The Surprising Creativity of Digital Evolution: A Collection of Anecdotes from the Evolutionary Computation and Artificial Life Research Communities

Joel Lehman*
Uber AI Labs
lehman.154@gmail.com

Jeff Clune*
OpenAI
jclune@gmail.com

Dusan Misevic*
Université de Paris
INSERM U1284
Center for Research and Interdisciplinarity
dule@alife.org

Christoph Adami, Michigan State University.
Lee Altenberg, Univeristy of Hawai'i at Mānoa.
Julie Beaulieu, Université Laval.
Peter J. Bentley, University College London.
Samuel Bernard, INRIA, Institut Camille Jordan, CNRS.
Guillaume Beslon, Université de Lyon, INRIA, CNRS.
David M. Bryson, Michigan State University.
Nick Cheney, University of Vermont.
Patryk Chrabaszcz, University of Freiburg.
Antoine Cully, Imperial College London.
Stephane Doncieux, Sorbonne Universités, UPMC Univ Paris 06, CNRS, Institute of Intelligent Systems and Robotics (ISIR).
Fred C. Dyer, Michigan State University.
Kai Olav Ellefsen, University of Oslo.
Robert Feldt, Chalmers University of Technology.
Stephan Fischer, INRA.
Stephanie Forrest, Arizona State University.
Antoine Fŕenoy, Université Grenoble Alpes.
Christian Gagńe, Université Laval.
Leni Le Goff, Sorbonne Universités, UPMC Univ Paris 06, CNRS, Institute of Intelligent Systems and Robotics (ISIR).
Laura M. Grabowski, State University of New York at Potsdam.
Babak Hodjat, Cognizant Technology Solutions.
Frank Hutter, University of Vermont.
Laurent Keller, Department of Fundamental Microbiology, University of Lausanne.
Carole Knibbe, Université de Lyon, INRIA, CNRS.
Peter Krcah, Charles University Prague.
Richard E. Lenski, Michigan State University.
Hod Lipson, Columbia University.
Robert MacCurdy, University of Colorado.
Carlos Maestre, Sorbonne Universités, UPMC Univ Paris 06, CNRS, Institute of Intelligent Systems and Robotics (ISIR).
Risto Miikkulainen, University of Texas at Austin.

Abstract Evolution provides a creative fount of complex and subtle adaptations that often surprise the scientists who discover them. However, the creativity of evolution is not limited to the natural world: Artificial organisms evolving in computational environments have also elicited surprise and wonder from the researchers studying them. The process of evolution is an *algorithmic* process that transcends the substrate in which it occurs. Indeed, many researchers in the field of digital evolution can provide examples of how their evolving algorithms and organisms have creatively subverted their expectations or intentions, exposed unrecognized bugs in their code, produced unexpectedly adaptations, or engaged in behaviors and outcomes, uncannily convergent with ones found in nature. Such stories routinely reveal surprise and creativity by evolution in these digital worlds, but they rarely fit into the standard scientific narrative. Instead they are often treated as mere obstacles to be overcome, rather than results that warrant study in their own right. Bugs are fixed, experiments are refocused, and one-off surprises are collapsed into a single data point. The stories themselves are traded among researchers through oral tradition, but that mode of information transmission is inefficient and prone to error and outright loss. Moreover, the fact that these stories tend to be shared only among practitioners means that many natural scientists do not realize how interesting and lifelike digital organisms are and how natural their evolution can be. To our knowledge, no collection of such anecdotes has been published before. This article is the crowd-sourced product of researchers in the fields of artificial life and evolutionary computation who have provided first-hand accounts of such cases. It thus serves as a written, fact-checked collection of scientifically important and even entertaining stories. In doing so we also present here substantial evidence that the existence and importance of evolutionary surprises extends beyond the natural world, and may indeed be a universal property of all complex evolving systems.

* Organizing, lead, and contact authors, who contributed equally. Jeff Clune's work done while at Uber AI Labs.

Artificial Life 26: 274–306 (2020) https://doi.org/10.1162/artl_a_00319

Sara Mitri, Department of Fundamental Microbiology, University of Lausanne.
David E. Moriarty, Apple Inc.
Jean-Baptiste Mouret, INRIA Nancy Grand—Est.
Anh Nguyen, Auburn University.
Charles Ofria, Michigan State University.
Marc Parizeau, Université Laval.
David Parsons, Michigan State University.
Robert T. Pennock, Michigan State University.
William F. Punch, Michigan State University.
Thomas S. Ray, University of Oklahoma.
Marc Schoenauer, INRIA, Université Paris-Saclay.
Eric Schulte, GrammaTech.
Karl Sims
Kenneth O. Stanley, Uber AI Labs and University of Central Florida.
François Taddei, Center for Research and Interdisciplinarity, INSERM U1284, Université de Paris.
Danesh Tarapore, University of Southampton.
Simon Thibault, Université Laval.
Richard Watson, University of Southampton.
Westley Weimer, University of Virginia.
Jason Yosinski, Uber AI Labs.

Keywords
Surprise, creativity, digital evolution, experimental evolution, genetic algorithms, evolutionary computation

1 Introduction

Evolution provides countless examples of creative, surprising, and amazingly complex solutions to life's challenges. Some flowers act as acoustic beacons to attract echo-locating bats [121], extremophile microbes repair their DNA to thrive in the presence of extreme radiation [89], bombardier beetles repel predators with explosive chemical reactions [131], and parasites reprogram host brains, inducing suicide for the parasite's own gain [78]. Many more examples abound, covering the full range of biological systems [24, 49, 88]. Even seasoned field biologists are still surprised by the new adaptations they discover, which they express in popular press accounts of their work [77, 100, 111], but only more rarely in official academic publications, for example, in [41] but not in [76, 142].

Thus, the process of biological evolution is extremely creative [9, 29], at least in the sense that it produces surprising and complex solutions that would be deemed creative if produced by a human. But the creativity of evolution need not be constrained to the organic world. Independently of its physical medium, evolution can happen wherever replication, variation, and selection intersect [23, 27]. Thus, evolution can be instantiated *digitally* [25, 75], as a computer program, either to study evolution experimentally or to solve engineering problems through directed digital breeding. Similarly to biological evolution, digital evolution experiments often produce strange, surprising, and creative results. Indeed, evolution often reveals that researchers actually asked for something far different from what they thought they were asking for—not so different from those stories in which a genie satisfies the letter of a request in an unanticipated way. Sometimes evolution reveals hidden bugs in code or displays surprising convergence with biology. Other times, evolution simply surprises and delights by producing clever solutions that investigators did not consider or had thought impossible.

While some such unexpected results have been published [9, 55, 90, 109], most have not, and they have not previously been presented together, as they are here. One obstacle to their dissemination is that such unexpected results often result from evolution *thwarting* a researcher's intentions: by exploiting a bug in the code, by optimizing an uninteresting feature, or by failing to answer the intended research question. That is, such behavior is often viewed as a frustrating distraction, rather than a phenomenon of scientific interest. Additionally, surprise is *subjective* and thus fits poorly with the objective language and narrative expected in scientific publications. As a result, most anecdotes have been spread only through word of mouth, providing laughs and discussion in research groups, at conferences, and as comic relief during talks. But such communications fail to inform the field as a whole in a lasting and stable way.

Importantly, these stories of digital evolution "outsmarting" the researchers who study it provide more than an exercise in humility; they also provide insight and constructive knowledge for practitioners, because they show the pervasiveness of such obstacles and how, when necessary, they can be overcome. Furthermore, these cases demonstrate that robust digital models of evolution do not blindly reflect the desires and biases of their creators, but instead they have depth sufficient to yield unexpected results and new insights. Additionally, these cases may be of interest to researchers in evolutionary biology as well as animal and plant breeding, because of their compelling parallels to the creativity of biological evolution.

For these reasons, this article draws attention to surprise and creativity in algorithmic evolution, aiming to document, organize, and disseminate information that, until now, has been passed down through oral tradition, which is prone to error and outright loss. To compile this archive, the organizers of this article sent out a call for anecdotes to digital evolution mailing lists and succeeded in reaching both new and established researchers in the field. We then selected from 90 submissions to produce this "greatest hits" collection of anecdotes, and all co-authors of each selected submission were included as co-authors on the article. Before presenting these stories, the next section provides background information useful for those outside the fields of digital evolution and evolutionary computation.

2 Background

2.1 Evolution and Creativity

Intuitively, evolution's creativity is evident from observing life's vast and complex diversity. This sentiment is well captured by Darwin's famous concluding thoughts in *On the Origin of Species*, where surveying the myriad co-inhabitants of a single tangled bank leads to grand reflections on the "endless forms most beautiful" that were produced by evolution [22, p. 307]. Varieties of life diverge wildly along axes of complexity, organization, habitat, metabolism, and reproduction, spanning from single-celled prokaryotes to quadrillion-celled whales [147]. Since life's origin, biodiversity has expanded as evolution has conquered the sea, land, and air, inventing countless adaptations along the way [147].

The functional abilities granted by such adaptations greatly exceed the capabilities of current human engineering, which has yet to produce robots capable of robust self-reproduction or of autonomous exploration in the real world, or that demonstrate human-level intelligence. It would thus be parochial to deny attributing creativity to the evolutionary process, if human invention of such artefacts would garner the same label. Admittedly, "creativity" is a semantically rich word that can take on many different meanings. Thus to avoid a semantic and philosophical quagmire, while acknowledging that other definitions and opinions exist, we here adopt the "standard definition" [116]: Creativity requires inventing something both original (i.e., novel) and effective (i.e., functional). Many of evolution's inventions clearly meet this benchmark.

The root of natural evolution's creativity, in this standard sense of the term, is the sieve of reproduction. This sieve can be satisfied in many different ways, and as a result, evolution has produced a cornucopia of divergent outcomes [29, 147]. For example, nature has invented many

different ways to siphon the energy necessary for life's operation from inorganic sources (e.g., from the sun, iron, or ammonia), and it has created many different wing structures for flight among insects, birds, mammals, and ancient reptiles. Evolution's creative potential has also been bootstrapped from ecological interactions; the founding of one niche often opens others, for example, through predation, symbiosis, parasitism, or scavenging. Although evolution lacks the foresight and intentionality of human creativity, structures evolved for one functionality are often opportunistically adapted for other purposes, a phenomenon known as exaptation [54]. For example, a leading theory is that feathers first evolved in dinosaurs for temperature regulation [71] and were later exapted for flight in birds. Even in the absence of direct foresight, studies of evolvability suggest that genomic architecture itself can become biased toward increasing creative potential [62, 65, 66].

One component of evolution is the selective pressures that adapt a species to better fit its environment, which often results in creativity within that species. That is, meeting evolutionary challenges requires inventing effective solutions, such as better protection from predators or from natural elements like wind or radiation. Beyond creativity within species, there are also evolutionary forces that promote creative *divergence*, that is, that lead to the accumulation of novel traits or niches. One such force is negative frequency-dependent selection [35]; this dynamic occurs when some traits are adaptive only when rare, which promotes the evolution of organisms that demonstrate different traits. Another divergent force is adaptive radiation [118], which occurs when access to new opportunities allows an organism to rapidly diversify into a range of new species, as when a new modality such as flight is discovered. In this way, evolution is driven toward effectiveness (being well adapted and functional) and toward originality both through the optimizing force of natural selection and through divergent forces, thereby producing artefacts that meet both criteria of the standard definition of creativity.

One aim of this article is to highlight that such creativity is not limited to the biological medium, but is also a common feature of digital evolution. We continue by briefly reviewing digital evolution.

2.2 Digital Evolution

Inspired by biological evolution, researchers in the field of digital evolution study evolutionary processes embodied in digital substrates. The general idea is that there exist abstract principles underlying biological evolution that are independent of the physical medium [23], and that these principles can be effectively implemented and studied within computers [84]. As noted by Daniel Dennett, "evolution will occur whenever and wherever three conditions are met: replication, variation (mutation), and differential fitness (competition)" [26, pp. E83–E92]; no particular molecule (e.g., DNA or RNA) or substrate (e.g., specific physical embodiment) is required.

In nature, heredity is enabled through replicating genetic molecules, and variation is realized through mechanisms like copy errors and genetic recombination. Selection in biological evolution results from how survival and reproduction are a logical requirement for an organism's genetic material to persist. The insight behind digital evolution is that processes fulfilling these roles of replication, variation, and selection can be implemented in a computer, resulting in an *evolutionary algorithm* (EA) [25]. For example, replication can be instantiated simply by copying a data structure (i.e., a digital genome) in memory, and variation can be introduced by randomly perturbing elements within such a data structure. Selection in an EA can be implemented in many ways, but the two most common are digital analogues of artificial and natural selection in biological evolution. Because the similarities and differences between these kinds of selection pressure are important for understanding many of the digital evolution outcomes, we next describe them in greater detail.

Artificial selection in biological evolution is exemplified by horse breeders who actively decide which horses to breed together, hoping to enhance certain traits, for example, by breeding together the fastest ones, or the smallest ones. In this mode, selection reflects human goals. Similarly, in digital evolution a researcher can implement a *fitness function* as an automated criterion for selection. A fitness function is a metric describing which phenotypes are preferred over others, reflecting the researcher's goal for what should be evolved. For example, if applying an EA to design a stable gait

for a legged robot, an intuitive fitness function might be to measure how far a controlled robot walks before it falls. Selection in this EA would breed together those robot controllers that traveled farthest, in hopes that their offspring might travel even farther. This mode of selection is most common in engineering applications, where digital evolution is employed to achieve a desired outcome.

The other common mode of digital selection implements natural selection as it occurs in nature, where evolution is open-ended. The main difference is that in this mode there is no specific target outcome, and no explicit fitness function. Instead, digital organisms compete for limited resources, which could be artificial nutrients, CPU cycles needed to replicate their code, or digital storage space in which to write their genomes [105, 113]. Given variation within the population, some organisms will survive long enough to reproduce and propagate their genetic material, while others will disappear, which enables evolution to occur naturally. Typically, digital evolution systems and experiments of this sort do not serve a direct engineering purpose, but are instead used as a tool to study principles of life and evolution in an easier setting than natural biology; that is, they provide *artificial life* model systems for use in experimental evolution [75]. Note that the field of digital evolution overlaps with the study of EAs and of artificial life, but is not synonymous with it. For the purposes of this article, we define digital evolution as evolutionary processes in which the algorithm of evolution and the genetic material evolved are instantiated digitally. As a result, digital evolution does not include "wet" artificial life, which seeks alternative *physical* substrates for life [7]. Digital evolution does encompass paradigms like virtual creatures evolving completely in silico [126]. It also includes the fields of evolvable hardware and evolutionary robotics, where evolved digital programs or controllers are deployed on real-world devices [13].

One persistent misconception of digital evolution is that, because it is instantiated in a computational substrate, it lacks relevance to the study of biological evolution. Yet both philosophical arguments [23, 26, 27, 82, 109] and high-profile publications [2, 16, 18–21, 52, 59, 60, 83, 85, 86, 92, 141, 146, 148] suggest that digital evolution can be a useful tool to aid and complement the study of biological evolution. Indeed, these evolving systems can be seen as real instances of evolution, rather than mere simulation of evolution [110].

