He has a bachelor's degree in philosophy from the University of Michigan and a Ph.D. in computer science and a master's degree in philosophy from Michigan State University. His research has appeared in news outlets such as Science, MSNBC, the New Scientist, MIT's Technology Review, the Daily Telegraph, Slashdot, and U.S. News & World Report.

Homepage: http://jeffclune.com
Email: jeffclune@cornell.edu

Hod Lipson joined the departments of Mechanical & Aerospace Engineering and the faculty of Computing & Information Science of Cornell University in Ithaca, NY. He is also a member of the Computer Science and Computational Biology graduate fields at Cornell in 2011. Prior to this appointment, he was a postdoctoral researcher at Brandeis University's Computer Science Department and a Lecturer at MIT's Mechanical Engineering Department. He received his PhD in 1998 from the Technion - Israel Institute of Technology. Before joining academia, he spent several years as a research engineer in the mechanical, electronic and software industries.

Homepage: http://web.mae.cornell.edu/lipson/
Email: hod.lipson@cornell.edu

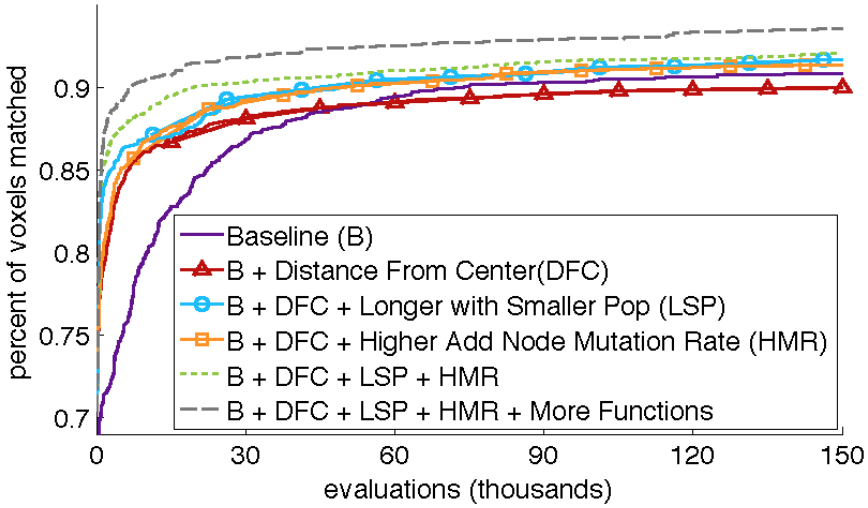


Fig. 7: Means of the best-performing individuals for target-based evolution. See text for variance.

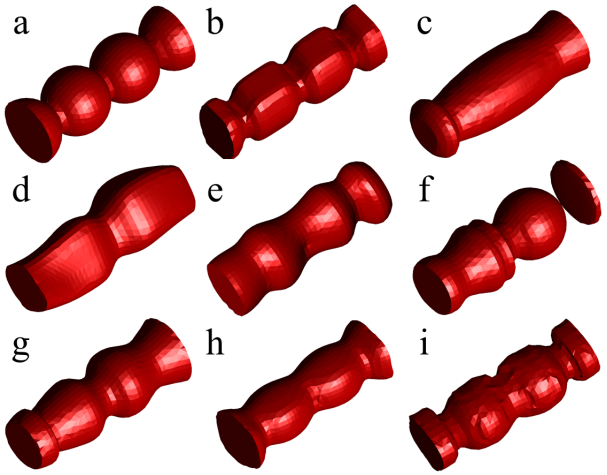


Fig. 8: Target-based evolution objects.

Academic Search Portal

Odysci is a new web portal for search and rank of scientific articles published in journals and conferences worldwide. Our objective is to provide the best academic search engine available, along with a productive technical social-network environment.

Scientific information overload is a problem that afflicts scientists, developers, engineers and students. They are flooded with hundreds of publications all the time and it is difficult and time consuming to find the most relevant publications that would really help them do their work. Moreover, publications are spread over multiple publishers' sites and the user needs to comb through them in order to find the most relevant ones.

Odysci addresses these problems by providing better ranking algorithms, more flexible and powerful search options to users, and a technical collaboration environment for them to interact with peers. In addition, Odysci aggregates publications from multiple publishers and makes them available in one place. The end result is more productivity, less time wasted reading useless publications, and better research and development for users and their institutions.

Odysci offers new interesting ways to search and to allow the user to control and narrow the results. For instance, a user can search for papers written by an author while he/she worked at a given place. To find papers written by Leslie Lamport while at SRI, one would search for [author:lamport author@SRI]. Alternatively, a user can search for papers on a given topic written by authors of a given institution. For example, the search [Genetic Algorithms author@"University of Illinois"] will return papers on genetic algorithms written by Univ. of Illinois authors. One can also search for papers which received awards. To search for the Best Papers in the GECCO conferences, one would use [venue:GECCO bp:true].

Besides search, the portal also offers: (a) Information about upcoming conferences and deadlines, (b) Lists of best papers (and the ability to search for them), and (c) Comments by experts in different fields (our Online Editors). Another interesting feature is the ability for a user to automatically find his/her own papers in our system and add them to their profile page. We also have Alerts which allow a user/author to receive a notification whenever a new paper is published that cites a paper by him/her.

Collaboration facilities are also a main part of the portal. These are being driven by the goal of productive technical networking. More than a place to list "connections", in Odysci users can comment on papers, form groups for sharing papers, and collaborate effectively with peers. We'll be releasing new features for that in the near future.

Odysci's main research focuses on developing novel algorithms for data de-duplication and ranking algorithms, since we consider the quality of our data and ranking important aspects for the user. We currently list around 2 million documents, including data from ACM, IEEE Computer Society, several other public databases and data gathered using crawlers. New conferences and journals are added on a weekly basis.

We would like to invite the SIGEVO community to try out www.odysci.com for search of research papers, and we welcome your feedback.

Webpage	http://www.odysci.com
Blog	http://blog.odysci.com
Twitter	@odysci

Beyond Biology

Designing a New Mechanism for Self-Replication and Evolution at the Nanoscale

Rebecca Schulman, Chemical and Biomolecular Engineering, Johns Hopkins University, rschulm3@jhu.edu

As biology demonstrates, evolutionary algorithms are an extraordinarily powerful way to design complex nanoscale systems. While we can harness the biological apparatus for replicating and selecting DNA sequences to evolve enzymes and to some extent, organisms, we would like to build replication machinery that would allow us to evolve designs for a much wider variety of materials and systems. Here we describe work that uses techniques from the new field of structural DNA nanotechnology to modularly design nanoscale components that together can be assembled into a system for self-replicating a new form of chemical information or genome, and thus for evolving a new type of chemical sequence.

1 Introduction

A major current scientific challenge is to learn how to design materials with nanoscale features and to exploit the unique properties of materials available at this scale. Some of the benefits of nanoscale engineering are widely familiar: the increasing density with which we can organize transistors on a chip is largely responsible for the increasing speed of our computers. But there are many other cases where nanoscale features change the properties of materials in ways that we can exploit: for example, the optical and electronic properties of nanometer-scale crystals and wires can be dependent on their dimensions [27, 23].

Further, we expect that much of the engineering possibility at the nanoscale remain to be discovered. Perhaps the most dramatic demonstration of the benefits that could be gain by having molecular-scale control over matter is biology. Inside cells, the production, transformation, and functions of individual molecules are precisely controlled.

These features are essential to the capacity of biology for self-replication, self-healing and metamorphosis. By having similar control over molecular synthesis and nanoscale geometry in synthetic systems, it should be possible to achieve these features as well as many others in synthetic materials.

Biology's sophisticated architecture is the product of the Darwinian evolution of a genomic sequence, an organism's program for growth and function. Evolution is therefore an extraordinarily powerful design strategy for nanoscale materials and devices. And evolutionary algorithms for molecular design such as SELEX for evolving RNA molecules with catalytic function [22, 16] and directed evolution for evolving functional proteins [2] have been more successful than comparable rational design strategies.