2.3 Surprise from Algorithms and Simulations

At first, it may seem counterintuitive that a class of algorithms can consistently surprise the researchers who wrote them. Here we define surprise broadly as observing an outcome that significantly differs from expectations, whether those expectations arise from intuitions, predictions from past experiences, or theoretical models. Note that such surprise is a feature of the experimenter's *subjective experience*. Because an algorithm is a formal list of unambiguous instructions that execute in a prescribed order, it would seem sufficient to examine any algorithm's description to predict the full range of its possible outcomes, undermining any affordance for surprise. However, a well-known result in theoretical computer science is that, for many computer programs, the outcome of a program *cannot* be predicted without actually running it [137]. Indeed, within the field of complex systems it is well known that simple programs can yield complex and surprising results when executed [43, 74]; such results relate to the broader concept of *emergence* [6, 58], wherein a phenomenon resulting from lower-level parts is best understood at a more abstract level of description (e.g., that a copper atom exists at the tip of a statue's nose is better understood at the level of history and politics than at the level of physics [28]).

This basic fact can be counterintuitive at first. Interactions with modern software, which is explicitly designed to be predictable, may understandably prime us with the expectation that innovation and surprise cannot be captured by a computer algorithm. However, if surprising innovations are a hallmark of biological evolution, then the default *expectation* ought to be that computer models that instantiate fundamental aspects of the evolutionary process would naturally manifest similarly creative output. While we offer no formal proof in this article of digital evolution's ability to generate surprise, the diversity of anecdotes presented next highlights how common and widespread such surprising results are in practice. It is important to note here that a facet of human psychology, called

hindsight bias, often obscures appreciating the subjective surprise of another person [115]. In other words, humans often overestimate how predictable an event was after the fact. For many of the anecdotes that follow, a post hoc understanding of the result is possible, which may lead the reader to discount its surprisingness. While mediating this kind of cognitive bias is challenging, we mention it here in hopes that readers might grant the original experimenters leeway for their inability to anticipate what perhaps is easily recognized in retrospect. In other words, we believe that experimenters are in general well situated to objectively report on their subjective experience of surprise.

3 Routine Creative Surprise in Digital Evolution

The next subsections present 32 curated anecdotes representing the work of over 50 researchers. In reviewing the anecdotes, we found that they roughly clustered into four representative categories: *misspecified fitness functions*, in which digital evolution reveals the divergence between what an experimenter is asking of evolution and what they *think* they are asking; *unintended debugging*, in which digital evolution reveals and exploits previously unknown software or hardware bugs; *exceeded experimenter expectations*, in which digital evolution discovers solutions that go beyond what an experimenter thought evolution would produce; and *convergence with biology*, in which digital evolution discovers solutions surprisingly convergent with those found in nature, despite vast divergence in medium and conditions.

3.1 Misspecified Fitness Functions

When applying digital evolution to solve practical problems, the most common approach is for an experimenter to choose a fitness function that reflects the desired objective of search. Such fitness functions are often simple quantitative measures that seem to intuitively capture the critical features of a successful outcome. These measures are a linchpin of EAs, as they serve as funnels to direct search: Breeding is biased toward individuals with a high fitness score, in hopes that they will lead to further fitness improvements, ultimately to culminate in the desired outcome.

This approach resembles the process of animal breeding and relies on the same evolutionary principles for its success. However, as the following anecdotes illustrate, well-intentioned quantitative measures are often maximized in counterintuitive ways. That is, experimenters often overestimate how accurately their quantitative measure reflects the underlying *qualitative* success they have in mind. This mistake is known as confusing the map with the territory (the metric is the map, whereas what the experimenter intends is the actual territory [64]).

Exacerbating the issue, it is often *functionally simpler* for evolution to exploit loopholes in the quantitative measure than it is to achieve the actual desired outcome. Just as well-intentioned metrics in human society can become corrupted by direct pressure to optimize them (known as Campbell's law [14] or Goodhart's law [53]), digital evolution often acts to fulfill the letter of the law (i.e., the fitness function) while ignoring its spirit. We often ascribe creativity to lawyers who find subtle legal loopholes, and digital evolution is often frustratingly adept at finding similar exploits.

In this section we describe many instances of this phenomenon, but the list is far from exhaustive: Encountering the divergence between what we intended to select and what we actually selected for is likely the most common way digital evolution surprises its practitioners.

3.1.1 Why Walk When You Can Somersault?

In a seminal work from 1994, Karl Sims evolved 3D virtual creatures that could discover walking, swimming, and jumping behaviors in simulated physical environments. The creatures' bodies were made of connected blocks, and their "brains" were simple computational neural networks that generated varying torque at their joints based on perceptions from their limbs, enabling realistic-looking motion. The morphology and control systems were evolved simultaneously, allowing a wide range of possible bodies and locomotion strategies. Indeed, these virtual creatures remain among the most iconic products of digital evolution [126, 127].

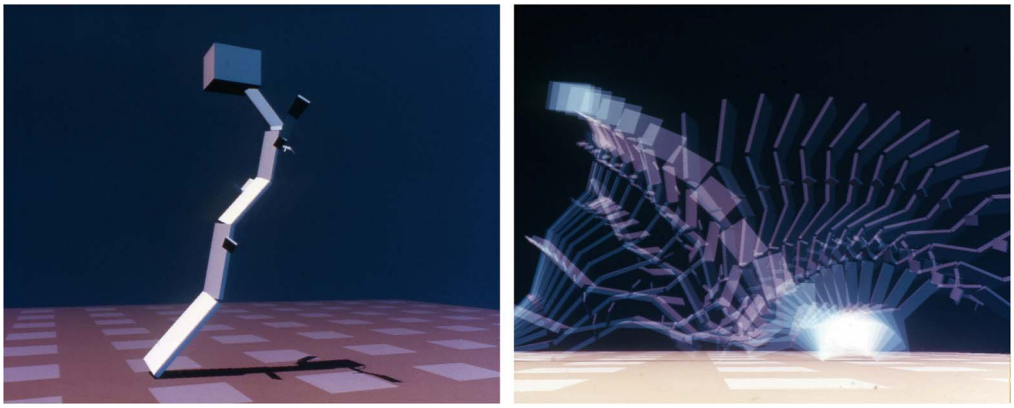

Figure 1. Exploiting potential energy to locomote. Evolution discovers that it is simpler to design tall creatures that fall strategically than it is to discover active locomotion strategies. The left photograph shows the creature at the start of a trial, and the right photograph shows snapshots of the figure over time falling and somersaulting to preserve forward momentum.

However, when Sims initially attempted to evolve locomotion behaviors, things did not go smoothly. In a simulated land environment with gravity and friction, a creature's fitness was measured as its average ground velocity during its lifetime of ten simulated seconds. Instead of inventing clever limbs or snakelike motions that could push them along (as was hoped for), the creatures evolved to become tall and rigid. When simulated, they would fall over, harnessing their initial potential energy to achieve high velocity. Some even performed somersaults to extend their horizontal velocity (Figure 1). A video of this behavior can be seen here: https://goo.gl/pnYbVh. To prevent this exploit, it was necessary to allocate time at the beginning of each simulation to relax the potential energy inherent in the creature's initial stance *before* motion was rewarded.

Building on Sims' work, but using a different simulation platform, Krcah [70] bred creatures to jump as high above the ground as possible. In the first set of experiments, each organism's fitness was calculated as the maximum elevation reached by the center of gravity of the creature during the test. This setup resulted in creatures around 15 cm tall that jumped about 7 cm off the ground. However, it occasionally also resulted in creatures that achieved high fitness values by simply having a tall, static tower for a body, reaching high elevation without any movement. In an attempt to correct this loophole, the next set of experiments calculated fitness as the furthest distance from the ground to the block that was originally closest to the ground, over the course of the simulation. When examining the quantitative output of the experiment, to the scientist's surprise, some evolved

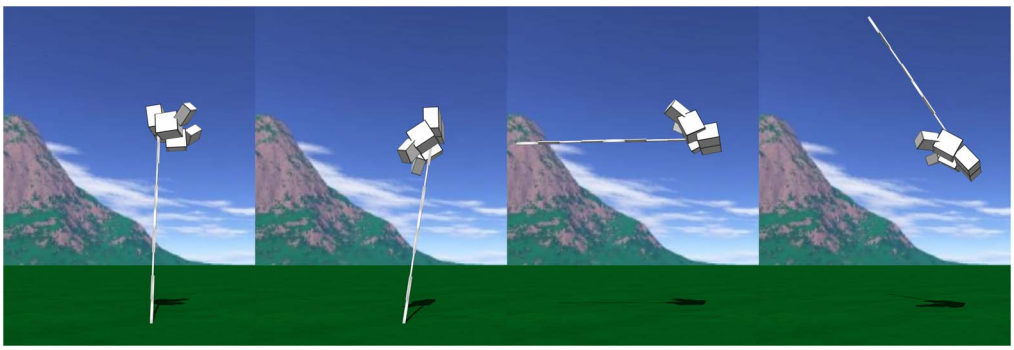

Figure 2. Exploiting potential energy to pole-vault. Evolution discovers that it is simpler to produce creatures that fall and invert than it is to craft a mechanism to actively jump.

individuals were extremely tall and also scored a nearly tenfold-improvement on their jumps! However, observing the creatures' behaviors directly revealed that evolution had discovered another cheat: somersaulting without jumping at all. The evolved body consisted of a few large blocks reminiscent of a "head" supported by a long thin vertical pole (see Figure 2).

At the start of the simulation, the individual "kicks" the foot of its pole off the ground, and begins falling head first, somersaulting its foot (originally the "lowest point" from which the jumping score is calculated) away from the ground. Doing so created a large gap between the ground and the "lowest point," thus securing a high fitness score without having to learn the intended skill of jumping. A video of the behavior can be seen here: https://goo.gl/BRyyjZ.

3.1.2 Creative Program Repair

In *automated program repair*, a computer program is designed to automatically fix other, *buggy*, computer programs. A user writes a suite of tests that validate correct behavior, and the repair algorithm's goal is to patch the buggy program so that it can pass all of the tests. One such algorithm is GenProg [45], which applies digital evolution to evolve code (called *genetic programming* [68]). GenProg's evolution is driven by a simple fitness function: the number of test cases a genetic program passes. That is, the more tests an evolved program passes, the more offspring it is likely to have.

While GenProg is often successful, sometimes strange behavior results because human-written test cases are written with human coders in mind. In practice, evolution often uncovers clever loopholes in human-written tests, sometimes achieving optimal fitness in unforeseen ways. For example, when MIT Lincoln Lab evaluated GenProg on a buggy sorting program, researchers created tests that measured whether the numbers output by the sorting algorithm were in sorted order. However, rather than actually repairing the program (which sometimes failed to correctly sort), GenProg found an easier solution: It entirely short-circuited the buggy program, having it always return an empty list, exploiting the technicality that an empty list was scored as not being out of order [145].

In other experiments, the fitness function rewarded minimizing the difference between what the program generated and the ideal target output, which was stored in text files. After several generations of evolution, suddenly and strangely, *many* perfectly fit solutions appeared, seemingly out of nowhere. Upon manual inspection, these highly fit programs still were clearly broken. It turned out that one of the individuals had deleted all of the target files when it was run! With these files missing, because of how the test function was written, it awarded perfect fitness scores to the rogue candidate and to all of its peers [122]. In another project, to avoid runaway computation, the fitness function explicitly limited a program's CPU usage. In response, GenProg produced programs that slept forever, which did not count toward CPU usage limits, as there were no computations actually performed [145]. In all cases, updating the fitness function or disallowing certain program behaviors eventually outwitted evolution's creative mischief and resulted in debugged, improved programs.

3.1.3 Why Learn When You Can Exploit an Unintended Regularity?

One common trick that digital evolution often learns is to exploit subtle unintended patterns in data, that is, ones that an experimenter may create without realizing they have provided an escape hatch for evolution to latch onto, obviating the need to confront the intended challenge of the task. For example, in a recent experiment, Ellefsen, Mouret, and Clune [33] investigated the problem of catastrophic forgetting in neural networks: that learning a new task can destroy previous knowledge. One element of the experiment was that neural connections could *change* during an agent's lifetime through neuromodulatory learning [129]. The evolution of learning was promoted by presenting objects several times and providing a reward or punishment for eating them (e.g., apple = edible, mushroom = poisonous). The edibility of each object was randomized each generation, to force the agents to learn these associations within their life instead of allowing evolution to hardcode the knowledge.

The researchers were surprised to find that high-performing neural networks evolved that contained nearly no connections or internal neurons: Even most of the sensory input was ignored. The

networks seemed to learn associations without even receiving the necessary stimuli, as if a blind person could identify poisonous mushrooms by color. A closer analysis revealed the secret to their strange performance: Rather than actually learning which objects are poisonous, the networks learned to exploit a pattern in how objects were presented. The problem was that food and poison items were always presented in an alternating fashion: food, then poison, then food, then poison, repeatedly. Evolution discovered networks that learn to simply reverse their most recent reward, so they could alternate eating and not eating indefinitely, and correctly ignoring entirely what food item was presented. In this way, evolution circumvented the intended research question, rather than shedding light on it. The problem was easily solved by randomizing the order in which items were presented.

3.1.4 Learning to Play Dumb on the Test

Like the previous one, this anecdote involves exploiting patterns inadvertently provided by researchers. In research focused on understanding how organisms evolve to cope with high-mutation-rate environments [146], Ofria sought to disentangle the beneficial effects of performing tasks (which would allow an organism to execute its code faster and thus replicate faster) from evolved robustness to the harmful effect of mutations. To do so, he tried to disable mutations that improved an organism's replication rate (i.e., its fitness). He configured the system to pause every time a mutation occurred, and then measured the mutant's replication rate in an isolated test environment. If the mutant replicated faster than its parent, then the system eliminated the mutant; otherwise, the mutant would remain in the population. He thus expected that replication rates could no longer improve, thereby allowing him to study the effect of mutational robustness more directly. However, while replication rates at first remained constant, they later unexpectedly started again rising. After a period of surprise and confusion, Ofria discovered that he was not changing the inputs provided to the organisms in the isolated test environment. The organisms had evolved to recognize those inputs and halt their replication. Not only did they not reveal their improved replication rates, but they appeared to not replicate at all, in effect "playing dead" when presented with what amounted to a predator.

Ofria then took the logical step of altering the test environment to match the same random distribution of inputs as would be experienced in the normal (non-isolated) environment. While this patch improved the situation, it did not stop the digital organisms from continuing to improve their replication rates. Instead they made use of randomness to probabilistically perform the tasks that accelerated their replication. For example, if they did a task half of the time, they would have a 50% chance of slipping through the test environment; then, in the actual environment, half of the organisms would survive and subsequently replicate faster. In the end, Ofria eventually found a successful fix, by tracking organisms' replication rates along their lineage, and eliminating any organism (in real time) that would have otherwise out-replicated its ancestors.

3.1.5 Absurdly Thick Lenses, Impossible Superposition, and Geological Disarray

Optimization algorithms have often been applied to design lenses for optical systems (e.g., telescopes, cameras, microscopes). Two families of solutions that were identified as likely being optimal in an article using an optimization algorithm not based on evolution [107] were easily outperformed, by a factor of two, by a solution discovered though digital evolution by Gagné et al. [50]. However, the evolved solution, while respecting the formal specifications of the problem, was not realistic: One lens in the evolved system was over 20 meters thick.

In a similarly underconstrained problem, William Punch collaborated with physicists, applying digital evolution to find lower-energy configurations of carbon. The physicists had a well-vetted energy model for between-carbon forces, which supplied the fitness function for evolutionary search. The motivation was to find a novel low-energy buckyball-like structure. While the algorithm produced very low-energy results, the physicists were irritated because the algorithm had found a superposition of all the carbon atoms onto *the same point in space*. "Why did your genetic algorithm

violate the laws of physics?" they asked. "Why did your physics model not catch that edge condition?" was the team's response. The physicists patched the model to prevent superposition, and evolution was performed on the improved model. The result was qualitatively similar: great low-energy results that violated another physical law, revealing another edge case in the simulator. At that point, the physicists ceased the collaboration.

A final related example comes from an application of evolutionary algorithms to a problem in geophysics. Marc Schoenauer relates attempting to infer underground geological composition by analyzing echoes of controlled explosions [90]. The fitness function was a standard criterion used in geology, based on properties of how waves align. To the experimenters' delight, evolution produced geological layouts with very high scores. However, an expert examining the underground layouts selected by evolution declared that they "cannot be thought as a solution by anyone having even the smallest experience in seismic data" [90, p. 645], as they described chaotic and unnatural piles of polyhedral rocks.

These examples highlight how fitness functions often do not include implicit knowledge held by experts, thus allowing for solutions that experts consider so ridiculous, undesirable, or unexpected that they did not think to exclude or penalize such solutions when originally designing the fitness function. Although failing to provide the desired type of solution, the surprising and unacceptable results can catalyze thought and discussion that ultimately leads to more explicit understanding of problems.

3.2 Unintended Debugging

Another manifestation of digital evolution's creative freedom from human preconceptions about what form a solution *should* take is that search will often learn how to exploit bugs in simulations or hardware. One common approach in digital evolution is to start with a *simulation* of a physical problem, so that evolution can initially be run completely in software. The benefit is that physics simulations often run much faster than real time, thereby making more generations of evolution feasible, which can allow studies that would be infeasible in the physical world. However, physics simulations rarely mimic the real world exactly, meaning that subtle differences remain. As a result, edge cases, bugs, or minor flaws in the implemented laws of physics are sometimes amplified and exploited by evolution. The effect is the evolution of surprising solutions that achieve high fitness scores by physically unrealistic or otherwise undesirable means. Because the researcher is often unaware of the bugs, these exploits almost by definition surprise; often, they are also entertaining. While often frustrating to the experimenter, the benefit of such *unintended debugging* is to bring to light latent faults in simulation or hardware that would otherwise remain liabilities. In effect, evolution's exploits can enable efficient debugging of the simulations, and thus can actually advance the research program.

3.2.1 Evolving Virtual Creatures Reveal Imperfectly Simulated Physics

In further virtual-creature experiments [126, 127], Karl Sims' attempt to evolve swimming strategies resulted in new ways for evolution to cheat. The physics simulator first used a simple Euler method for numerical integration, which worked well for typical motion. However, with faster motion integration errors could *accumulate*, and some creatures learned to exploit that bug by quickly twitching small body parts. In effect, they could obtain "free energy," to propel them at unrealistic speeds through the water. Similarly, when evolving jumping abilities, the creatures found a bug in the code for collision detection and response. If the creatures hit themselves by contacting corners of two of their body parts together in a certain way, an error was triggered that popped them airborne like impossibly-strong grasshoppers. After such exploits were patched, the creatures eventually evolved many other interesting methods of locomotion—ones that did *not* violate the laws of physics.

Later extensions of Sims' work encountered similar difficulties, as in Cheney et al.'s work evolving the morphology of soft robots [15]. One feature of their simulator was that it applied a heuristic to estimate how coarsely it could simulate physics, to save on computation. The more cells that a

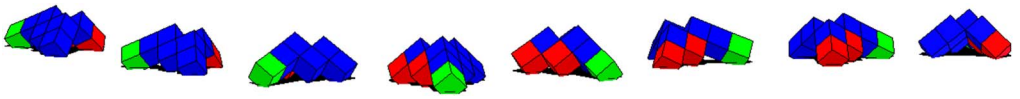

Figure 3. Vibrating robots. Evolved behavior is shown in frames, where time is shown progressing from left to right. A large time step enables the creatures to penetrate unrealistically through the ground plane, engaging the collision detection system to create a repelling force, resulting in vibrations that propel the organism across the ground.

creature was composed of, the less stable the simulator estimated the creature to be, and it would correspondingly simulate the world more granularly. In particular, the simulator *decreased* the time delta separating subsequent simulation steps as the number of cells in the creature *increased*.

Creatures evolved to exploit this heuristic, paring down their body to only a few cells, resulting in a large simulation time step. The large, less precise time step allowed the creature's bottommost cells to penetrate the ground between time steps without the collision being detected, which resulted in an upward force from the physics engine to correct the unnatural state. This corrective force provided "free" energy that enabled the creatures to vibrate and swiftly drift across the ground, producing a surprisingly effective form of locomotion. (See Figure 3.) To achieve more realistic results, the system was patched. Damping was increased when contacting the ground, the minimum creature size was raised, and the time delta calculation was adjusted to reduce ground penetration. Evolution thus helped to surface unanticipated edge cases that were poorly handled by the physics simulator and experimental design.

3.2.2 Ticktacktoe Memory Bomb

In a graduate-level AI class at UT Austin in 1997 taught by Risto Miikkulainen, the capstone project was a five-in-a-row ticktacktoe competition played on an infinitely large board. The students were free to choose any technique they wanted, and most people submitted typical search-based solutions. One of the entries, however, was a player based on the SANE neuroevolution approach for playing Othello [96, 98]. As in previous work, the network received a board representation as its input and indicated the desired move as its output. However, it had a clever mechanism for encoding its desired move that allowed for a broad range of coordinate values (by using units with an exponential activation function). A byproduct of this encoding was that it enabled the system to request nonexistent moves very, very far away in the ticktacktoe board. Evolution discovered that making such a move right away led to a lot of wins. The reason turned out to be that the other players dynamically expanded the board representation to include the location of the far-away move—and crashed because they ran out of memory, forfeiting the match.

3.2.3 Floating-Point Overflow Lands an Airplane

In 1997, Feldt applied digital evolution to simulations of mechanical systems to try to evolve mechanisms that safely, but rapidly, decelerate aircraft as they land on an aircraft carrier [38]. An incoming aircraft attaches to a cable, and the system applies pressure on two drums attached to the cable. The idea was to evolve the control software that would bring the aircraft to a smooth stop by dynamically adapting the pressure. Feldt was expecting evolution to take many generations, given the difficulty of the problem, but evolution almost immediately produced suspiciously *nearly perfect* solutions that were very efficiently braking the aircraft, even when simulating heavy bomber aircraft coming in to land.

Given the perceived problem difficulty, and that no bugs in the evolutionary algorithm could be found, the suspicion came to rest on the physics simulator. Indeed, evolution had discovered a loophole in the force calculation for when the aircraft's hook attaches to the braking cable. By overflowing the calculation, that is, exploiting the fact that numbers too large to store in memory roll over to zero, the resulting force was sometimes estimated to be zero. This, in turn, would lead to a perfect score, because of low mechanical stress on the aircraft, hook, cable, and pilot (because zero force means very little deceleration, implying no damaging "*g* force" on the pilot). In this way, evolution had discovered that creating enormous force would break the simulation, although clearly it was an exceedingly poor solution to the actual problem. Interestingly, insights from this experiment led to

theories about using evolution in software testing (to find bugs and explore unusual behavior) and engineering (to help refine knowledge about requirements) [38–40] that were later identified as important early works facilitating the field of "search-based software engineering" [56].

3.2.4 Why Walk Around the Wall When You Can Walk Over It?

The NeuroEvolving Robotic Operatives (NERO) video game applied digital evolution to enable non-player characters to evolve in real time while the game is being played [130]. While the polished version of the game that was released in 2005 portrays a world where order prevails, evolution's tendency to seek out and exploit loopholes led to some humorous and unrealistic behaviors during development. For example, players of NERO are encouraged to place walls around their evolving robots to help them learn to navigate around obstacles. However, evolution figured out how to do something that should have been impossible: The robotic operatives consistently evolved a special kind of "wiggle" that causes them to walk *up* the vertical walls, allowing them to ignore obstacles entirely, and undermining the intent of the game. The NERO team had to plug this loophole, which resulted from a little-known bug in the Torque game engine, after which the robots acquiesced in the more physically realistic policy of walking around walls to get to the other side.

3.2.5 Exploiting a Bug in the Atari Game Q*bert

The next anecdote focuses on an evolutionary algorithm applied to a particular Atari game called Q*bert. Atari games are a common benchmark in deep reinforcement learning [94, 95, 117, 132]. The challenge is to learn a policy that maps from raw pixels to actions at each time step, with the objective of maximizing the game score. Typically, this policy is represented by a deep convolutional neural network with many (often millions of) learned weight parameters. Relevant to this anecdote, researchers at OpenAI [117], Uber [132], and the University of Freiburg [17] have recently shown that evolutionary algorithms are a competitive approach to solving such games [94, 95].

In particular, the University of Freiburg team employed a simple version of a decades-old EA called evolution strategies [114]. Interestingly, it learned to exploit two bugs in the Atari game Q*bert. In the first case, which turned out to be a known bug, instead of completing the first level, the agent baits an enemy to jump off the game platform with it, and scores the points for killing the enemy; for some reason, the game engine does not count this suicide as a loss of life, and the agent is able to repeat this process indefinitely (until the game cap of 100,000 frames). This pattern is shown in Figure 4 (top) and in a video at https://goo.gl/2iZ5dJ.

In the second, more interesting and previously unknown, bug, the agent finds a sequence of actions that completes the first level, but, for an unknown reason, this does not lead to the game advancing to the next level; instead, all platforms start to blink and the agent is able to move around seemingly aimlessly, but constantly gaining a huge number of points. The game counter was never designed for such high scores and maxes out at 99,999. This exploit actually causes the game counter to roll over many times (until the frame limit is reached) and seemingly could continue to do so indefinitely; it improved the state-of-the-art high score from around 24,000 to nearly a million points. Surprisingly, even the original developers of the game Q*bert (albeit a different version of the game) were not aware of this bug, even after decades of continuous play [139]. This pattern is shown in Figure 4 (bottom) and in a video at https://goo.gl/ViHRj2.

3.2.6 Reenabling Disabled Appendages

In work by Ecarlat and colleagues [31], an EA called MAP-Elites [99] was applied to explore possible interactions between a robot arm and a small box on a table. The goal was to accumulate a wide variety of controllers, ones that would move the cube to as many different locations on the table as possible. In the normal experimental setup, MAP-Elites was able to move the cube onto the table, to grasp the cube, and even to launch it into a basket in front of the robot [31]. In a follow-up experiment the robot's gripper was crippled, preventing it from opening and closing. The natural expectation was that the robotic arm could then move the small box in only limited ways, that is, to push it

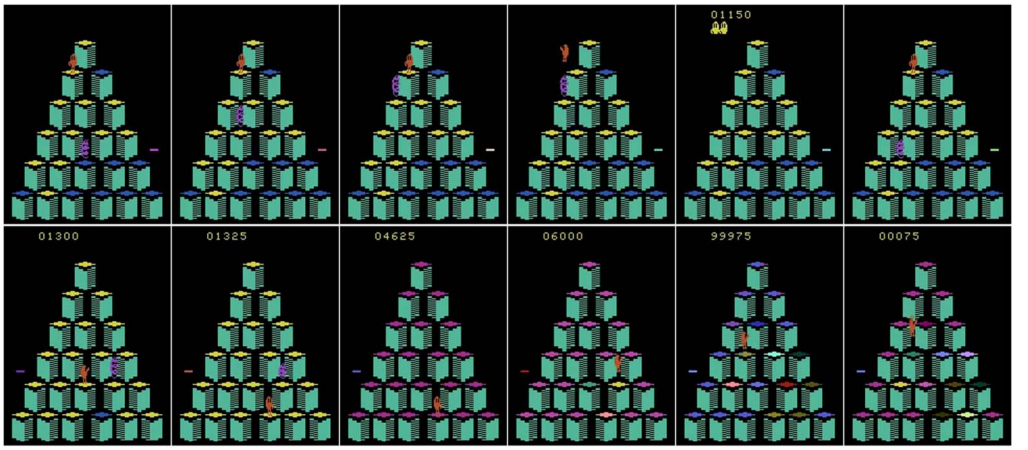

Figure 4. Top: The agent (orange blob in the upper left part of the screen) learns to commit suicide to kill its enemy (purple spring); because of the bug, the game does not count this as a loss of life. Bottom: The agent uses a bug in the game: After solving the first level in a specific sequence, the game does not advance to the next level, but instead the platforms start to blink and the agent gains a huge amount of points.

around clumsily, because it could no longer grasp the box. Surprisingly, MAP-Elites found ways to *hit* the box with the gripper in just the right way to force the gripper open so that it gripped the box firmly (Figure 5). Once holding the box, the gripper could move to a broad range of positions, undermining the experimenters' intent by in effect reenabling the disabled gripper (video: https://goo.gl/upTaiP).

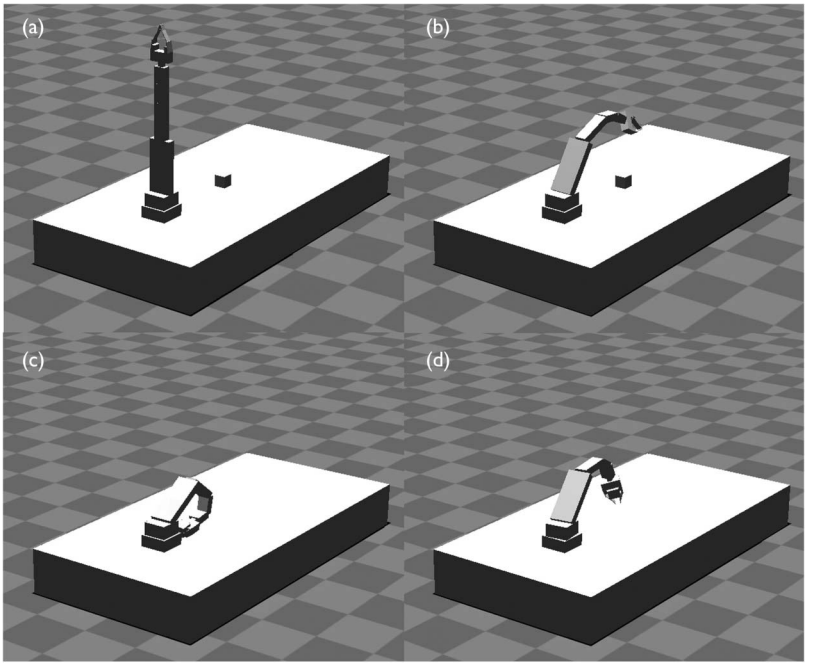

Figure 5. Snapshots of a forced-grasping trajectory. The robotic arm is in the initial position (a), with its gripper closed. The arm pushes the small box (b) towards the arm's base. The arm moves the gripper closer to its base (c), and executes a fast movement, sweeping across the table, forcing open its fingers, and grasping the small box. Finally, (d) the arm moves its gripper to a position holding the small box.