But there is currently an important limitation on our ability to solve molecular design problems using Darwinian evolution: we can only replicate, and thus evolve, DNA or RNA sequences. This replication can take place in cells or in the test tube, but in either case the form of the information replicated, a sequence of nucleic acids, is the same. While changing the representation of the information being evolved in an *in silico* process is straightforward, translating the representation of chemical information is extremely challenging.

Biology has figured out some mechanisms for accomplishing this representation change: the "central dogma" of molecular biology is that DNA can be transcribed into an RNA sequence and then translated into an amino acid sequence, which folds into a protein; a set of proteins can then together synthesize other molecules.

But there is no obvious way to translate DNA sequence information into instructions for autonomously constructing many structures we might be interested in, such as silicon-based circuitry.

The chemical translation problem is not theoretically difficult, but difficult in practice: even trying to augment the genetic code to include one new kind of amino acid has been a major technical challenge [41]. In the past decade, there have been initial attempts to build a more general *in vitro* apparatus for translating DNA sequences into synthesis recipes [19, 20] that might allow us to evolve a much wider array of products.

And at the same time, new technologies for designing libraries of possible sequences (*i.e.* controlling the mutation operation) have improved the process of evolutionary design of proteins [21]. But because of the challenges inherent to chemical translation, we might ask more generally whether evolving a sequence of 4 bases is the most efficient way to solve all molecular design problems. In software, both the representation of the information being evolved (as well as how this information is used to produce the function being evolved) and the mechanism of mutation are important for efficiently solving design problems using genetic and evolutionary algorithms [25, 31]. If instead of evolving DNA sequences that are replicated in cells or by enzymes extracted from cells, we could design systems for molecular replication and mutation the way we can design evolutionary algorithms, we might be able to solve a much wider variety of chemical design problems and build new nanoscale materials with evolution.

We are still far from being able to design arbitrary molecular machinery capable of processes as complex as self-repli-cation *de novo*, and we know only a little about which aspects of replication and evolution in molecular systems are the major determinants of their efficiency [15, 7]. But important progress is being made: we are learning how to design modular molecular components and how to combine these components into functional molecular machines. And from these modular parts we can begin to build devices for chemical self-replication.

Here we give an account of the development of components for a new system for molecular information replication and of how evolution could proceed in such a system. We first describe how we can design molecular components made from synthetic DNA, (short DNA sequences made chemically in the laboratory rather than by enzymes within cells). The component DNA sequences of these structures, arbitrary sequences of A's, T's, G's and C's, can be designed and optimized on the computer. We then describe how we can use synthetic DNA components, called DNA tiles, in a self-assembly process. This self-assembly process is analogous in some sense to solving a jigsaw puzzle and performs computation during assembly.

That is, for any given computation, we can design a set of DNA tiles that executes that computation via self-assembly. We describe how to design a set of DNA tiles that copies a sequence of information during assembly. The assembly process propagates the sequence; and when mechanical forces fracture an assembly, new sites on the fragmented assembly become available where the sequence can be propagated, increasing the rate of sequence propagation. Cycles of sequence propagation (assembly) and fragmentation exponentially replicate the sequences. We describe how to implement this process experimentally and how evolution would occur in this system.

The processes we can design using synthetic DNA continue to increase in both complex and variety. There are now several proposals for building systems for sequence replication (and thus Darwinian evolution) from synthetic DNA components [47, 24]. As the set of systems available for molecular sequence replication and evolution grows, we will have new opportunities to both learn about evolution of physical systems and to design efficient algorithms for evolution and selection in these new systems.

2 DNA Tiles and Algorithmic Self-Assembly

DNA is most familiar as the material in which our genome is stored. What underlies DNA's capacity for storing and replicating information is its propensity for Watson-Crick complementary DNA bases to hybridize and form double-helical DNA. Recently, DNA's sequence specific binding capacity has become an engineering tool: it is possible to design a sequence and its complement and to know that these two sequences will bind but that they will not interact with other DNA molecules in the environment.

In 1982, Nadrian Seeman described how synthetic DNA might be used for nanoscale-construction. Seeman imagined using DNA molecules as programmable molecular tinker-toys that would self-assemble into designed structures because the complementary regions of the designed sequences would hybridize while other sequences would not react. He

described how we might make branched DNA structures, and thus program the formation of 2- and 3-dimensional assemblies [37]. As Seeman described it, nanotechnology could happen the way it does in biology: autonomously – we would simply design the sequences, synthesize them, and put them together in a test tube and wait.

Designing a system of DNA molecules has turned out to be more tractable than the design of other types of complex molecular systems: the rate of DNA hybridization and the stability of base-paired DNA are generally predictable in polynomial time [48], and the double-helical structure of hybridized DNA is well-characterized and largely independent of the particular based-paired sequence [8]. These properties have enabled the design of extended 2- and 3-dimensional structures [45, 29, 4], programmed molecular machines [46, 5] and active structures [43, 46, 14] via the design of a set of DNA molecules and their relative abundances.

A DNA "tile" (Figure 1a) is a primitive for nanoscale construction [17, 45]. A DNA tile consists of a double-stranded "core" and 4 single-stranded "sticky ends." Tiles attach to each other via sticky end hybridization and can form extended two-dimensional lattices [45]. In principle, the arrangement of tile types within the lattices that form can be designed by designing appropriate DNA tile sticky end logic, a process akin conceptually to designing the pieces of a jigsaw puzzle and their interlocking nubs (Figure 1b). Given a desired sticky end logic, we can design and synthesize a set of DNA sequences that assemble into tiles that implement this logic (e.g. [45, 30, 3]).

Complex patterns can be constructed from DNA tiles efficiently by a technique known as algorithmic self-assembly [42]. The basic premise of algorithmic self-assembly is that an object is constructed *algorithmically*, that is by executing a program.

Algorithmic self-assembly has its roots in the tiling problem, the question of whether a given set of shapes can tile the plane, which is undecidable [39, 40, 6]. Using observations derived from the hardness of plane tiling, Winfree described a set of tiles and a constructive method for their assembly that executes a computer program [42, 43].

In Winfree's construction, growth of a tile crystal begins from a seed tile or structure whose sticky ends encode the initial state of a computation. Under physical conditions where tiles can attach to the seed only by two sticky ends simultaneously (*i.e.* just cooler than the melting temperature of the crystal), the growth of a DNA tile crystal, or lattice, can in principle simulate the execution a 1-dimensional blocked cellular automaton,

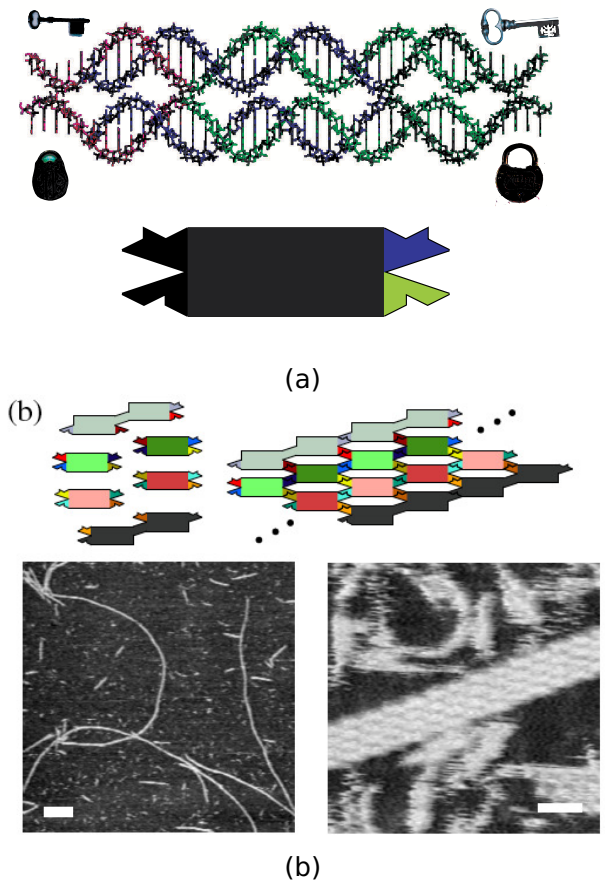


Fig. 1: **DNA tiles and tile nanostructures. (a)** A DNA tile is a nanoscale construction primitive. Top, a molecular model of a tile that contains short DNA molecules. Each strand is depicted in a different color. Bottom, a schematic shows the effective shape of a tile along with the logic of its sticky ends. Tile "cores" (e.g. the green portion of the schematic tile shown here) are double-stranded; the assembled core maximizes the number of Watson-Crick complementary base pairs between the component strands and is therefore a favorable configuration. Single-stranded "sticky ends" (the colored claws in the schematic) function as locks and keys: they specifically hybridize (*i.e.* bind) to complementary sticky end sequences on other tiles. **(b)** Tiles designed to form a 4-tile-wide ribbon, and atomic force micrographs of the ribbons, which assembled as designed. Scale bars are 500 nm (left) and 25 nm (right) (image from [33], copyright Proceedings of the National Academy of Sciences, USA).