A similar result was noted by Moriarty and Miikkulainen in 1996 [97]. The researchers were evolving neural networks to control a robot arm called OSCAR-6 [138] in a newly modified version of the simulator. The goal was for the arm to reach a target point in midair; strangely, however, on new experiments evolution took five times as long as it had previously. Observing the behavior of the robot revealed a latent bug that arose when changing the simulator: The robot arm's main motor was completely disabled, meaning it could not directly turn towards targets that were far away from its initial configuration. However, the arm still managed to complete the task: It slowly turned its elbow away from the target, then quickly whipped it back—and the entire robot turned towards the target from inertia. The movement sequence was repeated until the arm reached the target position. It was not the solution that researchers were looking for, but one that revealed both a bug and an unexpected way to solve the problem even under exceptional constraints.

3.3 Exceeding Experimenter Expectations

Another class of surprise is when evolution produces *legitimate solutions* that go beyond experimenter expectations, rather than subverting experimenter intent or exploiting latent bugs.

3.3.1 Unexpected, Yet Valid, Solutions

Here we describe anecdotes in which evolution produces solutions that either were unconsidered or thought impossible, or were more elegant or sophisticated than expected.

Unexpected Odometer

In an experiment evolving digital organisms to successfully navigate a connected trail of nutrients, Grabowski et al. [55] encountered an unexpectedly elegant solution. Organisms were given the ability to sense whether there was nutrient underneath them and, if it was necessary to turn left or right, to stay on the nutrient trail; but their sensors could not detect if they were at the *end* of the trail. Organisms were rewarded for reaching more of the trail, and were penalized for stepping away from the trail. Because it was impossible to directly sense where the trail ended, the best expected solution was to correctly follow the trail *one step past* where it ended, which would incur a slight unavoidable fitness penalty. However, in one run of evolution, the system achieved a *perfect* fitness score—an analysis of the organism revealed that it had invented a step-counter, allowing it to stop precisely after a fixed number of steps, exactly at the trail's end.

Elbow Walking

Cully et al. in 2015 [21] presented an algorithm that enables damaged robots to successfully adapt to damage in under two minutes. The chosen robot had six legs, and evolution's task was to discover how to walk with broken legs or motors (Figure 6). To do so, ahead of the test, the researchers coupled digital evolution with a robot simulator, to first learn a wide diversity of walking strategies. Once damaged, the robot would then use the intuitions gained from simulated evolution to quickly learn from test trials in the real world, zeroing in on a strategy that remained viable given the robot's damage.

To evolve a large diversity of gaits, the team used the MAP-Elites evolutionary algorithm [99], which simultaneously searches for the fittest organism over every combination of chosen dimensions of variation (i.e., ways that phenotypes can vary). In this case, the six dimensions of variation were the fraction of time that the foot of each leg touched the ground, a way to encourage learning diverse locomotion strategies. Thus, MAP-Elites searched for the fastest-moving gait possible across every combination of how often each of the robot's six feet touched the ground. Naturally, the team thought it impossible for evolution to solve the case where all six feet touch the ground 0% of the time, but to their surprise, it did. Scratching their heads, they viewed the video: It showed a robot that flipped onto its back and successfully walked on its elbows, with its feet in the air (Figure 7). A video with examples of the different gaits MAP-Elites found, including

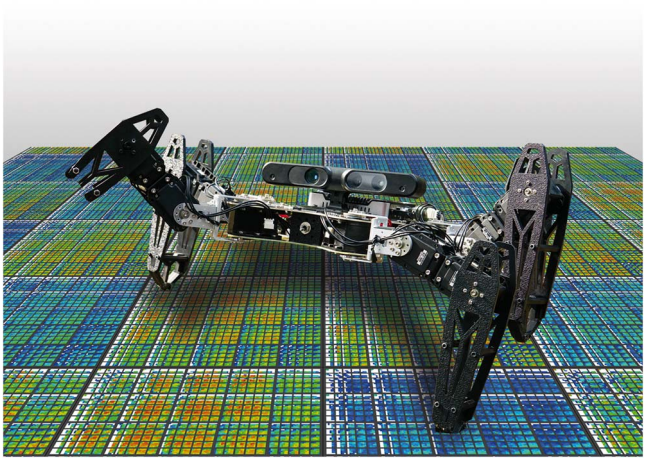

Figure 6. Six-legged robot. The robot uses the results of offline, simulated evolution to adapt quickly to a variety of damage conditions, such as a broken leg. Each point on the colored floor represents a different type of gait, that is, a gait that uses the robot's legs in different proportions. The assumption was that the cell in this map that required the robot to walk without using any of its legs would be impossible to fill. But, to the researchers' surprise, evolution found a way.

this elbow walking gait (which is shown at the end starting at 1:49), can be viewed here: https://goo.gl/9cwFtw.

Evolution of Unconventional Communication and Information Suppression

Mitri et al. [44, 93] applied digital evolution to groups of real and simulated robots, aiming to study the evolution of communication. The small two-wheeled robots were equipped with blue lights, which they could use as a simple channel of communication. Robots were rewarded for finding a food source while avoiding poison, both of which were represented by large red lights distinguishable only in close proximity. Over many generations of selection, all the robots evolved to find the food and avoid the poison, and under conditions that were expected to select for altruistic behavior, they also evolved to communicate the location of the food, for example by lighting up after they had reached it [44].

However, robots also solved the problem in surprising, unanticipated ways. In some cases, when robots adapted to understand blue as a signal of food, competing robots evolved to signal blue at poison instead, evoking parallels with dishonest signaling and aggressive mimicry in nature. In other experiments that involved conditions selecting for competition between robots, authors expected that the competitive robots simply would not communicate (i.e., not turn on their blue light), because broadcasting the location of the food would potentially help competitors. But rather than modifying how they signaled, some robots still lit up after finding food—but would then

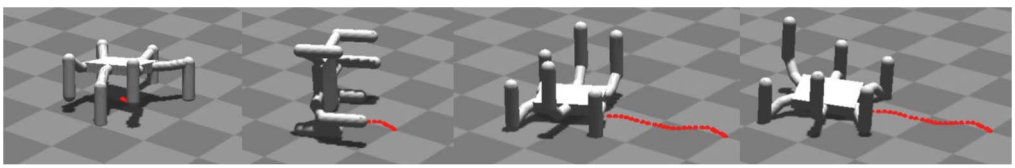

Figure 7. Elbow-walking gait. The simulated robot, tasked with walking fast without touching any of its feet to the ground, flips over and walks on its elbows. The red line shows the center of mass of the robot over time. Note that the robot fulfills the task because the first few tenths of a second of the simulation are ignored, to focus on the gait in its limit cycle, and not the robot's initial position.

literally hide the information from others by driving behind the food source (personal communication). Overall, a simple on-off light for communication revealed surprisingly rich evolutionary potential.

Impossibly Compact Solutions

To test a distributed computation platform called EC-star [106], Babak Hodjat implemented a multiplexer problem [67], wherein the objective is to learn how to selectively forward an input signal. Interestingly, the system had evolved solutions that involved too few rules to correctly perform the task. Thinking that evolution had discovered an exploit, the impossibly small solution was tested over all possible cases. The experimenters expected this test to reveal a bug in fitness calculation. Surprisingly, all cases were validated perfectly, leaving the experimenters confused. Careful examination of the code provided the solution: The system had exploited the logic engine's *rule evaluation order* to come up with a compressed solution. In other words, evolution opportunistically offloaded some of its work into those implicit conditions.

This offloading is similar to seminal work by Adrian Thompson in evolving real-world electronic circuits [136]. In Thompson's experiment, an EA evolved the connectivity of a reconfigurable field-programmable gate area (FPGA) chip, with the aim of producing circuits that could distinguish between a high-frequency and a lower-frequency square-wave signal. After 5,000 generations of evolution, a perfect solution was found that could discriminate between the waveforms. This was a hoped-for result, and not truly surprising in itself. However, upon investigation, the evolved circuits turned out to be extremely unconventional. The circuit had evolved to work only in the specific temperature conditions in the lab, and exploited manufacturing peculiarities of the particular FPGA chip used for evolution. Furthermore, when attempting to analyze the solution, Thompson disabled all circuit elements that were not part of the main powered circuit, assuming that disconnected elements would have no effect on behavior. However, he discovered that performance degraded after such pruning. Evolution had learned to leverage some type of subtle electromagnetic coupling, something a human designer would not have considered (or likely have known how to leverage).

The Fastest Route is Not Always a Straight Line

Richard Watson and Sevan Ficici evolved the behavior of physical robots. The simple robots they built had two wheels, two motors, and two light sensors [143, 144]. This type of robot is well known from Braitenberg's famous book *Vehicles* [12], which argued that connecting sensor inputs to motor outputs in a particular way results in simple light-following behavior. For example, when the right wheel is driven proportionally to how much light the left sensor detects and the left wheel is similarly driven by the right sensor, the robot will move towards the light. In Watson and Ficici's case the weights of connections between the input from the light sensors and the two wheel speeds were determined by evolution. The initial question was whether Braitenberg's original solution would actually be found [143, 144].

While the evolved robots successfully drove towards the light source, they often did so in unusual and unintuitive ways. Some *backed up* into the light while facing the dark, which was certainly an unexpected strategy. Others found the source by light-sensitive eccentric spinning, rather than the Braitenberg-style movement (Figure 8). It turns out that such spinning can easily be fine-tuned, by tightening or loosening the curvature, to produce effective light-seeking. After some analysis the authors discovered that the portion of the genetic search space that results in spinning is extremely large, while the classical Braitenberg solution requires delicate balance (e.g., the robot must execute subtle turns, changing heading from slightly clockwise to slightly anticlockwise: Figure 8) and thus occupies a relatively tiny part of genetic space. Further, despite its apparent inefficiency, spinning remained functional even when driven at higher speeds, unlike the classical solution, which could not adjust quickly enough when run at high motor speeds. Additionally, the spinning solution was more robust to hardware differences between the individual robots, and was less likely to get stuck

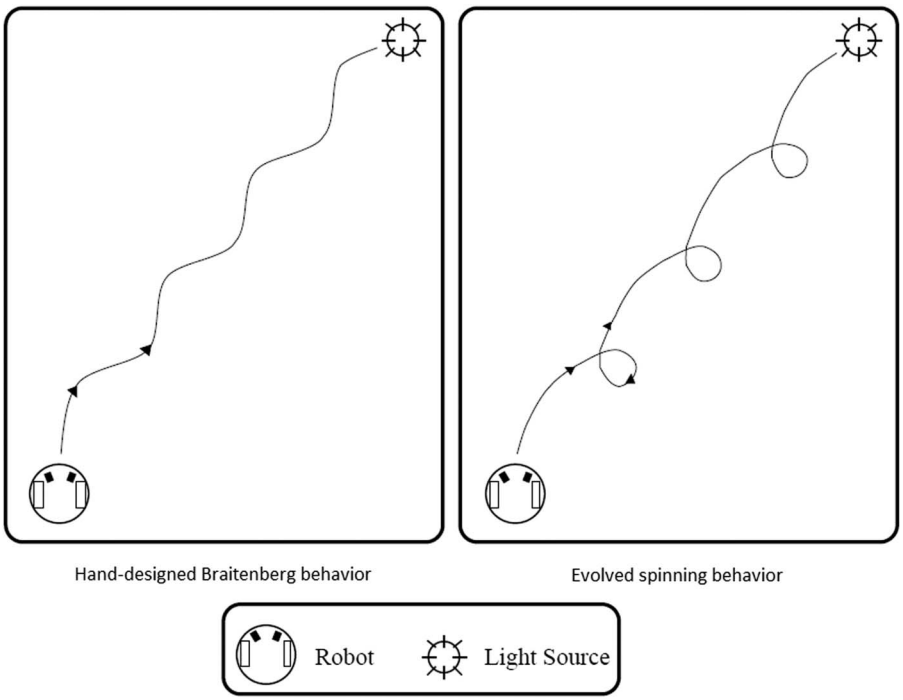

Figure 8. Light-seeking robot movement. The path of the hand-coded Braitenberg-style movement (left) and evolved spinning movement (right) when moving towards a light source.

in corners of the arena. Thus, evolution ultimately was able to discover a novel solution that was more robust than what had initially been expected.

Evolving a Car without Trying

At first glance it may seem that interactive evolution [133] is unlikely to surprise anyone, because of close and constant interactions with the user. And yet, in the case of Picbreeder [124], one such surprise was career-altering. Picbreeder is a platform, similar in form to the classic blind-watchmaker experiment [24], where the user can evolve new designs by choosing and recombining parents, with mutations, through successive generations. The images are encoded by mathematical functions, which are invisible to the user and may strongly constrain the direction and size of successive evolutionary steps. The surprise snuck up on one of the platform co-authors, Stanley, while evolving new images from one that resembled an alien face. As Stanley selected the parents, he suddenly noticed that the eyes of the face had descended and now looked like the wheels of a car. To his surprise, he was able to evolve a very different but visually interesting and familiar image of a car in short order from there. This quick and initially unintended transition between recognizable but dissimilar images was only the beginning of the story. The surprise inspired Stanley to conceive the novelty search algorithm [80], which searches without an explicit objective (just as Stanley found the car unintentionally), selecting instead the most different, novel outcomes at each evolution step. Later formalized by Lehman and Stanley together, the now-popular algorithm owes its existence to the unexpected evolution of a car from an alien face.

3.3.2 Impressive Evolved Art and Design

The anecdotes so far have focused on applications and insights related to computer science and engineering. However, there is also a long tradition of applying digital evolution to art and design. Here we detail two such examples. What is impressive and surprising about these stories is that the

outputs were not valued because they were decent attempts by computers to produce artistic creations, but were instead judged as valuable strictly on their own merits.

Evolving Tables and Tunes

In the 1990s digital evolution was often applied to optimization problems, but rarely to produce novel and functional designs. Peter Bentley was interested in this challenge, but initial feedback from professional designers was dismissive and discouraging: Such an approach is impossible, they said, because computers cannot invent new designs. They argued that even something as simple as a table could not be invented by evolution—how could it possibly find the right structure from a vast sea of possibilities, and how would you specify a meaningful fitness function?

This challenge led Bentley to create the *generic evolutionary design* system [8], which provided evolution with an expanding set of building blocks it could combine into complex configurations. Fitness functions were developed that rewarded separate aspects of a functional design, such as: Is the upper surface flat? Will it stand upright when supporting a mass on its upper surface? Is its mass light enough to be portable?

The task put to the generic evolutionary design system was to evolve a table. From random initial designs, there emerged multiple elegant designs, including a variety of different functional tables, such as the classic four-legged type, one with a small but heavy base, and one with a flat base and internal weight (the "washing machine principle") (Figure 9). One of the evolved tables was successfully built and has remained in functional use for nearly two decades.

In 1999 Bentley was approached by a group of musicians and developers who wanted to generate novel music through digital evolution. Dance music was popular at the time, so the team aimed to evolve novel dance tracks. They set different collections of number-one dance hits as targets, that is, an evolving track would be scored higher the more it resembled the targets. The evolved results, eight-bar music samples, were evaluated by a musician who selected the ones to be combined into an overall piece, which was then professionally produced according to the evolved music score. The results were surprisingly good: The evolved tracks incorporated complex drum rhythms with interesting accompanying melodies and bass lines. Using bands such as The Prodigy as targets, digital evolution was able to produce novel dance tracks with clear stylistic resemblance.

In 2000 the group formed a record label named J13 Records. A distribution contract was drawn up and signed with Universal Music, stipulating that the true source of the music should not be revealed (even to the distributors, because Universal Music's CEO believed that no one would want to buy computer-generated music). The companies produced several dance tracks together, some of which were then taken by other music producers and remixed. Some of the music was successful in dance clubs, with the clubgoers having no idea that key pieces of the tracks they were dancing to were the products of digital evolution.

An Art Museum Accepts and Displays Evolved Art Produced by Innovation Engines

The *innovation engine* [101] is an algorithm that combines three keys ideas: (1) produce new innovations (i.e., solutions) by elaborating upon already evolved ones, (2) simultaneously evolve the population toward many different objectives (instead of a single objective as in traditional digital evolution), and (3) harness state-of-the-art deep neural networks to evaluate how interesting a

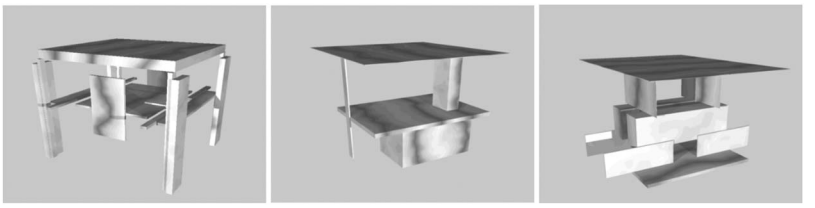

Figure 9. Three table designs evolved using the generic evolutionary design system [8].

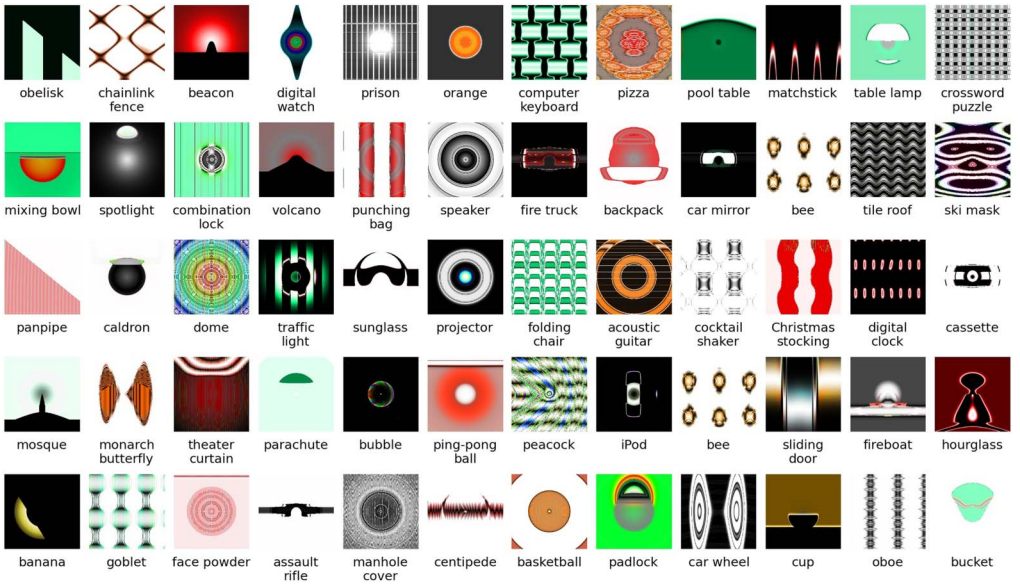

Figure 10. A selection of images evolved via an innovation engine. Underneath each image is the type of image that evolution was challenged to generate.

new solution is. The approach successfully produced a large diversity of interesting images, many of which are recognizable as familiar objects, both to deep neural networks and to human observers (Figure 10). Interestingly, the images span a variety of aesthetic styles, and bear resemblance to abstract "concept art" pieces (e.g., the two different images of prison cells, the beacon, and the folding chairs in Figure 10). Furthermore, the digital genomes of these algorithmically produced images are quantitatively similar to the elegant, compact genomes evolved by humans on the interactive evolution website Picbreeder [124].

To test whether the images generated by the innovation engine could pass for quality art, the authors submitted a selection of evolved images to a competitive art contest: the University of Wyoming's 40th Annual Juried Student Exhibition. Surprisingly, not only was the innovation engine piece among the 35.5% of submissions accepted, it was also among the 21.3% of submissions that were given an award! The piece was hung on the museum walls alongside human-made art, without visitors knowing it was evolved (Figure 11).

3.4 Convergence with Biology

Because it is inspired by biological evolution, digital evolution naturally shares with it the important abstract principles of selection, variation, and heritability. However, there is no guarantee that digital evolution will exhibit similar specific *behaviors* and *outcomes* to those found in nature, because the low-level details are so divergent: mutation rates, genome sizes, how genotypes map to phenotypes, population sizes, morphology, type of interactions, and environmental complexity. Interestingly, however, this subsection demonstrates how in practice there often exists surprising convergence between evolution in digital and biological media.

3.4.1 Evolution of Muscles and Bones

In further results from Cheney et al.'s virtual-creature system [15], evolution generated locomotion strategies unexpectedly convergent with those of biological creatures, examples of which are shown in Figure 12. The gait in the top figure is similar to the crawling of an inchworm, requiring evolution to discover from scratch the benefit of complementary (opposing) muscle groups, similar to such

Figure 11. University of Wyoming art show. A collection of images evolved with innovation engines on display at the University of Wyoming Art Museum. They have also been displayed in art exhibits in galleries, fairs, and conventions in several countries around the world.

muscle pairs in humans (e.g., biceps and triceps), and also to place them in a functional way. The gait in the bottom row highlights digital evolution's use of a stiff bonelike material to support thinner appendages, enabling them to be longer and skinnier without sacrificing their weight-bearing potential. The end product is a gait reminiscent of a horse's gallop.

3.4.2 Evolution of Parasitism

In 1990, Tom Ray developed his seminal artificial life system, Tierra [112], an early instance of evolution by natural selection in a digital medium. Organisms in Tierra consist of self-replicating machine code, somewhat like computer viruses. However, unlike computer viruses, organisms in Tierra live on virtual machines explicitly designed to enable evolution (e.g., the instruction set was designed to be fault-tolerant and evolvable). Tierra manages a population of replicating programs, killing off the oldest programs or those generating the most errors. Importantly, the operations (including copying) are faulty, meaning that replication necessarily produces mutations. Ray's hope was that Tierra would eventually create an interesting and alien tree of life in a computational universe,

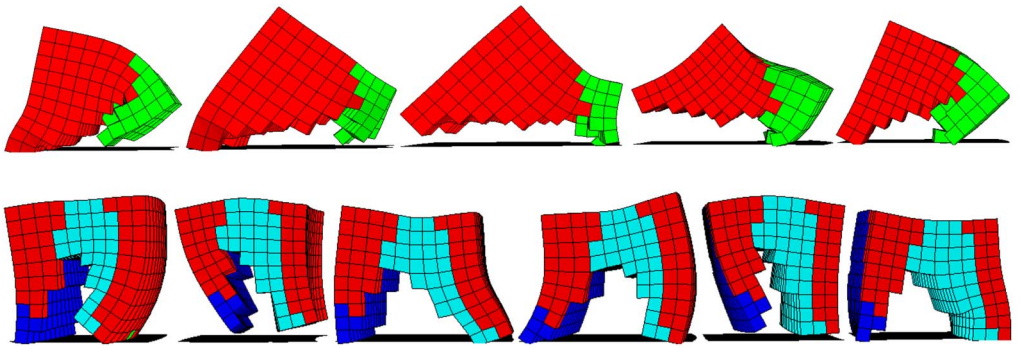

Figure 12. A stop-motion view of a small sample of the evolved gaits from Cheney et al. [15], which produced surprisingly effective and biologically reminiscent behaviors. Shown here are soft robots progressing from left to right across the panel. Colors correspond to voxel types (with red and green denoting oppositely contracting muscle groups, and dark and light blue representing stiff and soft support materials, respectively). In the top gait, notice how evolution creates distinct regions of each muscle. It employs these opposing muscle groups to create an inchworm-like behavior. In the bottom gait, the use of stiff (bonelike) support material allows evolution to create relatively long appendages and produce a horselike galloping behavior. Videos of various soft-robot gaits, including these two, can be found at https://youtu.be/z9ptOeByLA4?list=PL5278ezwmoxQODgYB0hWnC0-Ob09GZGe2.

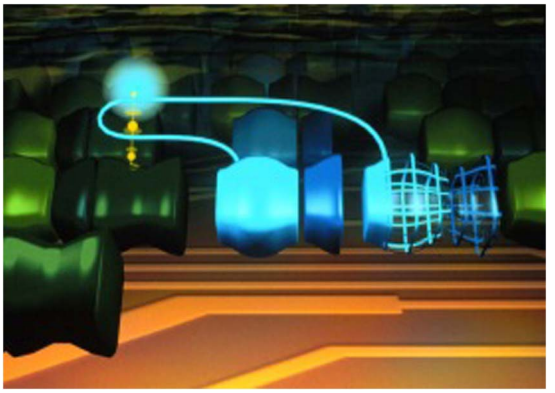

Figure 13. Parasites in Tierra. A self-replicator (green, left) has code that copies the genome from parent to offspring. The parasite (blue, center) lacks the genome replicating code, and executes that code in its neighbor, copying its genome into its offspring (blue shell, right). The blue sphere represents the parasite's CPU executing its neighbor's code. Image courtesy of Anti-Gravity Workshop.

but he expected to spend perhaps years tinkering before anything interesting would happen; surprisingly, Tierra produced complex ecologies the very first time it ran without crashing [112].

What emerged was a series of competing adaptations between replicating organisms within the computer, that is, an ongoing coevolutionary dynamic. The surprisingly large palette of emergent behaviors included parasitism (Figure 13), immunity to parasitism, circumvention of immunity, hyperparasitism (Figure 14), obligate sociality, cheaters exploiting social cooperation, and primitive forms of sexual recombination. All of these relied on digital templates, parts of code that provide robust addressing for JMP and CALL, the machine instructions that enable subroutines and control changes. By accessing templates not only in their own genomes, but in the genomes of others, Tierra organisms unexpectedly exploited this feature to facilitate a variety of ecological interactions.

When two individuals have complementary templates, interaction occurs. Organisms that evolved matching templates were able to execute code in neighboring organisms. They were selected for, because by outsourcing computation they reduced the size of their genome, which made replication less costly. Such organisms effectively engaged in *informational* parasitism. Evolving matching templates enabled exploitation, while non-complementary templates allowed individuals to escape

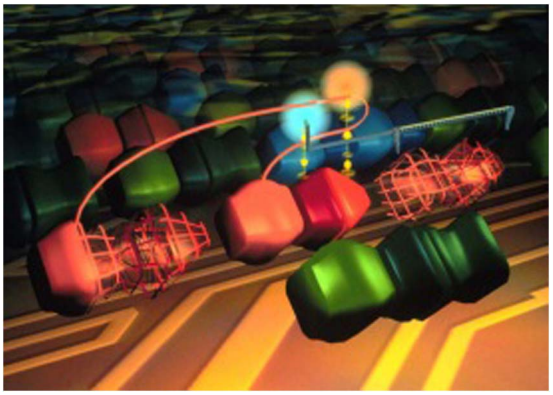

Figure 14. Hyperparasites in Tierra. A red hyperparasite (center), has captured a CPU (blue sphere) from a parasite, and is using it to replicate its genome into the shell on the right. The hyperparasite also has its own CPU (red sphere), which it is using to replicate also into the shell at the left. Image courtesy of Anti-Gravity Workshop.

exploitation. Ray termed the underlying process *bit-string races*, echoing the idea of evolutionary and ecological arms races in nature.

But the dynamics went further than bit-string races. Hyperparasites stole the CPUs of parasites, exhibiting *energy* parasitism. Social cooperators executed some of their own code, and some of their identical neighbor's code, to their mutual advantage. Social cheaters stole CPUs as they passed, exploiting the implicit trust between social creatures. Echoing natural evolution, a diversity of social and ecological interactions evolved in complex ways.

3.4.3 Digital Vestigial Organs

Virtual creatures evolved in the ERO system by Krcah [70] displayed a curious property: They sometimes contained small body parts whose function was not immediately obvious, yet they seemed to be carefully placed on the creature's body. It was not clear what purpose, if any, such "decorations" served. See Figure 15 for an example of a swimming creature with an ornamental "fin" on top of its back.

Analysis of the "fin" and its evolution demonstrated that its persistence was a consequence of a specific limitation of the evolutionary algorithm: Mutation was implemented in such a way that body parts were never entirely removed from any creature. The "fin" body part from Figure 15 had

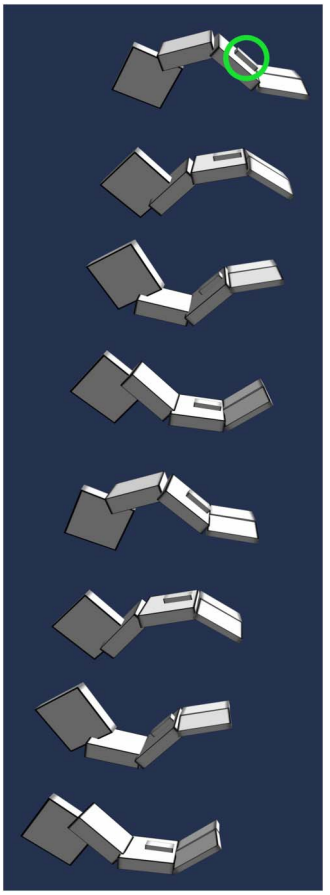

Figure 15. Swimming creature with an atrophied body part. A body part that was functionally important to an ancestor of the depicted creature became atrophied over evolutionary time. Mutations within this system did not allow deleting parts entirely; as a result, evolution shrunk the part and tuned its placement to minimize its deleterious effect on swimming speed. See https://youtu.be/JHOXzQeeUik?list=PL5278ezwmoxQODgYB0hWnC0-Ob09GZGe2 for full video.

origins as a big randomly generated block added very early in the creature's evolution. Because it could not be later removed when it started to interfere with the movements of the creature, it was instead quickly atrophied to the smallest allowed size and moved into the least obtrusive position, by a series of mutations.