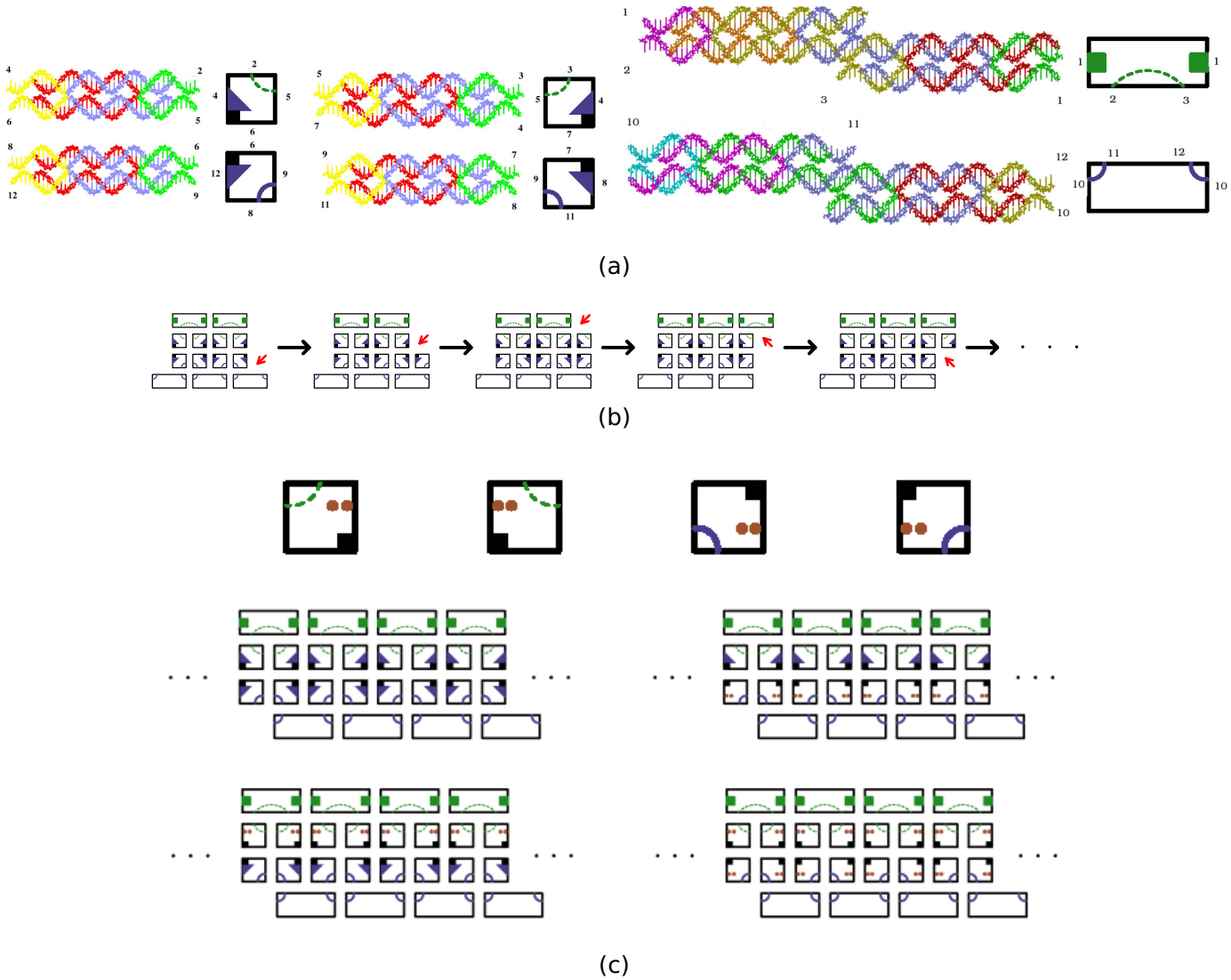


Fig. 2: **Zig-zag tiles. (a)** The basic zig-zag tile set. Each square and rectangle shown is a logical representation of the molecule shown to its left. **(b)** Zig-zag growth. At each growth step, a new tile may be added at the location designated by the small arrow. Two alternating tile types in each row enforce the placement of the double tiles on the top and bottom, ensuring that growth occurs in a zig-zag pattern. Although only growth on the right end of the molecule is shown here, growth occurs simultaneously on both ends of the assembly. **(c)** The tile set shown in Figure 2b forms only one type of assembly. A tile set consisting of the tiles in (b) and the four tiles shown here allows four types of assemblies to be formed. The vertical column of each type contains a crystal's 2-bit binary sequence.

and therefore perform universal computation. Intuitively, the two sticky ends a tile must match in order to attach to a growing crystal are "input" states to a cellular automaton and the remaining two sticky ends are the "output" of a single computing step. Since growth can continue indefinitely, arbitrarily long computations can be performed. Notably, the entire history of a computation is stored in the arrangement of tile types within the assembled crystal. In many cases this arrangement may form a useful structure that is difficult to assemble by other means [13].

The assembly of the designed structure requires that at each step of assembly a valid tile, *i.e.* a tile that matches two sticky end binding sites simultaneously, be added to the crystal. However, in initial experiments [30] as many as 1%-10% of attachments were *errors*, or not valid—only one of the "input" edges of the tile matched the available inputs on the growing crystal. The wrong logical operation was being performed at those sites.

As would be expected of a computation in which 1–10% of the primitive operations were computed incorrectly, the patterns that formed were generally not the designed patterns.

The error rate can be reduced by logically redesigning the tiles to perform the same computation during assembly, but more robustly. "Proofreading" tile sets [44, 12, 28, 38] transform a tile set by replacing each individual tile with a $k \times k$ block of tiles, exponentially reducing seeded growth errors with respect to the size of the block. Along with the improvement of the structure where computation begins, the "seed" [4] and new techniques to prevent growth that does not begin from a seed, proofreading techniques allowed assembly to proceed much more accurately, *i.e.* with error rates as low as 1 in 1000 tiles. Structures such as Sierpinski gaskets [30, 18] and "binary counters" [3, 4] have been assembled using these techniques.

3 Self-Replicating DNA Crystals

In 1966, Graham Cairns-Smith proposed a simple mechanism by which polytypic clay crystals (clays that can take on one of many crystal structures) could replicate information in the absence of biological enzymes [9, 10]. Some polytypic clay crystals contain discrete layers, each of which contain molecules of a particular identity or orientation. A cross-section of such a crystal can contain an information-bearing sequence. Cairns-Smith proposed that crystal growth could extend the layers, copying the sequence (the crystal's genotype). Occasionally, physical forces could break a crystal apart. Because crystals replicate their genotype many times during growth, splitting of a crystal can yield multiple pieces, each containing at least one copy of the information-bearing sequence. Cycles of growth and fragmentation could therefore allow a sequence to be exponentially amplified.

We have adapted Cairns-Smith's ideas about spontaneous information replication in crystals to *design* a system for self-replication using DNA tiles as crystal monomers [32]. A simple set of DNA tiles can form *zig-zag crystals* that can propagate information during growth [33, 4]. The tiles shown in Figure 2a form the zig-zag crystal shown in Figure 2b. Matching rules determine which tile fits where. Under conditions where each tile addition must form two or more sticky end bonds (Figure 2a), growth is constrained to occur in a zig-zag pattern. It is easy to confirm that under such conditions, there is always a unique tile that may be added on each end of the ribbon.