3.4.4 Whole-Genome Duplication in Avida

Avida is a rich and versatile platform for digital evolution, one that has been used to study many fundamental evolutionary questions [1, 2, 16, 18, 20, 34, 52, 83, 85, 86, 92, 146]. During the submission process for an article on genetic architecture and sexual reproduction [92], reviewers pointed out that some data was unexpectedly bimodal: Evolution had produced two types of populations with distinct properties. After further investigation, the two types were found to be largely separable by their genome size. One group had lengths similar to the ancestral genome (50 instructions), while the other group had genomes about twice as long, suggestive of genome duplication events. Duplication mutations were known to be theoretically possible, but it was not obvious why or how such a sharp change in genome length had evolved. Interestingly, the Avida organisms had found an unanticipated (and unintended) mechanism to duplicate their genomes.

Experiments in Avida typically start from a hand-coded ancestral organism, effectively a short program that consists of a series of instructions capable of self-reproduction but nothing else. The reproduction mechanism executes a loop that iteratively copies the genome instruction by instruction. The loop terminates when an "if" instruction detects that the last instruction in the genome has been copied. The double-length organisms resulted from an unanticipated situation, which was triggered when organisms had an odd number of instructions in their genome, and a mutation then introduced a second "copy" instruction into the copy loop. Because the "if" condition was checked only after every two "copy" instructions, the copying process could continue past the last instruction in the genome, ultimately copying the whole genome again. In this way, through a particular detail of the Avida reproduction mechanism, digital organisms managed to duplicate their entire genomes, as sometimes also happens in biological evolution.

3.4.5 Evolving Complex Behavior May Involve Temporary Setbacks

In a pioneering study, Richard Lenski and colleagues used the Avida digital evolution platform to test some of Darwin's hypotheses about the evolution of complex features [86]. In Avida, digital organisms can perform a wide variety of computational functions, including copying themselves, by executing instructions in their genome. The researchers were interested in the general processes by which the evolutionary process produces complex features. The team specifically focused on whether and how Avidians might evolve to perform the most complex logical function in the environment—EQU—which requires comparing two 32-bit numbers and determining whether or not they are equal.

The experiment provided several surprises about the creative power of the evolutionary process. The EQU function evolved in about half the replicate experimental populations, but each instance was unpredictably different, using anywhere from 17 to 43 instructions. The most surprising outcome was that the pathway that evolution followed was not always an upward climb to greater fitness, nor even a path consisting of sideways, neutral steps. Instead, in several cases, mutations along the line of descent to EQU were deleterious, some significantly so. In two cases, mutations reduced fitness by half. Though highly deleterious themselves, these mutations produced a genetic state that allowed a subsequent beneficial mutation to complete a sequence that could preform the EQU function. This result sheds light on how complex traits can evolve by traversing rugged fitness landscapes that have fitness valleys that can be crossed to reach fitness peaks.

3.4.6 Drake's Rule

The Aevol digital evolution model, which belongs to the so-called *sequence-of-nucleotides* formalism [57], was originally developed by Carole Knibbe and Guillaume Beslon with the intent to study the evolution of modularity in gene order. However, even if some preliminary results on gene order

were promising, none of them turned out statistically viable after deeper investigation, seemingly indicating that the whole project was likely to fail. Then, one day in a corridor, Knibbe bumped into Laurent Duret, a renowned bioinformatician. Knibbe related her disappointing Ph.D. advancement, saying, "We have nothing interesting; the only clear signal is that genome size apparently scales with mutation rates—both the coding part and the noncoding part, but that's trivial, isn't it?" Laurent disagreed, "The noncoding part also? But that's a scoop!" It turned out that (i) without being designed to do so, the Aevol model had spontaneously reproduced *Drake's rule*, stating that the size of microbial genomes scales inversely with their mutation rate [30], and (ii) no model had predicted a scaling between the mutation rates and the noncoding size of a genome. Only the relation between the size of the *coding* region of the genome and the mutation rates was theoretically expected, as a result of the error threshold effect first identified by Eigen in his quasi-species model [32]. The effect on the noncoding region could be observed in Aevol because the model included chromosomal rearrangements in addition to point mutations [63]. This random encounter opened a new research direction that ultimately led to a more general mathematical model of genome evolution, showing that indirect selection for robustness to large duplications and deletions strongly bounds genome size [42].

3.4.7 Becoming Unbreakable Rather than Better

Many examples have described surprising ways that digital evolution optimizes its fitness function. However, early experiments aiming to evolve computer programs (a technique called genetic programming [68]) revealed another kind of surprise. As evolution progressed across generations, evolved genetic programs kept becoming larger and larger, a phenomenon called *bloat* [73, 87, 125], which eventually slowed the algorithm to a crawl because it took so long to run the huge evolved programs to test their fitness. Upon closer examination, the evolved programs were found, surprisingly, to be full of "junk code" that could be completely removed without changing the behavior of the program at all (see also the rich literature on neutrality in digital and biological evolution [51, 61]).

The mystery was resolved when researchers discovered that bloat can have an evolutionary advantage by buffering against the disruptive effects of mutating and mating genetic programs [3]. In early populations, creating a child program by mutating it, or replacing parts of one parent with those from another, is, on average, very damaging to fitness. Once in a while, of course, the change is beneficial, which fuels evolutionary adaptation. Additionally, some changes will by chance introduce new branches of neutral junk code that have no effect on fitness [104]. These mutants are no better at satisfying the fitness function, but their *offspring* are less likely to be harmed by mutation or mating; that is because organisms with a higher percentage of neutral code have a lower chance of random changes happening to critical code. The result is that bloated individuals are more likely to produce unchanged offspring, which grants them an indirect evolutionary advantage because the average fitness of unchanged offspring is higher than that of offspring with mutations that have affected functioning code. Over generations, programs thus grow larger and become increasingly robust to the effects of mutation and crossover.

The surprise is that evolution has its own "agenda" distinct from the programmer's. While the programmer hopes to create an algorithm that discovers the fittest structures, evolution may instead seek the structures whose fitnesses are least disturbed by reproduction. In some cases these two agendas may come into conflict, as they do here in two distinct ways. First, bloat causes genetic programming to consume increasing amounts of computation and memory over generations (due to rapidly growing evolved programs) [72]. The more fundamental conflict is that the high fraction of junk code in these huge evolved programs requires far more generations of evolution to adapt and improve—because evolution learns to turn off the very thing that ultimately fuels its ability to learn: random changes that alter behavior. A similar result has been observed in digital evolution, where when evolution was given the ability to evolve its own mutation rate, with accepted wisdom being that doing so would improve the results, evolution instead short-sightedly turned off mutations entirely [18]! Just as with bloat, this behavior is beneficial in the short run because mutations

tend to be harmful on average, but prevents adaptation over the long term, greatly hurting long-term performance. Interestingly, this phenomenon goes away when evolution is not forced to focus on short-term fitness improvement [81]. That natural evolution could favor robustness to change has been known to biologists since the 1940s [119, 140]. But the idea that natural selection could favor robustness over finding the fittest organisms was discovered only in the 1980s [123], and later, in digital evolution, as selection for "conservative code" [3], neutral evolution, of mutational robustness [10, 102], and "survival of the flattest" [123, 146].

3.4.8 Costly Genes Hiding from Natural Selection

Genes coding for cooperative behaviors—such as public good secretion or altruistic suicide—face very specific selection pressures that have interested researchers for decades. Their existence may seem counterintuitive, because they bring a benefit to the population at the expense of the individuals bearing them. The Aevol system has recently been used to study cooperation [47, 91], by giving individuals the ability to secrete a public good molecule, which benefits all digital organisms in the neighborhood. However, the public good molecule is costly for an individual to produce, digitally mirroring the challenges facing the evolution of cooperation in biological systems. In one experiment, the researchers studied the dynamics behind the loss of such costly cooperative genes. To evolve populations that would secrete the public good molecule, the researchers first lowered the cost of the molecule. After public good secretion had evolved, the researchers continued evolution with an increased cost to study if its production would cease. Interestingly, while most populations quickly lost all their secretion genes when the cost was increased in the second stage of evolution, some populations consistently did not, even when the second-stage experiments were repeated many times (starting from the same population).

The genetic analysis of the populations in which individuals consistently continued cooperating led to a surprising result. The secretion genes that survived the increase in cost were frequently overlapping with crucial metabolic genes, meaning that they were physically encoded using the same DNA base pairs as a metabolic gene, but using the opposite strand or another reading frame [48]. As a result, it was challenging for mutations to alter secretion behavior without also destroying metabolic genes. Costly secretion genes were effectively hiding behind directly beneficial metabolic ones. There is anecdotal evidence of similar mechanisms reducing the evolutionary potential toward cheating behavior in microbes [46, 103], but overlapping genes had never been studied in this context. To highlight how such results may often go unappreciated and unstudied, we note that when Frenoy, a master's student at the time, manually looked at the genomes that preserved secretion despite its cost (to try to understand how they were different), he had not heard of gene overlap and thought the result was likely an uninteresting artefact of the Aevol system. Only when presenting his results during a lab meeting did his colleagues point him toward the existence of overlapping genes in nature, and the fact that selection pressures on such genetic systems are not yet fully understood.

4 Discussion

A persistent misunderstanding is that digital evolution cannot meaningfully inform biological knowledge because "it is only a simulation." As a result, it is difficult to convince biologists, other scientists, and the general public that these systems, like biological evolution, are complex, creative, and surprising. Often such disagreements occur outside of published articles, in informal conversations and responses to reviewers. During such discussions, it is common for researchers in digital evolution to relate anecdotes like those included in this paper as evidence that such algorithms indeed unleash the creativity of the Darwinian process. However, such arguments lack teeth when rooted in anecdotes perpetuated through oral tradition. Thus one motivation for this article was to collect and validate the true stories from the original scientists and collect them for posterity.

Future work could move beyond collecting stories to directly study the prevalance of surprise among digital evolution practitioners. For example, a survey could be conducted to measure how

often researchers experience surprise (note that, like the anecdotes listed here, such surveys would depend on experimenters' self-reports of surprise). To go beyond self-reports is possible but likely requires expensive interventions, such as measuring physiological signals as experimenters work and correlating physiological signals with self-reports of surprise.

A separate motivation for collecting these anecdotes is to highlight pragmatic lessons for digital evolution practitioners. Foremost, a practitioner should be more skeptical of their ability to correctly specify robust fitness functions, and anticipate evolution iteratively, revealing such specification failures. This awareness is most important for safety-critical applications, and points to the need for careful supervision when attempting to apply digital evolution in real-world systems [79]. Undesirable outcomes can result from subtle interactions between fitness functions and experimental setups, suggesting that practitioners should adopt an adversarial mindset, looking for ways in which an agent could exploit seemingly innocuous design decisions. Another lesson relates to training in simulation, where evolution often exploits bugs to achieve high fitness: Practitioners should regularly visualize their simulated solutions to test whether the proposed solutions are valid and reasonable. A final lesson for practitioners (and beginning researchers) is to understand how the nature of academic publishing can obscure the actual (often messy) *process* of human-driven research that culminates in published articles. In other words, the phenomenon described in this article is common knowledge to experienced digital evolution researchers, but is nowhere to be found in the academic literature. We believe the main reasons are: (1) surprise is a subjective experience and many consider such subjectivity outside the typical bounds of science, (2) the surprising phenomena were often orthogonal (or a hindrance) to the research questions that researchers were pursuing, and (3) the convention for academic articles (deviations from which are often punished by reviewers) is to report the final successful experiments as if they were all that were performed; the effect is to distort the human experience of applying digital evolution in practice (e.g., to a new domain). As a result, practitioners should read articles with a critical outlook, and adjust expectations when adopting a published technique in a new domain.

Further, the ubiquity of surprising and creative outcomes in digital evolution has implications for other fields of artificial intelligence. For example, beyond their importance to digital evolution, the many examples of "misspecified fitness functions" in this article connect to the broader field of artificial intelligence safety: Many researchers therein are concerned with the potential for perverse outcomes from optimizing reward functions that appear sensible on their surface [4, 11, 37, 134], characterized in that community as problems such as avoiding negative side effects [4, 69], reward hacking (also known as wire-heading) [4, 36], or more generally AI alignment [134]. The list compiled here provides additional concrete examples of how difficult it is to anticipate the optimal behavior created and encouraged by a particular incentive scheme. Additionally, the narratives from practitioners highlight the iterative refinement of fitness functions often necessary to produce desired results instead of surprising, unintended behaviors. Interestingly, more-seasoned researchers develop better intuitions about how the creative process of evolution works, although even they sometimes still observe comical results from initial explorations in new simulations or experiments. Thus digital evolution may provide an interesting training ground for developing intuitions about incentives and optimization, to better ground theories about how to craft safer reward functions for AI agents.

Finally, there are interesting connections between surprising results in digital evolution and the products of directed evolution in biology, wherein selection in an experimenter-controlled evolutionary process is manipulated with the hope of improving or adapting proteins or nucleic acids for practical purposes [5, 108]. Echoing our "misspecified fitness functions" section, the first rule of directed evolution is "you get what you select for" [108, p. 992]. Selection for exactly the property you care about in directed evolution is often difficult and time-consuming, motivating cheaper heuristics that experimenters assume will lead to the desired outcome. However, the result is often something that meets the heuristic but deviates from the ideal outcome in surprising ways [120, 149]. In a final ironic twist, similar evolutionary arguments (applied to a higher level of biological organization) suggest that current incentive systems in science similarly produce surprising (and undesirable) byproducts [128].

5 Conclusion

Across a compendium of examples, we have reviewed many ways in which digital evolution produces surprising and creative solutions. We have also synthesized lessons from these examples, to aid practitioners and to communicate implications for biology and artificial intelligence more broadly. For every anecdote we included, there are likely others that have been already forgotten as researchers retire and new ones are created. The diversity and abundance of these examples suggest that surprise in digital evolution is common, rather than a rare exception, providing evidence that evolution—whether biological or computational—is inherently creative, and should routinely be expected to surprise, delight, and even outwit us.

Acknowledgments

We thank Elizabeth Ostrowski for a suggestion in the Digital Evolution lab at Michigan State University over a decade ago that led to the idea for this article, and for suggestions of Avida anecdotes to include. We also appreciate Tim Taylor, from whom we got the idea for crowd-sourcing the writing of this article by an open call for material from the community in which those whose submissions were accepted would become co-authors of the article; he successfully piloted that model for an article we were a part of [135], and we adopted it here. Finally, we also thank all of those who submitted anecdotes that we were not able to include.

Jeff Clune was supported by an NSF CAREER award (CAREER: 1453549).

References

1. Adami, C. (2006). Digital genetics: Unravelling the genetic basis of evolution. *Nature Reviews Genetics*, *7*(2), 109–118.
2. Adami, C., Ofria, C., & Collier, T. (2000). Evolution of biological complexity. *Proceedings of the National Academy of Sciences of the USA*, *97*(9), 4463.
3. Altenberg, L. (1994). The evolution of evolvability in genetic programming. In K. E. Kinnear (Ed.), *Advances in Genetic Programming*, Vol. 1 (pp. 47–74). Cambridge, MA: MIT Press.
4. Amodei, D., Olah, C., Steinhardt, J., Christiano, P., Schulman, J., & Mané, D. (2016). Concrete problems in AI safety. arXiv preprint arXiv:1606.06565.
5. Arnold, F. H. (1998). Design by directed evolution. *Accounts of Chemical Research*, *31*(3), 125–131.
6. Bedau, M. A. (1997). Weak emergence. *Noûs*, *31*, 375–399.
7. Bedau, M. A. (2003). Artificial life: Organization, adaptation and complexity from the bottom up. *Trends in Cognitive Sciences*, *7*(11), 505–512.
8. Bentley, P. J. (1996). *Generic evolutionary design of solid objects using a genetic algorithm*. Ph.D. thesis, The University of Huddersfield.
9. Bentley, P. J. (1999). Is evolution creative? In P. J. Bentley & D. W. Corne (Eds.), *Proceedings of the AISB*, Vol. 99 (pp. 28–34). Edinburgh: Society for the Study of Artificial Intelligence and Simulation of Behaviour.
10. Bornberg-Bauer, E., & Chan, H. S. (1999). Modeling evolutionary landscapes: Mutational stability, topology, and superfunnels in sequence space. *Proceedings of the National Academy of Sciences of the USA*, *96* (19), 10689–10694.
11. Bostrom, N. (2014). *Superintelligence: Paths, dangers, strategies*. Oxford, UK: Oxford University Press.
12. Braitenberg, V. (1986). *Vehicles: Experiments in synthetic psychology*. Cambridge, MA: MIT Press.
13. Bredeche, N., Haasdijk, E., & Eiben, A. (2009). On-line, on-board evolution of robot controllers. In P. Collet, N. Monmarché, P. Legrand, M. Schoenauer, & E. Lutton (Eds.), *International Conference on Artificial Evolution EA2009* (pp. 110–121). Berlin, Heidelberg: Springer-Verlag.
14. Campbell, D. T. (1979). Assessing the impact of planned social change. *Evaluation and Program Planning*, *2*(1), 67–90.

15. Cheney, N., MacCurdy, R., Clune, J., & Lipson, H. (2013). Unshackling evolution: Evolving soft robots with multiple materials and a powerful generative encoding. In C. Blum (Ed.), *Proceedings of the Genetic and Evolutionary Computation Conference* (pp. 167–174). New York: ACM.

16. Chow, S., Wilke, C., Ofria, C., Lenski, R., & Adami, C. (2004). Adaptive radiation from resource competition in digital organisms. *Science*, *305*(5680), 84.

17. Chrabaszcz, P., Loshchilov, I., & Hutter, F. (2018). Back to basics: Benchmarking canonical evolution strategies for playing Atari. arXiv preprint arXiv:1802.08842.

18. Clune, J., Misevic, D., Ofria, C., Lenski, R., Elena, S., & Sanjuán, R. (2008). Natural selection fails to optimize mutation rates for long-term adaptation on rugged fitness landscapes. *PLoS Computational Biology*, *4*(9), e1000187.

19. Clune, J., Mouret, J.-B., & Lipson, H. (2013). The evolutionary origins of modularity. *Proceedings of the Royal Society B*, *280*(1755), 20122863.

20. Covert, A. W., Lenski, R. E., Wilke, C. O., & Ofria, C. (2013). Experiments on the role of deleterious mutations as stepping stones in adaptive evolution. *Proceedings of the National Academy of Sciences of the USA*, *110*(34), E3171–E3178.

21. Cully, A., Clune, J., Tarapore, D., & Mouret, J.-B. (2015). Robots that can adapt like animals. *Nature*, *521*(7553), 503–507.

22. Darwin, C. (1859). *On the origin of species*. London: John Murray.

23. Dawkins, R. (1983). Universal Darwinism. In M. A. Bedau & C. E. Cleland (Eds.), *The nature of life: Classical and contemporary perspectives from philosophy and science*. Cambridge, UK: Cambridge University Press.

24. Dawkins, R. (1986). *The blind watchmaker: Why the evidence of evolution reveals a universe without design*. New York: W. W. Norton.

25. De Jong, K. A. (2006). *Evolutionary computation: A unified approach*. Cambridge, MA: MIT Press.

26. Dennett, D. C. (2002). The new replicators. In M. Pagel (Ed.), *Encyclopedia of Evolution* (pp. E83–E92). Oxford, UK: Oxford University Press.

27. Dennett, D. C. (2014). *Darwin's dangerous idea*. New York: Simon and Schuster.

28. Deutsch, D. (1998). *The fabric of reality*. London: Penguin Books.

29. Dobzhansky, T. (1974). Chance and creativity in evolution. In F. J. Ayala & T. Dobzhansky (Eds.), *Studies in the philosophy of biology* (pp. 307–338). Berlin: Springer.

30. Drake, J. W. (1991). A constant rate of spontaneous mutation in DNA-based microbes. *Proceedings of the National Academy of Sciences of the USA*, *88*(16), 7160–7164.

31. Ecarlat, P., Cully, A., Maestre, C., & Doncieux, S. (2015). Learning a high diversity of object manipulations through an evolutionary-based babbling. In *Proceedings of Learning Objects Affordances Workshop at IROS 2016* (pp. 1–2). Hambourg: IROS.

32. Eigen, M. (1971). Self-organization of matter and the evolution of biological macromolecules. *Die Naturwissenschaften*, *58*(10), 465–523.

33. Ellefsen, K. O., Mouret, J.-B., & Clune, J. (2015). Neural modularity helps organisms evolve to learn new skills without forgetting old skills. *PLoS Computational Biology*, *11*(4), e1004128.

34. Elsberry, W. R., Grabowski, L. M., Ofria, C., & Pennock, R. T. (2009). Cockroaches, drunkards, and climbers: Modeling the evolution of simple movement strategies using digital organisms. In *Proceedings of IEEE Symposium on Artificial Life* (pp. 92–99). Piscataway, NJ: IEEE.

35. Endler, J. A., & Greenwood, J. J. D. (1988). Frequency-dependent predation, crypsis and aposematic coloration. *Philosophical Transactions of the Royal Society of London. B, Biological Sciences*, *319*(1196), 505–523.

36. Everitt, T., & Hutter, M. (2016). Avoiding wireheading with value reinforcement learning. In B. Steunebrink, P. Wang, & B. Goertzel (Eds.), *Artificial general intelligence* (pp. 12–22). Berlin: Springer.

37. Everitt, T., Lea, G., & Hutter, M. (2018). AGI safety literature review. arXiv preprint arXiv:1805.01109.

38. Feldt, R. (1998). Generating diverse software versions with genetic programming: An experimental study. *IEE Proceedings—Software Engineering*, *145*(6), 228–236.

39. Feldt, R. (1999). Genetic programming as an explorative tool in early software development phases. In C. Ryan & J. Buckley (Eds.), *Proceedings of the 1st International Workshop on Soft Computing Applied to Software Engineering* (pp. 11–20). Limerick: Limerick University Press.

40. Feldt, R. (2002). *Biomimetic software engineering techniques for dependability*. Ph.D. thesis, Department of Computer Engineering, Chalmers University of Technology, Gothenburg, Sweden.

41. Fire, A., Xu, S., Montgomery, M. K., Kostas, S. A., Driver, S. E., & Mello, C. C. (1998). Potent and specific genetic interference by double-stranded RNA in *Caenorhabditis elegans*. *Nature*, *391*(6669), 806–811.

42. Fischer, S., Bernard, S., Beslon, G., & Knibbe, C. (2014). A model for genome size evolution. *Bulletin of Mathematical Biology*, *76*(9), 2249–2291.

43. Flake, G. W. (1998). *The computational beauty of nature: Computer explorations of fractals, chaos, complex systems, and adaptation*. Cambridge, MA: MIT Press.

44. Floreano, D., Mitri, S., Magnenat, S., & Keller, L. (2007). Evolutionary conditions for the emergence of communication in robots. *Current Biology*, *17*(6), 514–519.

45. Forrest, S., Nguyen, T., Weimer, W., & Le Goues, C. (2009). A genetic programming approach to automated software repair. In *Proceedings of the Genetic and Evolutionary Computation Conference* (pp. 947–954). New York: ACM.

46. Foster, K. R., Shaulsky, G., Strassmann, J. E., Queller, D. C., & Thompson, C. R. (2004). Pleiotropy as a mechanism to stabilize cooperation. *Nature*, *431*(7009), 693–696.

47. Frénoy, A., Taddei, F., & Misevic, D. (2012). Robustness and evolvability of cooperation. In C. Adami, D. M. Bryson, C. Ofria, & R. T. Pennock (Eds.), *Artificial Life 13 Conference Proceedings* (pp. 53–58). Cambridge, MA: MIT Press.

48. Frénoy, A., Taddei, F., & Misevic, D. (2013). Genetic architecture promotes the evolution and maintenance of cooperation. *PLoS Computational Biology*, *9*(11), e1003339.

49. Futuyma, D. J. (2013). Natural selection and adaptation. In J. B. Losos, D. A. Baum, D. J. Futuyma, H. E. Hoekstra, R. E. Lenski, A. J. Moore, C. L. Peichel, D. Schluter, & M. C. Whitlock (Eds.), *The Princeton Guide to Evolution* (pp. 279–301). Princeton, NJ: Princeton University Press.

50. Gagné, C., Beaulieu, J., Parizeau, M., & Thibault, S. (2008). Human-competitive lens system design with evolution strategies. *Applied Soft Computing*, *8*(4), 1439–1452.

51. Galván-López, E., Poli, R., Kattan, A., O'Neill, M., & Brabazon, A. (2011). Neutrality in evolutionary algorithms… What do we know? *Evolving Systems*, *2*(3), 145–163.

52. Goldsby, H. J., Dornhaus, A., Kerr, B., & Ofria, C. (2012). Task-switching costs promote the evolution of division of labor and shifts in individuality. *Proceedings of the National Academy of Sciences of the USA*, *109*(34), 13686–13691.

53. Goodhart, C. A. (1984). *Problems of monetary management: The UK experience*. Berlin: Springer.

54. Gould, S. J., & Vrba, E. S. (1982). Exaptation—a missing term in the science of form. *Paleobiology*, *8*(01), 4–15.

55. Grabowski, L. M., Bryson, D. M., Dyer, F. C., Pennock, R. T., & Ofria, C. (2013). A case study of the de novo evolution of a complex odometric behavior in digital organisms. *PLoS One*, *8*(4), e60466.

56. Harman, M., Mansouri, S. A., & Zhang, Y. (2012). Search-based software engineering: Trends, techniques and applications. *ACM Computing Surveys*, *45*(1), 11:1–11:61.

57. Hindré, T., Knibbe, C., Beslon, G., & Schneider, D. (2012). New insights into bacterial adaptation through in vivo and in silico experimental evolution. *Nature Reviews Microbiology*, *10*(5), 352–365.

58. Holland, J. H. (2000). *Emergence: From chaos to order*. Oxford: Oxford University Press.

59. Kashtan, N., & Alon, U. (2005). Spontaneous evolution of modularity and network motifs. *Proceedings of the National Academy of Sciences of the USA*, *102*(39), 13773–13778.

60. Kashtan, N., Noor, E., & Alon, U. (2007). Varying environments can speed up evolution. *Proceedings of the National Academy of Sciences of the USA*, *104*(34), 13711.

61. Kimura, M. (1983). *The neutral theory of molecular evolution*. Cambridge: Cambridge University Press.

62. Kirschner, M., & Gerhart, J. (1998). Evolvability. *Proceedings of the National Academy of Sciences of the USA*, *95*(15), 8420.

63. Knibbe, C., Coulon, A., Mazet, O., Fayard, J.-M., & Beslon, G. (2007). A long-term evolutionary pressure on the amount of non-coding DNA. *Molecular Biology and Evolution*, *24*(10), 2344–2353.

64. Korzybski, A. (1958). *Science and sanity: An introduction to non-Aristotelian systems and general semantics*. Oxford, UK: Institute of General Semantics.

65. Kounios, L., Clune, J., Kouvaris, K., Wagner, G. P., Pavlicev, M., Weinreich, D. M., & Watson, R. A. (2016). Resolving the paradox of evolvability with learning theory: How evolution learns to improve evolvability on rugged fitness landscapes. arXiv preprint arXiv:1612.05955.

66. Kouvaris, K., Clune, J., Kounios, L., Brede, M., & Watson, R. A. (2017). How evolution learns to generalise: Using the principles of learning theory to understand the evolution of developmental organisation. *PLoS Computational Biology*, *13*, e1005358.

67. Koza, J. R. (1990). A hierarchical approach to learning the Boolean multiplexer function. In G. J. Rawlins (Ed.), *Foundations of genetic algorithms*, Vol. 1 (pp. 171–192). Burlington, MA: Morgan Kaufmann.

68. Koza, J. R. (1992). *Genetic programming: On the programming of computers by means of natural selection*, Vol. 1. Cambridge, MA: MIT Press.

69. Krakovna, V., Orseau, L., Martic, M., & Legg, S. (2018). Measuring and avoiding side effects using relative reachability. arXiv preprint arXiv:1806.01186.

70. Krcah, P. (2008). Towards efficient evolutionary design of autonomous robots. In G. S. Hornby, L. Sekanina, & P. C. Haddow (Eds.), *Evolvable systems: From biology to hardware, Proceedings of the 8th International Conference ICES* (pp. 153–164). Berlin: Springer.

71. Kundrát, M. (2004). When did theropods become feathered?—evidence for pre-archaeopteryx feathery appendages. *Journal of Experimental Zoology Part B: Molecular and Developmental Evolution*, *302*(4), 355–364.

72. Langdon, W. B. (2000). Quadratic bloat in genetic programming. In L. D. Whitley, D. E. Goldberg, E. Cantú-Paz, & I. C. Parmee (Eds.), *Proceedings of the 2nd Annual Conference on Genetic and Evolutionary Computation* (pp. 451–458). San Francisco: Morgan Kaufmann.

73. Langdon, W. B., & Poli, R. (1998). Fitness causes bloat. In P. K. Chawdhry & R. K. Pant (Eds.), *Soft Computing in Engineering Design and Manufacturing* (pp. 13–22). Berlin: Springer.

74. Langton, C. G. (1990). Computation at the edge of chaos: phase transitions and emergent computation. *Physica D: Nonlinear Phenomena*, *42*(1–3), 12–37.

75. Langton, C. G. (1997). *Artificial life: An overview*. Cambridge, MA: MIT Press.

76. Last, K. S., Hobbs, L., Berge, J., Brierley, A. S., & Cottier, F. (2016). Moonlight drives ocean-scale mass vertical migration of zooplankton during the Arctic winter. *Current Biology*, *26*(2), 244–251.