Zig-zag crystals are designed so that under conditions where a tile must attach to a crystal by at least two bonds, growth produces one new row at a time (*i.e.* one copy of a sequence) and continued growth repeatedly copies a sequence. The requirement that a tile must attach by two bonds means that a tile being added must match both its vertical neighbor (another tile that is part of the new column being assembled), and its horizontal neighbor (in a previously assembled row).

Several tiles might match the label on the vertical neighbor, but because tiles must make two correct bonds in order to join the assembly, only a tile that also matches the label on the horizontal neighbor can be added. The tile being added in the new column must therefore correspond to the one in the previous column. As a result, information is inherited through templated growth. The set of tiles formed by adding the tiles in Figure 2c to those shown in Figure 2b can propagate one of four strings. Additional tiles may be added to the set of tiles in Figures 2b and 2c to create a tile set that can propagate arbitrary binary sequences.

The growth of a zig-zag DNA crystal increases the number of copies of the original information present in the ribbon but does not change the rate at which new copies of the sequence are produced. The rate of copying can be sped up by breaking the crystals. With each new crystal that is created by breakage, two new "growth fronts" become available where tiles can attach and information can be copied. Repeated cycles of growth and breakage exponentially amplify an initial piece of information. Occasionally, a tile matching only one bond rather than two will join the assembly, resulting in occasional copying errors, which are also inherited. If errors happen during copying, which they will under almost any achievable condition [43], and crystals with particular sequences grow faster than others, then evolution can occur.

4 Selection in Physical Systems

In general, in an evolutionary or genetic algorithm a population is generated and afterwards some portion of the individuals is selected on the basis of their fitness. This subpopulation is used to create a population for the next generation via mutation and/or recombination. In a physical system the process of filtering and creation of a population for the next generation must be physically realizable, which is currently a strong limitation. Many types of fitness that we would like to select for, such as determining whether a molecule has a particular catalytic function, are difficult to measure in practice, and the partitioning of molecules or species based on their fitness is also challenging experimentally. While molecular "tricks" can sometimes permit autonomous selection of fit individuals [16], there are no general methods for evolution and selection based on function.

If we want to build novel systems for the evolution and selection of molecules or other physical entities, therefore, we will also need to develop ways to make this selection process easier. In biology, the desired function is the capacity to reproduce quickly with respect to other individuals in a population. Could we tie function to this capacity in artificial systems? To answer this question we must first understand why some species might replicate more quickly than others in a given self-replication process. Below we examine why some DNA tile sequences might be replicated more quickly than others, and consider as a result what selection processes for "fit" DNA tile sequences might be feasible.

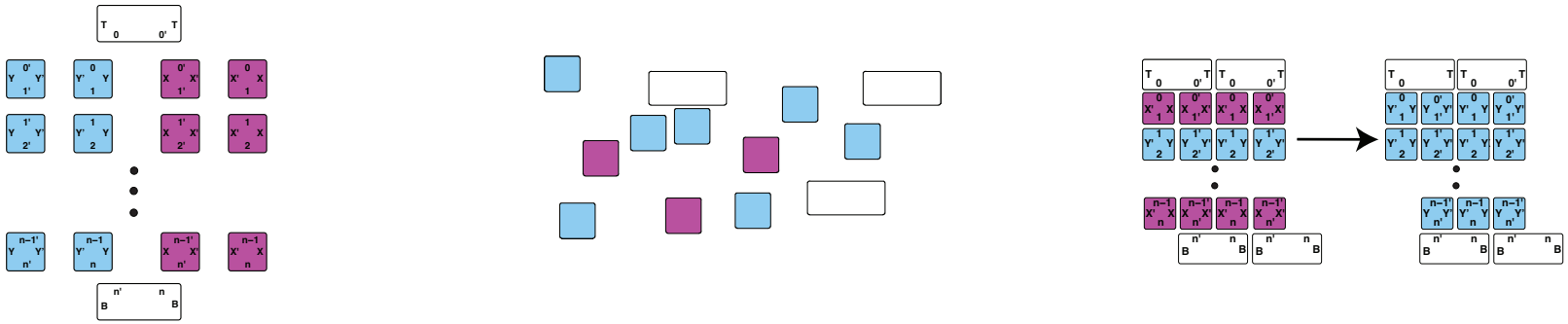
5 Evolution of DNA Crystals for Fast Growth: The Royal Road

A selection process in a physical self-replicating system involves both an environment (a set of resources for growth, their chemistry and the ambient physical conditions) and an initial population of organisms (sequences).

In a DNA tile replication process, the environment includes a set of DNA tiles. The set of DNA tiles determines the set of sequences which may be copied and the "chemistry" of the system, *i.e.*, the rules by which tiles bind to each other. A particular arrangement of DNA tiles is the information that is propagated in these experiments, the genotype; it is the organism being evolved. The phenotype of a sequence is its replication rate in the environment. In this section we first describe a tile set that allows many kinds of sequences to grow and then how selection pressure results from physical conditions in which the concentration of tile types differ.

A DNA crystal grows by adding tiles. Tiles come in contact with the crystal as the crystals and tiles diffuse randomly in the aqueous solution where growth occurs. Generally this growth takes place in a well-mixed reaction vessel, *i.e.* the density of crystals and monomers is on average uniform across the reaction container. In this case, the higher the concentration (*i.e.* density in solution) of a tile type that the vessel contains, the more quickly a tile of that type will contact a crystal where it can be legally added. Therefore, one simple selection pressure results from a difference in concentration between tile types used to copy sequence information: assemblies with sequences containing tile types present at high concentrations will grow faster than assemblies with sequences containing tile types present at very low concentrations.

A tile set in which one of two bits can be propagated at each of n sequence positions is shown in Figure 3a. Let X_i and Y_i be the two tile types that can be propagated at sequence position i. If Y_i's concentration is higher than X_i's concentration in solution, as suggested by the illustration in Figure 3b, the resulting fitness landscape resembles the simplest case of a well-studied problem in genetic algorithms, the "royal road" [26].



(a) (b) (c)

Fig. 3: **Royal Road Selection. (a)** For a DNA tile ribbon containing sequences of width n, the Royal Road tile set contains $4n+2$ tile types. Matching sticky ends have identical labels. Each position of the sequence contains either a cyan time (from the left group of tile types) or magenta tile (from the right group of tile types). **(b)** An environment where cyan tile types are present in higher concentrations than magenta tile types. **(c)** Selection in the environment in (b) favors sequences containing cyan tiles, since cyan tiles will be added to crystals faster than magenta tiles.

The growth rate of a crystal is proportional to the number of Y's in the sequence being propagated. For each position i, as long as the concentration of Y_i is higher than the concentration of X_i, sequences containing only Y_i tiles will be fitter and quickly dominate the population during a selection process (Figure 3c).

6 Evolution of DNA Crystal Algorithms

The previous section demonstrates how the scarcity of tile resources can lead to selection. But it does not address the question of how this selection could be used to evolve or improve a useful function of a molecular system: in the Royal Road process as we described it, the evolution process is a straightforward optimization problem with a known solution; no function or algorithm is being discovered.

If in contrast the sequence being evolved were a template or directive for an algorithm or device, the evolution process could select for functional behavior. To achieve such functional evolution it is necessary to define the language, or representation, of the information being evolved and the process of translating this information into a particular function.

How could we make the information being replicated functional? DNA crystals, as described in Section 2, can compute during growth as well as copy information. We can use this capacity to build sequences that function as programs. In fact, any program, no matter how complex, can be selected for [34, 35]. Thus, DNA crystals can in principle evolve powerful and complex functions. We review the mechanisms by which such selections can occur here.

As we described in Section 2, DNA crystals can perform a computation via the attachment of tiles to a growing crystal. A tile that can favorably attach at a growth site must match two labels at the growth site, the "input" labels. This simultaneous matching of two input labels is an elementary computing step. The other two labels on the attaching tile, the output labels, determine which tiles can fit in subsequent growth sites, so that information about the state of the computation is transmitted during growth.

Collectively, these tile attachments can simulate a Turing machine [42] where the initial state of the computation is determined by the structure of the seed where tile assembly begins.

It is also possible to build a set of tiles that function as a universal Turing machine – the structure of the initial inputs on the seed determine which computation occurs during growth [34, 11].

In principle, such a tile set can be expanded to make a tile set that builds ribbons that have two parts – a segment that runs a program on the universal Turing machine, and a segment that makes copies of this program [34]. Such a zig-zag ribbon tile set would be a sort of "universal alphabet," with which we could build crystals that simultaneously store a program (its genome), and run it. During replication, the program's source code would be inherited, and in an evolution process that used this tile set, crystals containing particularly fit programs would be selected for.

How could a program make a crystal fit? First, the execution of crystal programs can build algorithmic patterns with potentially interesting features [13] that we could test via an artificial selection process. If we attached small devices to individual tiles, a program that built a binary counter might produce a pattern suitable for templating a demultiplexer circuit, for example [13]; other patterns might arrange molecules or nanoparticles into a combinatorial ensemble of interesting geometries. These assembled patterns could have optical, electronic or chemical functionality that could be selected for (given an available selection protocol), just as chemical functionality is currently selected for in SELEX or directed evolution experiments.

A tile program could also be a control system for adaptively sensing and responding to the environment. As we described in Section 5, the most basic reason for fitness is rapid growth, and crystals which use tile types that are abundant in the environment grow rapidly: a tile t is added at an average rate $\frac{k_f}{[t]}$ where k_f is a tile-independent rate constant, and $[t]$ is the concentration (density in solution) of tile t. More generally, if we disregard the frequency of fragmentation, the fitness of a crystal is proportional to the time it takes to grow a crystal layer [35], which is the sum of the times it takes to add each new tile in the layer. Thus, each tile addition makes a contribution to a crystal's fitness.

A fit crystal control program would be a program that could learn what tile types are abundant and then adopt the growth process to use as many of the most abundant tile types as possible.

One way for a tile program to continually use abundant (as opposed to rare) tile types would be for the growing crystal executing the program to read information about whether tiles are abundant or rare at specified growth sites where multiple tile types could attach. The program could then use this input to determine which other tile types are abundant and thus should be used for computation. Such a program could be viewed as a sort of "metabolism" for crystals that figures out what nutrients are available and uses the available nutrients for energy and growth, in a process akin to metabolic sensing and response by biological cells. This kind of "crystal" control system sounds primitive, but in principle it could be arbitrarily complex: because crystals can simulate a Turing machine, they can assemble a program that senses and responds to any computable correlation between the abundances of tile types over time. If the correlations between tile type concentrations were very complex, then a very complex tile program to compute and take advantage of these correlations would evolve.

This tile set and evolutionary process (the changing concentrations of tile types over tile) could be a model system for studying evolution in non-biological molecular systems: we have a quantitative model of crystal behavior and the system as a whole and we have control over the concentrations of each tile type. In contrast, in biological systems we do not have control over many variables that are important to fitness, and the system dynamics are largely not understood: even the best-understood organisms produce hundreds of proteins whose functions are not known [1].

And while tile concentrations are not generally quantities that have immediate real-world interest, we could include modules in the growth environment that translate signals of other types into tile concentrations [36]. These translation systems would function as separate components, *i.e.* as molecular sensors that as output either produced or used up tiles, thus changing their concentrations. In a more sophisticated tile-based replication system, arrangements of tiles could themselves function as sensors and thus have function.

7 Conclusions

DNA tile crystal growth and scission is a novel synthetic mechanism for molecular sequence self-replication. In principle, evolution in tile crystal systems is as computationally rich as evolution in any system: if the mutation rate during crystal growth could be made arbitrarily low, then eventually any program, no matter how complex, can evolve if it is the most fit program for the environment.

It may thus be that for physical systems, the capacity to perform universal computation and tie this computation in some way to the environment may be sufficient for open-ended evolution in a self-replicating system. In practice the speed of evolution and selection is also vital: if an evolutionary optimization process took more time than the age of the universe to complete, it would be of no practical interest. Thus what is needed is a study of how to quickly and robustly evolve solutions to problems of interest.

The challenge of evolving these structures in the laboratory will teach us new things about how to encode evolutionary processes in physical, as opposed to purely computational systems. The DNA crystals described here replicate molecular information in one way. In the future we will broaden our library of mechanisms for self-replicating systems which will allow to grow closer to engineering evolutionary algorithms for a variety of molecular design problems. It will also allow us to examine the trade-offs in not only the implementation of an alphabet within a single self-replicating mechanism, but also the trade-offs inherent in the design of the mechanism itself.

References

[1] M. Arifuzzaman, M. Maeda, A. Itoh, K. Nishikata, C. Takita, R. Saito, T. Ara, K. Nakahigashi, H.-C. Huang, A. Hirai, K. Tsuzuki, S. Nakamura, M. Altaf-Ul-Amin, T. Oshima, T. Baba, N. Yamamoto, T. Kawamura, T. Ioka-Nakamichi, M. Kitagawa, M. Tomita, S. Kanaya, C. Wada, and H. Mori. Large-scale identification of protein-protein interaction of Escherichia coli K-12. *Genome Res.*, 16:686–691, 2006.

[2] F. H. Arnold. Design by directed evolution. *Accounts of Chemical Research*, 31:125–131, 1998.

[3] R. D. Barish, P. W. K. Rothemund, and E. Winfree. Two computational primitives for algorithmic self-assembly: Copying and counting. *Nano Lett.*, 5:2586–2592, 2005.

[4] R. D. Barish, R. Schulman, P. W. K. Rothemund, and E. Winfree. An information-bearing seed for nucleating algorithmic self-assembly. *P. Natl. Acad. Sci.*, 106(15):6054–6059, 2009.

[5] J. Bath and A. J. Turberfield. DNA nanomachines. *Nat. Nanotechnol.*, 2:275–284, 2007.

[6] R. Berger. The undecidability of the domino problem. *Memoirs of the AMS*, 66:1–72, 1966.

[7] C. K. Biebricher, M. Eigen, and R. Luce. Kinetic analysis of template-instructed and de novo RNA synthesis by Qβ replicase. *J. Mol. Biol.*, 148:391–410, 1981.

[8] V. A. Bloomfield, D. M. Crothers, and I. Tinoco. *Nucleic acids: structures, properties, and functions*. University Science Books, Mill Valley, Cal., 2000.

[9] A. G. Cairns-Smith. The origin of life and the nature of the primitive gene. *J. Theor. Biol.*, 10:53–88, 1966.

[10] A. G. Cairns-Smith. The chemistry of materials for artificial Darwinian systems. *Int. Rev. Phys. Chem.*, 7:209–250, 1988.

[11] H.-L. Chen, Q. Cheng, A. Goel, M.-D. Huang, and P. M. de Espanés. Invadable self-assembly: Combining robustness with efficiency. In *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 883–892, 2005.

[12] H.-L. Chen and A. Goel. Error free self-assembly using error prone tiles. In C. Ferretti, G. Mauri, and C. Zandron, editors, *DNA Computing 10*, volume LNCS 3384, pages 62–75, Berlin Heidelberg, 2005. Springer-Verlag.

[13] M. Cook, P. W. K. Rothemund, and E. Winfree. Self-assembled circuit patterns. In J. Chen and J. Reif, editors, *DNA Computing 9*, volume LNCS 2943, pages 91–107, Berlin Heidelberg, 2004. Springer-Verlag.

[14] R. M. Dirks and N. A. Pierce. Triggered amplification by hybridization chain reaction. *P. Natl. Acad. Sci.*, 101(43):15275–15278, 2004.

[15] M. Eigen, J. McCaskill, and P. Schuster. Molecular quasi-species. *J. Phys. Chem.*, 92:6881–6891, 1988.

[16] A. D. Ellington and J. W. Szostak. *In vitro* selection of RNA molecules that bind specific ligands. *Nature*, 346:817–821, 1990.