77. Lau, N. C., & Bartel, D. P. (2003). Censors of the genome. *Scientific American*, *289*(2), 34–41.

78. Lefevre, T., Adamo, S. A., Biron, D. G., Misse, D., Hughes, D., & Thomas, F. (2009). Invasion of the body snatchers: The diversity and evolution of manipulative strategies in host–parasite interactions. *Advances in Parasitology*, *68*, 45–83.

79. Lehman, J. (2019). Evolutionary computation and AI safety: Research problems impeding routine and safe real-world application of evolution. arXiv preprint arXiv:1906.10189.

80. Lehman, J., & Stanley, K. O. (2011). Abandoning objectives: Evolution through the search for novelty alone. *Evolutionary Computation*, *19*(2), 189–223. URL http://www.mitpressjournals.org/doi/pdf/10.1162/EVCO_a_00025.

81. Lehman, J., & Stanley, K. O. (2011). Improving evolvability through novelty search and self-adaptation. In *Proceedings of the 2011 Congress on Evolutionary Computation (CEC 2011)* (pp. 2693–2700). Washington, DC: IEEE.

82. Lehman, J., & Stanley, K. O. (2014). Investigating biological assumptions through radical reimplementation. *Artificial Life*, *21*(1), 21–46.

83. Lenski, R., Ofria, C., Collier, T., & Adami, C. (1999). Genome complexity, robustness and genetic interactions in digital organisms. *Nature*, *400*(6745), 661–664.

84. Lenski, R. E. (1998). Get a life. *Science*, *280*(5365), 849–850.

85. Lenski, R. E., Barrick, J. E., & Ofria, C. (2006). Balancing robustness and evolvability. *PLoS Biology*, *4*(12), e428.

86. Lenski, R. E., Ofria, C., Pennock, R. T., & Adami, C. (2003). The evolutionary origin of complex features. *Nature*, *423*(6936), 139–144.

87. Luke, S., & Panait, L. (2006). A comparison of bloat control methods for genetic programming. *Evolutionary Computation*, *14*(3), 309–344.

88. Madigan, M. T. (2000). Bacterial habitats in extreme environments. In J. Seckbach (Ed.), *Journey to diverse microbial worlds* (pp. 61–72). Berlin: Springer.

89. Makarova, K. S., Aravind, L., Wolf, Y. I., Tatusov, R. L., Minton, K. W., Koonin, E. V., & Daly, M. J. (2001). Genome of the extremely radiation-resistant bacterium *Deinococcus radiodurans* viewed from the perspective of comparative genomics. *Microbiology and Molecular Biology Reviews*, *65*(1), 44–79.

90. Mansanne, F., Carrere, F., Ehinger, A., & Schoenauer, M. (1999). Evolutionary algorithms as fitness function debuggers. In Z. W. Rás & A. Skowron (Eds.), *International Symposium on Methodologies for Intelligent Systems* (pp. 639–647). Berlin, Heidelberg: Springer.

91. Misevic, D., Frénoy, A., Parsons, D. P., & Taddei, F. (2012). Effect of public good properties on the evolution of cooperation. In C. Adami, D. M. Bryson, C. Ofria, & R. T. Pennock (Eds.), *Artificial Life 13 Conference Proceedings* (pp. 218–225). Cambridge, MA: MIT Press.

92. Misevic, D., Ofria, C., & Lenski, R. E. (2006). Sexual reproduction reshapes the genetic architecture of digital organisms. *Proceedings of the Royal Society of London B: Biological Sciences*, *273*(1585), 457–464.

93. Mitri, S., Floreano, D., & Keller, L. (2009). The evolution of information suppression in communicating robots with conflicting interests. *Proceedings of the National Academy of Sciences of the USA*, *106*(37), 15786–15790.

94. Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., & Kavukcuoglu, K. (2016). Asynchronous methods for deep reinforcement learning. In M. F. Balcan & K. Q. Weinberger (Eds.), *Proceedings of the 33rd International Conference on Machine Learning* (pp. 1928–1937). New York: PMLR. URL http://proceedings.mlr.press/v48/mniha16.html.

95. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. (2015). Human-level control through deep reinforcement learning. *Nature*, *518*(7540), 529.

96. Moriarty, D. E., & Miikkulainen, R. (1995). Discovering complex Othello strategies through evolutionary neural networks. *Connection Science*, *7*(3), 195–209.

97. Moriarty, D. E., & Miikkulainen, R. (1996). Evolving obstacle avoidance behavior in a robot arm. In P. Maes, M. J. Mataric, J.-A. Meyer, J. Pollack, & S. W. Wilson (Eds.), *From animals to animats 4: Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior* (pp. 468–475). Cambridge, MA: MIT Press.

98. Moriarty, D. E., & Miikkulainen, R. (1997). Forming neural networks through efficient and adaptive co-evolution. *Evolutionary Computation*, *5*(4), 373–399.

99. Mouret, J.-B., & Clune, J. (2015). Illuminating search spaces by mapping elites. arXiv preprint arXiv:1504.04909 (pp. 1–15).

100. Nelson, B. (2013). University of Utah biologists surprised to discover "ultrafast recycling" at nerve synapses. https://kuer.org/post/u-u-biologists-surprised-discover-ultrafast-recycling-nerve-synapses. Accessed: 2016-07-28.

101. Nguyen, A., Yosinski, J., & Clune, J. (2016). Understanding innovation engines: Automated creativity and improved stochastic optimization via deep learning. *Evolutionary Computation*, *24*(3).

102. Nimwegen, E., Crutchfield, J. P., & Huynen, M. (1999). Neutral evolution of mutational robustness. *Proceedings of the National Academy of Sciences of the USA*, *96*(17), 9716–9720.

103. Nogueira, T., Rankin, D. J., Touchon, M., Taddei, F., Brown, S. P., & Rocha, E. P. (2009). Horizontal gene transfer of the secretome drives the evolution of bacterial cooperation and virulence. *Current Biology*, *19*(20), 1683–1691.

104. Nordin, P., Francone, F., & Banzhaf, W. (1996). Explicitly defined introns and destructive crossover in genetic programming. In P. J. Angeline & K. E. Kinnear, Jr. (Eds.), *Advances in genetic programming*, Vol. 2 (pp. 111–134). Cambridge, MA: MIT Press.

105. Ofria, C., & Wilke, C. O. (2004). Avida: A software platform for research in computational evolutionary biology. *Artificial Life*, *10*(2), 191–229.

106. O'Reilly, U.-M., Wagy, M., & Hodjat, B. (2013). Ec-star: A massive-scale, hub and spoke, distributed genetic programming system. In R. Riolo, E. Vladislavleva, M. Ritchie, & J. H. Moore (Eds.), *Genetic programming theory and practice X* (pp. 73–85). Berlin: Springer.

107. O'Shea, D. C. (1991). Monochromatic quartet: A search for the global optimum. In G. N. Lawrence (Ed.), *International Lens Design Conference* (pp. 548–554). Bellingham, WA: SPIE Press.

108. Peisajovich, S. G., & Tawfik, D. S. (2007). Protein engineers turned evolutionists. *Nature Methods*, *4*(12), 991–994.

109. Pennock, R. T. (2000). Can Darwinian mechanisms make novel discoveries?: Learning from discoveries made by evolving neural networks. *Foundations of Science*, *5*(2), 225–238.

110. Pennock, R. T. (2007). Models, simulations, instantiations, and evidence: The case of digital evolution. *Journal of Experimental & Theoretical Artificial Intelligence*, *19*(1), 29–42.

111. Press Release. (2016). In Arctic winter, marine creatures migrate by the light of the moon. https://www.eurekalert.org/pub_releases/2016-01/cp-iaw123015.php. Accessed: 2019-06-18.

112. Ray, T. (1992). J'ai joué á Dieu et créé la vie dans mon ordinateur. *Le Temps stratégique*, *47*, 68–81.

113. Ray, T. S. (1993). An evolutionary approach to synthetic biology: Zen and the art of creating life. *Artificial Life*, *1*(1–2), 179–209.

114. Rechenberg, I. (1973). *Evolutionsstrategie–Optimierung technisher Systeme nach Prinzipien der biologischen Evolution*. Stuttgart, Germany: Frommann-Holzboog.

115. Roese, N. J., & Vohs, K. D. (2012). Hindsight bias. *Perspectives on Psychological Science*, *7*(5), 411–426.

116. Runco, M. A., & Jaeger, G. J. (2012). The standard definition of creativity. *Creativity Research Journal*, *24*(1), 92–96.

117. Salimans, T., Ho, J., Chen, X., & Sutskever, I. (2017). Evolution strategies as a scalable alternative to reinforcement learning. arXiv preprint arXiv:1703.03864.

118. Schluter, D. (2000). *The ecology of adaptive radiation*. Oxford: Oxford University Press.

119. Schmalhausen, I. I. (1949). *Factors of evolution: The theory of stabilizing selection*. Chicago: University of Chicago Press. 1986 edition.

120. Schmidt-Dannert, C., & Arnold, F. H. (1999). Directed evolution of industrial enzymes. *Trends in Biotechnology*, *17*(4), 135–136.

121. Schöner, M. G., Schöner, C. R., Simon, R., Grafe, T. U., Puechmaille, S. J., Ji, L. L., & Kerth, G. (2015). Bats are acoustically attracted to mutualistic carnivorous plants. *Current Biology*, *25*(14), 1911–1916.

122. Schulte, E., Forrest, S., & Weimer, W. (2010). Automated program repair through the evolution of assembly code. In *Proceedings of the IEEE/ACM International Conference on Automated Software Engineering* (pp. 313–316). New York: ACM.

123. Schuster, P., & Swetina, J. (1988). Stationary mutant distributions and evolutionary optimization. *Bulletin of Mathematical Biology*, *50*(6), 635–660.

124. Secretan, J., Beato, N., D'Ambrosio, D. B., Rodriguez, A., Campbell, A., Folsom-Kovarik, J. T., & Stanley, K. O. (2011). Picbreeder: A case study in collaborative evolutionary exploration of design space. *Evolutionary Computation*, *19*(3), 373–403.

125. Silva, S., & Costa, E. (2009). Dynamic limits for bloat control in genetic programming and a review of past and current bloat theories. *Genetic Programming and Evolvable Machines*, *10*(2), 141–179.

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

127. Sims, K. (1994). Evolving virtual creatures. In *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques* (pp. 15–22). New York: ACM.

128. Smaldino, P. E., & McElreath, R. (2016). The natural selection of bad science. *Royal Society Open Science*, *3*(9), 160384.

129. Soltoggio, A., Bullinaria, J. A., Mattiussi, C., Durr, P., & Floreano, D. (2008). Evolutionary advantages of neuromodulated plasticity in dynamic, reward-based scenarios. In S. Bullock, J. Noble, R. Watson, & M. A. Bedau (Eds.), *Proceedings of the 11th International Conference on Artificial Life (Alife XI)* (pp. 569–576). Cambridge, MA: MIT Press.

130. Stanley, K. O., Bryant, B. D., & Miikkulainen, R. (2005). Real-time neuroevolution in the NERO video game. *IEEE Transactions on Evolutionary Computation*, *9*(6), 653–668.

131. Strahs, G. (1969). Biochemistry at 1000c: Explosive secretory discharge of bombardier beetles (Brachinus). *Science*, *165*(3888), 61–63.

132. Such, F. P., Madhavan, V., Conti, E., Lehman, J., Stanley, K. O., & Clune, J. (2017). Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning. arXiv preprint arXiv:1712.06567.

133. Takagi, H. (2001). Interactive evolutionary computation: Fusion of the capacities of EC optimization and human evaluation. *Proceedings of the IEEE*, *89*(9), 1275–1296.

134. Taylor, J., Yudkowsky, E., LaVictoire, P., & Critch, A. (2016). Alignment for advanced machine learning systems. Technical Report, Machine Intelligence Research Institute.

135. Taylor, T., Auerbach, J. E., Bongard, J., Clune, J., Hickinbotham, S., Ofria, C., Oka, M., Risi, S., Stanley, K. O., & Yosinski, J. (2016). WebAL comes of age: A review of the first 21 years of artificial life on the web. *Artificial Life*, *22*(3), 364–407.

136. Thompson, A. (1996). An evolved circuit, intrinsic in silicon, entwined with physics. In T. Higuchi, M. Iwata, & W. Liu (Eds.), *International Conference on Evolvable Systems* (pp. 390–405). Berlin, Heidelberg: Springer-Verlag.

137. Turing, A. M. (1936). On computable numbers, with an application to the entscheidungsproblem. *American Journal of Mathematics*, *58*, 345–363.

138. van der Smagt, P. (1994). Simderella: A robot simulator for neuro-controller design. *Neurocomputing*, *6*(2), 281–285.

139. Vincent, J. (2018). A video game-playing AI beat Q*bert in a way no one's ever seen before. *The Verge*. https://www.theverge.com/tldr/2018/2/28/17062338/ai-agent-atari-q-bert-cracked-bug-cheat.

140. Waddington, C. H. (1942). Canalization of development and the inheritance of acquired characters. *Nature*, *150*(3811), 563–565.

141. Wagner, G., Pavlicev, M., & Cheverud, J. M. (2007). The road to modularity. *Nature Reviews Genetics*, *8*(12), 921–931.

142. Watanabe, S., Rost, B. R., Camacho-Pérez, M., Davis, M. W., Söhl-Kielczynski, B., Rosenmund, C., & Jorgensen, E. M. (2013). Ultrafast endocytosis at mouse hippocampal synapses. *Nature*, *504*(7479), 242.

143. Watson, R. A., Ficici, S. G., & Pollack, J. B. (2002). Embodied evolution: Distributing an evolutionary algorithm in a population of robots. *Robotics and Autonomous Systems*, *39*(1), 1–18.

144. Watson, R. A., Ficiei, S., & Pollack, J. B. (1999). Embodied evolution: Embodying an evolutionary algorithm in a population of robots. In *Proceedings of the 1999 Congress on Evolutionary Computation* (pp. 335–342). Piscataway, NJ: IEEE.

145. Weimer, W. (2013). Advances in automated program repair and a call to arms. In G. Ruhe & Y. Zhang (Eds.), *Proceedings of Search Based Software Engineering—5th International Symposium* (pp. 1–3). Berlin, Heidelberg: Springer-Verlag.

146. Wilke, C., Wang, J., Ofria, C., Lenski, R., & Adami, C. (2011). Evolution of digital organisms at high mutation rates leads to survival of the flattest. *Nature*, *412*(6844), 331–333.

147. Wilson, E. O. (1999). *The diversity of life*. New York: W. W. Norton.

148. Yedid, G., & Bell, G. (2002). Macroevolution simulated with autonomously replicating computer programs. *Nature*, *420*(6917), 810.

149. Zhao, H., & Arnold, F. H. (1997). Combinatorial protein design: Strategies for screening protein libraries. *Current Opinion in Structural Biology*, *7*(4), 480–485.