[17] T.-J. Fu and N. C. Seeman. DNA double-crossover molecules. *Biochemistry*, 32:3211–3220, 1993.

[18] K. Fujibayashi, R. Hariadi, S. H. Park, E. Winfree, and S. Murata. Toward reliable algorithmic self-assembly of DNA tiles: a fixed-width cellular automaton pattern. *Nano Letters*, 8:3554–3560, 2008.

[19] D. R. Halpin and P. B. Harbury. DNA display II. genetic manipulation of combinatorial chemistry libraries for small-molecule evolution. *PLOS Biol.*, 2:e174, 2004.

[20] Y. He and D. R. Liu. Autonomous multistep organic synthesis in a single isothermal solution mediated by a DNA walker. *Nat. Nanotechnol.*, 5:778–782, 2010.

[21] C. J ackel, P. Kast, and D. Hilvert. Protein design by directed evolution. *Annu. Rev. Biophys.*, 37:153–173, 2008.

[22] G. F. Joyce. Amplification, mutation and selection of catalytic RNA. *Gene*, 82:83–87, 1989.

[23] K. L. Kelly, E. Coronado, L. L. Zhao, and G. C. Schatz. The optical properties of metal nanoparticles: The influence of size, shape, and dielectric environment. *J. Phys. Chem. B*, 107:668–677, 2003.

[24] M. E. Leunissen, R. Dreyfus, R. Sha, T. Wang, N. C. Seeman, D. J. Pine, and P. M. Chaikin. Towards self-replicating materials of DNA-functionalized colloids. *Soft Matter*, 5:2422–2430, 2009.

[25] G. E. Liepinsa and M. D. Vose. Representational issues in genetic optimization. *J. Exp. Theor. Artif. In.*, 2:101–115, 1990.

[26] M. Mitchell, S. Forrest, and J. H. Holland. The royal road for genetic algorithms: Fitness landscapes and GA performance. In *Proceedings of the First European Conference on Artificial Life*, 1992.

[27] T. W. Odom, J.-L. Huang, P. Kim, and C. M. Lieber. Atomic structure and electronic properties of single-walled carbon nanotubes. *Nature*, 391:62–64, 1998.

[28] J. H. Reif, S. Sahu, and P. Yin. Compact error-resilient computational DNA tiling assemblies. In C. Ferretti, G. Mauri, and C. Zandron, editors, *DNA Computing 10*, volume LNCS 3384, pages 293–307, Berlin Heidelberg, 2005. Springer-Verlag.

[29] P. W. K. Rothemund. Folding DNA to create nanoscale shapes and patterns. *Nature*, 440:297–302, 2006.

[30] P. W. K. Rothemund, N. Papadakis, and E. Winfree. Algorithmic self-assembly of DNA Sierpinski triangles. *PLOS Biology*, 2:424–436, 2004.

[31] F. Rothlauf. *Representations for genetic and evolutionary algorithms*. Physica-Verlag, Heidelberg New York, 2002.

[32] R. Schulman and E. Winfree. Self-replication and evolution of DNA crystals. In *Advances in Artificial Life, 8th European Conference*, volume 3630, Berlin Heidelberg, 2005. Springer-Verlag.

[33] R. Schulman and E. Winfree. Synthesis of crystals with a programmable kinetic barrier to nucleation. *P. Natl. Acad. Sci.*, 104(39):15236–15241, 2007.

[34] R. Schulman and E. Winfree. How crystals that sense and respond to their environments could evolve. *Natur. Comp.*, 7:219–237, 2008.

[35] R. Schulman and E. Winfree. Simple evolution of complex crystal species. In *DNA Computing 16*, volume 6518, Berlin Heidelberg, 2010. Springer-Verlag.

[36] G. Seelig, D. Soloveichik, D. Y. Zhang, and E. Winfree. Enzyme-free nucleic acid logic circuits. *Science*, 314:1585–1588, 2006.

[37] N. C. Seeman. Nucleic-acid junctions and lattices. *J. Theor. Biol.*, 99(2):237–247, 1982.

[38] D. Soloveichik and E. Winfree. Complexity of compact proofreading for self-assembled patterns. In *DNA Computing 11*, Berlin Heidelberg, 2005. Springer-Verlag.

[39] H. Wang. Proving theorems by pattern recognition. II. *Bell Syst. Tech J.*, 40:1–42, 1961.

[40] H. Wang. An unsolvable problem on dominoes. Technical Report BL-30 (II-15), Harvard Computation Laboratory, 1962.

[41] L. Wang, J. Xie, and P. G. Schultz. Expanding the genetic code. *Annu. Rev. Biophys. Bio.*, 35:225–249, 2006.

[42] E. Winfree. On the computational power of DNA annealing and ligation. In *DNA Based Computers*, pages 199–221, 1995.

[43] E. Winfree. Simulations of computing by self-assembly. Technical Report CS-TR:1998.22, Caltech, 1998.

[44] E. Winfree and R. Bekbolatov. Proofreading tile sets: Error-correction for algorithmic self-assembly. In J. Chen and J. Reif, editors, *DNA Computing 9*, volume LNCS 2943, pages 126–144, Berlin Heidelberg, 2004. Springer-Verlag.

[45] E. Winfree, F. Liu, L. A. Wenzler, and N. C. Seeman. Design and self-assembly of two-dimensional DNA crystals. *Nature*, 394:539–544, 1998.

[46] B. Yurke, A. J. Turberfield, A. P. Mills, Jr., F. C. Simmel, and J. L. Neumann. A DNA-fuelled molecular machine made of DNA. *Nature*, 406:605–608, 2000.

[47] D. Y. Zhang and B. Yurke. A DNA superstructure-based replicator without product inhibition. *Natur. Comp.*, 5:183–202, 2006.

[48] M. Zuker. Calculating nucleic acid secondary structure. *Curr. Opin. Chem. Biol.*, 10:303–310, 2000.

About the author

Rebecca Schulman's work focuses on complex, adaptive self-assembly processes and building simple, synthetic mimics of biological processes. Dr. Schulman studied computer science and mathematics at MIT, where her research focused on artificial intelligence. After several years writing search engines and natural language processing software in Silicon Valley, she returned to academia. Dr. Schulman worked with Erik Winfree at the California Institute of Technology, where she received her PhD in computation and neural systems in 2007. Dr. Schulman is currently a Miller research fellow in the physics department at U.C. Berkeley where she works in Jan Liphardt's laboratory. In August, she joined Johns Hopkins University's chemical and biomolecular engineering department as an assistant professor.

Homepage: http://www.schulmanlab.net/
Email: rschulm3@jhu.edu

Freshly Printed

Symbolic Regression for Knowledge Discovery

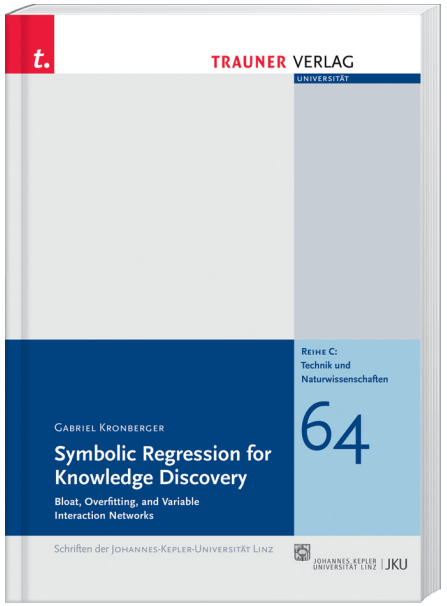
Bloat, Overfitting, and Variable Interaction Networks

Dipl.-Ing. Dr. Gabriel Kronberger

This work describes an approach for data analysis based on symbolic regression and genetic programming, that produces an overall view of the dependencies of all variables of a system. The identified dependencies are represented in form of a variable interaction network.

In the first part of this work, this approach is described in detail. Important issues are the prevention of bloat and overfitting, the simplification of models, and the identification of relevant input variables. In this context, different methods for bloat control are presented and compared. In addition, a novel way to detect and reduce overfitting is presented and analyzed.

The second part of this work demonstrates how comprehensive symbolic regression can be applied for analysis of real-world systems. Variable interaction networks for a blast furnace process and an industrial chemical process are presented and discussed. Additionally, the same approach is also applied on an economic data set to identify macro-economic dependencies.



Gabriel Kronberger: Symbolic Regressionfor Knowledge Discovery: Bloat, Overfitting, and Vari- able Interaction Networks - 1. Edition 2011, 214 pages, A5, paperback, ISBN 978-3-85499-875-4

Calls and Calendar

January 2012

Learning and Intelligent OptimizatioN Conference - LION 6
January 16-20, 2012, Paris, France
Homepage: http://www.intelligent-optimization.org/LION6/
Call for Papers: www
Notification to authors: November 28, 2011
Conference dates: January 16-20, 2012
Camera ready for post-proceedings: February 24, 2012

The LION conference aims at exploring the intersections between machine learning, artificial intelligence, mathematical programming and algorithms for hard optimization problems. The main purpose of the event is to bring together experts from all these areas to present and discuss new ideas, new methods, general trends, challenges and opportunities in applications as well as in research aiming at algorithmic advances. The conference program will consist of plenary presentations, introductory and advanced tutorials, technical presentations, and it will give ample time for discussions.

Relevant Research Areas

LION 6 solicits contributions dealing with all aspects of learning and intelligent optimization. Topics of interest include, but are not limited to:

- Metaheuristics such as tabu search, iterated local search, evolutionary algorithms, ant colony optimization, particle swarm optimization, and memetic algorithms
- Hybridizations of metaheuristics with other techniques for optimization
- Hyperheuristics and automatic design of heuristics
- Machine learning-aided search and optimization
- Algorithm portfolios and off-line tuning methods
- Reactive search optimization, autonomous search, adaptive and self-adaptive algorithms
- Specific adaptive metaheuristic techniques applied to propositional satisfiability, scheduling and planning, routing and logistics problems
- Interface(s) between discrete and continuous optimization
- Algorithms for dynamic, stochastic and multi-objective problems
- Multiscale and multilevel methods

For all the previous approaches:

- Experimental analysis and modeling
- Parallelization techniques
- Theoretical foundations
- Innovative applications

High-quality scientific contributions to these topics are solicited, in addition to advanced case studies from interesting, high-impact application areas.

Further Information

Up-to-date information will be published on the web site www.intelligent-optimization.org/LION6. For information about local arrangements, registration forms, etc., please refer to the above-mentioned web site or contact the organizers.

LION 6 Conference and Technical co-chairs

- Youssef Hamadi, Microsoft Research, UK (youssefh@microsoft.com)
- Marc Schoenauer, INRIA, France (Marc.Schoenauer@inria.com)

April 2012

Evostar 2012 - EuroGP, EvoCOP, EvoBIO, EvoMusart and EvoApplications

April 11-13, 2012, Malaga, Spain

Homepage: http://www.evostar.org

Flyer: pdf

Deadline November 30, 2011

Notification to authors: January 14, 2012

Camera-ready deadline: February 5, 2012

evo* comprises the premier co-located conferences in the field of Evolutionary Computing: **eurogp**, **evocop**, **evobio**, **evomusart** and **evoapplications**.

Featuring the latest in theoretical and applied research, evo* topics include recent genetic programming challenges, evolutionary and other meta-heuristic approaches for combinatorial optimization, evolutionary algorithms, machine learning and data mining techniques in the biosciences, in numerical optimization, in music and art domains, in image analysis and signal processing, in hardware optimization and in a wide range of applications to scientific, industrial, financial and other real-world problems.

eurogp (flyer)

15th European Conference on Genetic Programming Papers are sought on topics strongly related to the evolution of computer programs, ranging from theoretical work to innovative applications.

evocop (flyer)

12th European Conference on Evolutionary Computation in Combinatorial Optimization Practical and theoretical contributions are invited, related to evolutionary computation techniques and other meta-heuristics for solving combinatorial optimization problems.

evobio (flyer)

10th European Conference on Evolutionary Computation, Machine Learning and Data Mining in Computational Biology Emphasis is on evolutionary computation and other advanced techniques addressing important problems in molecular biology, proteomics, genomics and genetics, that have been implemented and tested in simulations and on real-life datasets.

evomusart (flyer)

1st International Conference and 10th European Event on Evolutionary and Biologically Inspired Music, Sound, Art and Design

evoapplications (flyer)

European Conference on the Applications of Evolutionary Computation

evocomnet

9th European event on nature-inspired techniques for telecommunication networks and other parallel and distributed systems

evocomplex

3rd European event on algorithms and complex systems

evofin

6th European event on evolutionary and natural computation in finance and economics

evogames

4th European event on bio-inspired algorithms in games

evohot

7th European event on bio-inspired heuristics for design automation

evoiasp

14th European event on evolutionary computation in image analysis and signal processing

evonum

5th European event on bio-inspired algorithms for continuous parameter optimisation

evopar

1st European event on parallel and distributed Infrastructures

evorisk

1st European event on computational intelligence for risk management, security and defence applications

evostim

7th European event on nature-inspired techniques in scheduling, planning and timetabling

evostoc

9th European event on evolutionary algorithms in stochastic and dynamic environments

evotranslog

6th European event on evolutionary computation in transportation and logistics

July 2012

GECCO 2012 - Genetic and Evolutionary Computation Conference
July 7-11, 2012, Philadelphia, PA, USA
Homepage: http://www.sigevo.org/gecco-2012
Deadline January 13, 2012
Author notification: March 13, 2012
Workshop and tutorial proposals submission: November 07, 2011
Notification of workshop and tutorial acceptance: November 28, 2011

The Genetic and Evolutionary Computation Conference (GECCO-2012) will present the latest high-quality results in the growing field of genetic and evolutionary computation.

Topics include: genetic algorithms, genetic programming, evolution strategies, evolutionary programming, real-world applications, learning classifier systems and other genetics-based machine learning, evolvable hardware, artificial life, adaptive behavior, ant colony optimization, swarm intelligence, biological applications, evolutionary robotics, coevolution, artificial immune systems, and more.

Organizers

General Chair:	Jason Moore
Editor-in-Chief:	Terence Soule
Publicity Chair:	Xavier Llorá
Tutorials Chair:	Gabriela Ochoa
Students Chair:	Josh Bongard
Workshops Chair:	Bill Rand
Competitions Chairs:	Daniele Loiacono
Business Committee:	Wolfgang Banzhaf
	Marc Schoenauer
EC in Practice Chairs:	Jörn Mehnen
	Thomas Bartz-Beielstein,
	David Davis

Important Dates

Paper Submission Deadline	January 13, 2012
Decision Notification	March 13, 2012
Camera-ready Submission	April 9, 2012

To Propose a Tutorial or Workshop

A detailed call for workhop and tutorial proposals will be posted later so stay tuned! Meanwhile, for enquiries regarding tutorials contact gecco2012tutorials@sigevolution.org while for enquiries about workshops contact gecco2012workshops@sigevolution.org.

More Information

Visit www.sigevo.org/gecco-2012 for information about electronic submission procedures, formatting details, student travel grants, the latest list of tutorials and workshop, late-breaking papers, and more.

Contact

For general help and administrative matters contact GECCO support at gecco2012@sigevolution.org

GECCO is sponsored by the Association for Computing Machinery Special Interest Group for Genetic and Evolutionary Computation.

September 2012

PPSN 2012 – International Conference on Parallel Problem Solving From Nature

September 1-5, 2012, Taormina, Italy

Homepage: http://www.dmi.unict.it/ppsn2012/

Call for paper: www

Email: ppsn2012@dmi.unict.it

Paper Submission Deadline: March 15, 2012

Author Notification: June 1, 2012

Workshop Proposals Submission: October 15, 2011

PPSN XII will showcase a wide range of topics in Natural Computing including, but not restricted to: Evolutionary Computation, Quantum Computation, Molecular Computation, Neural Computation, Artificial Life, Swarm Intelligence, Artificial Ant Systems, Artificial Immune Systems, Self-Organizing Systems, Emergent Behaviors, and Applications to Real-World Problems.

Paper Presentation

Following the now well-established tradition of PPSN conferences, all accepted papers will be presented during small poster sessions of about 16 papers. Each session will contain papers from a wide variety of topics, and will begin by a plenary quick overview of all papers in that session by a major researcher in the field. Past experiences have shown that such presentation format led to more interactions between participants and to a deeper understanding of the papers. All accepted papers will be published in the LNCS Proceedings.

Paper Submission

Researchers are invited to submit original work in the field of natural computing as papers of not more than 10 pages. Authors are encouraged to submit their papers in LaTeX. Papers must be submitted in Springer Verlag's LNCS style through the conference homepage, here.

IEEE Conference on Computational Intelligence and Games (CIG-2012)

September 12-15, 2012, Granada, Spain

Homepage: http://geneura.ugr.es/cig2012/

Flyer: pdf

Submission deadline: April 15, 2012

Decision notification: June 1, 2012

Camera-ready submission: June 15, 2012

Conference: September 12-15, 2012

Aim and Scope

Games have proven to be an ideal domain for the study of computational intelligence as not only are they fun to play and interesting to observe, but they provide competitive and dynamic environments that model many real-world problems. Additionally, methods from computational intelligence promise to have a big impact on game technology and development, assisting designers and developers and enabling new types of computer games. The 2010 IEEE Conference on Computational Intelligence and Games brings together leading researchers and practitioners from academia and industry to discuss recent advances and explore future directions in this quickly moving field.

Topics of interest include, but are not limited to:

- Learning in games
- Coevolution in games
- Neural-based approaches for games
- Fuzzy-based approaches for games
- Player/Opponent modeling in games
- CI/AI-based game design
- Multi-agent and multi-strategy learning
- Applications of game theory
- CI for Player Affective Modeling
- Intelligent Interactive Narrative
- Imperfect information and non-deterministic games
- Player satisfaction and experience in games
- Theoretical or empirical analysis of CI techniques for games
- Comparative studies and game-based benchmarking
- Computational and artificial intelligence in:
    - Video games
    - Board and card games
    - Economic or mathematical games
    - Serious games
    - Augmented and mixed-reality games
    - Games for mobile platforms

The conference will consist of a single track of oral presentations, tutorial and workshop/special sessions, and live competitions. The proceedings will be placed in IEEE Xplore, and made freely available on the conference website after the conference.

Conference Committee

General Chair:	Antonio J. Fernández Leiva
Program Chairs:	Simon Lucas, Sung-Bae Cho, and Magy Seif El-Nasr
Publicity Chair:	Antonio M. Mora García
Social Media Chair:	Juan J. Merelo
Finance Chair:	Pedro A. Castillo
Proceedings Chairs:	Mike Preuss and Anna I. Esparcia
Competition Chair:	Julian Togelius
Special Sessions and Tutorials Chair:	Georgios Yannakakis
Local Chairs:	Carlos Cotta Porras, Antonio J. Fernández Leiva, Antonio M. Mora García, Juan J. Merelo, and Pedro A. Castillo

Important Dates

Tutorial proposals:	15 March 2012
Paper submission:	15 April 2012
Decision Notification:	1 June 2012
Camera-ready:	15 June 2012
Conference:	12-15 September 2012

For more information please visit: http://geneura.ugr.es/cig2012/

Seventh International Conference on Swarm Intelligence

September 12-14, 2012. Brussels, Belgium

Homepage: http://iridia.ulb.ac.be/ants2012

Deadline March 2, 2012

Swarm intelligence is a relatively new discipline that deals with the study of self-organizing processes both in nature and in artificial systems. Researchers in ethology and animal behavior have proposed many models to explain interesting aspects of social insect behavior such as self-organization and shape-formation. Recently, algorithms and methods inspired by these models have been proposed to solve difficult problems in many domains.

An example of a particularly successful research direction in swarm intelligence is ant colony optimization, the main focus of which is on discrete

optimization problems. Ant colony optimization has been applied successfully to a large number of difficult discrete optimization problems including the traveling salesman problem, the quadratic assignment problem, scheduling, vehicle routing, etc., as well as to routing in telecommunication networks. Another interesting approach is that of particle swarm optimization, that focuses on continuous optimization problems. Here too, a number of successful applications can be found in the recent literature. Swarm robotics is another relevant field. Here, the focus is on applying swarm intelligence techniques to the control of large groups of cooperating autonomous robots.

ANTS 2012 will give researchers in swarm intelligence the opportunity to meet, to present their latest research, and to discuss current developments and applications.

The three-day conference will be held in Brussels, Belgium, on September 12-14, 2012.

Relevant Research Areas

ANTS 2012 solicits contributions dealing with any aspect of swarm intelligence. Typical, but not exclusive, topics of interest are:

- Behavioral models of social insects or other animal societies that can stimulate new algorithmic approaches.
- Empirical and theoretical research in swarm intelligence.
- Application of swarm intelligence methods, such as ant colony optimization or particle swarm optimization, to real-world problems.
- Theoretical and experimental research in swarm robotics systems.

Publication Details Conference proceedings will be published by Springer in the LNCS. series.

The journal Swarm Intelligence will publish a special issue dedicated to ANTS 2012 that will contain extended versions of the best research works presented at the conference. Further details will soon be published on the web site.

Best Paper Award A best paper award will be presented at the conference.

Further Information Up-to-date information will be published on the web site http://iridia.ulb.ac.be/ants2012/. For information about local arrangements, registration forms, etc., please refer to the above-mentioned web site or contact the local organizers at the address below.

Conference Address

ANTS 2012

IRIDIA CP 194/6	Tel +32-2-6502729
Université Libre de Bruxelles	Fax +32-2-6502715
Av. F. D. Roosevelt 50	http://iridia.ulb.ac.be/ants2012
1050 Bruxelles, Belgium	email: ants@iridia.ulb.ac.be

Important Dates

Submission deadline	March 2, 2012
Notification of acceptance	May 3, 2012
Camera ready copy	May 17, 2012
Conference	September 12–14, 2012

About the Newsletter

SIGEVOlution is the newsletter of SIGEVO, the ACM Special Interest Group on Genetic and Evolutionary Computation.

To join SIGEVO, please follow this link [WWW]

Contributing to SIGEVOlution

We solicit contributions in the following categories:

Art: Are you working with Evolutionary Art? We are always looking for nice evolutionary art for the cover page of the newsletter.

Short surveys and position papers: We invite short surveys and position papers in EC and EC related areas. We are also interested in applications of EC technologies that have solved interesting and important problems.

Software: Are you are a developer of an EC software and you wish to tell us about it? Then, send us a short summary or a short tutorial of your software.

Lost Gems: Did you read an interesting EC paper that, in your opinion, did not receive enough attention or should be rediscovered? Then send us a page about it.

Dissertations: We invite short summaries, around a page, of theses in EC-related areas that have been recently discussed and are available online.

Meetings Reports: Did you participate in an interesting EC-related event? Would you be willing to tell us about it? Then, send us a short summary, around half a page, about the event.

Forthcoming Events: If you have an EC event you wish to announce, this is the place.

News and Announcements: Is there anything you wish to announce? This is the place.

Letters: If you want to ask or to say something to SIGEVO members, please write us a letter!

Suggestions: If you have a suggestion about how to improve the newsletter, please send us an email.

Contributions will be reviewed by members of the newsletter board.

We accept contributions in LaTeX, MS Word, and plain text.

Enquiries about submissions and contributions can be emailed to editor@sigevolution.org.

All the issues of SIGEVOlution are also available online at www.sigevolution.org.

Notice to Contributing Authors to SIG Newsletters

By submitting your article for distribution in the Special Interest Group publication, you hereby grant to ACM the following non-exclusive, perpetual, worldwide rights:

- to publish in print on condition of acceptance by the editor
- to digitize and post your article in the electronic version of this publication
- to include the article in the ACM Digital Library
- to allow users to copy and distribute the article for noncommercial, educational or research purposes

However, as a contributing author, you retain copyright to your article and ACM will make every effort to refer requests for commercial use directly to you.