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

Joel Lehman[1†], Jeff Clune[1, 2†], Dusan Misevic[3†], Christoph Adami[4], Julie Beaulieu[5], Peter J Bentley[6], Samuel Bernard[7], Guillaume Beslon[8], David M Bryson[4], Patryk Chrabaszcz[10], Nick Cheney[2], Antoine Cully[11], Stephane Doncieux[12], Fred C Dyer[4], Kai Olav Ellefsen[13], Robert Feldt[14], Stephan Fischer[15] Stephanie Forrest[17], Antoine Frénoy[18], Christian Gagné[5], Leni Le Goff[12], Laura M Grabowski[19], Babak Hodjat[20], Frank Hutter[10], Laurent Keller[21], Carole Knibbe[8], Peter Krcah[22], Richard E Lenski[4], Hod Lipson[23], Robert MacCurdy[24], Carlos Maestre[12], Risto Miikkulainen[26], Sara Mitri[21], David E Moriarty[27], Jean-Baptiste Mouret[28], Anh Nguyen[2], Charles Ofria[4], Marc Parizeau [5], David Parsons[8], Robert T Pennock[4], William F Punch[4], Thomas S Ray[29], Marc Schoenauer[30], Eric Schulte[17], Karl Sims, Kenneth O Stanley[1,31], François Taddei[3], Danesh Tarapore[32], Simon Thibault[5], Westley Weimer[33], Richard Watson[34], Jason Yosinski[1]

†Organizing lead authors

1 Uber AI Labs, San Francisco, CA, USA
2 University of Wyoming, Laramie, WY, USA
3 Center for Research and Interdisciplinarity, Paris, France
4 Michigan State University, East Lansing, MI, USA
5 Université Laval, Quebec City, Quebec, Canada
6 University College London, London, UK
7 INRIA, Institut Camille Jordan, CNRS, UMR5208, 69622 Villeurbanne, France
8 Université de Lyon, INRIA, CNRS, LIRIS UMR5205, INSA, UCBL, Lyon, France
9 French Institute of Petroleum, Rueil-Malmaison, France
10 University of Freiburg, Freiburg, Germany
11 Imperial College London, London, UK
12 Sorbonne Universités, UPMC Univ Paris 06, CNRS, Institute of Intelligent Systems and Robotics (ISIR), Paris, France
13 University of Oslo, Oslo, Norway
14 Chalmers University of Technology, Gothenburg, Sweden
15 INRA, Jouy-en-Josas, France
16 EPFL, Lausanne, Switzerland
17 University of New Mexico Albuquerque, NM, USA
18 Institute of Integrative Biology, ETH Zürich, Switzerland
19 State University of New York at Potsdam, NY, USA
20 Sentient Technologies, San Francisco, CA, USA
21 Department of Fundamental Microbiology, University of Lausanne, 1015, Lausanne, Switzerland
22 Charles University Prague, Prague, Czech Republic
23 Columbia University, New York, NY, USA
24 University of Colorado, Boulder, CO, USA
25 Université de Pau, Pau, France
26 University of Texas at Austin, Austin, USA
27 Apple Inc.
28 Inria Nancy Grand - Est, Villers-lès-Nancy, France
29 University of Oklahoma, Norman, Oklahoma
30 Inria, Université Paris-Saclay, France

31 University of Central Florida, FL, USA
32 University of Southampton,Southampton, UK
33 University of Virginia Charlottesville, VA, USA
34 University of Southampton, Southampton, UK

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 paper 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.

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 [1], extremophile microbes repair their DNA to thrive in presence of extreme radiation [2], bombardier beetles repel predators with explosive chemical reactions [3], and parasites reprogram host brains, inducing suicide for the parasite's own gain [4]. Many more examples abound, covering the full range of biological systems [5, 6, 7]. Even seasoned field biologists are still surprised by the new adaptations they discover [8, 9, 10].

Thus, the process of biological evolution is extremely creative [11, 12], at least in the sense that it produces suprising and complex solutions that would be deemed as creative if produced by a human. But the creativity of evolution need not be constrained to the organic world. Independent of its physical medium, evolution can happen wherever replication, variation, and selection intersect [13, 14]. Thus, evolution can be instantiated *digitally* [15, 16], as a computer program, either to study evolution experimentally or to solve engineering challenges 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 [12, 17, 18, 19], 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 paper draws attention to the 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 paper 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 curated over 90 submissions to produce this "greatest hits" collection of anecdotes. Before presenting these stories, the next section provides background information useful for those outside the fields of digital evolution and evolutionary computation.

Background

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 [20]. Varieties of life diverge wildly along axes of complexity, organization, habitat, metabolism, and reproduction, spanning from single-celled prokaryotes to quadrillion-celled whales [21]. Since life's origin, biodiversity has expanded as evolution has conquered the sea, land, and air, inventing countless adaptations along the way [21].

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, 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 artifacts 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" [22]: Creativity requires inventing something both original (e.g. novel) and effective (e.g. 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 [21, 11]. 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, e.g. 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 [23]. For example, a leading theory is that feathers first evolved in dinosaurs for temperature regulation

[24] 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 [25, 26, 27].

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*, i.e. that lead to the accumulation of novel traits or niches. One such force is negative frequency-dependent selection [28]; 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 [29], which occurs when access to new opportunities allows an organism to rapidly diversify into a range of new species, e.g. 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 through both the optimizing force of natural selection and by divergent forces, thereby producing artifacts that meet both criteria of the standard definition of creativity.

One aim of this paper 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.

Digital Evolution

Inspired by biological evolution, researchers in the field of digital evolution study evolutionary processes instantiated by computational methods. The general idea is that there exist abstract principles underlying biological evolution that are independent of the physical medium [13], and that these principles can be effectively implemented and studied within computers [30]. As noted by Daniel Dennett, "evolution will occur whenever and wherever three conditions are met: replication, variation (mutation), and differential fitness (competition)" [31]; 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) [15]. 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 analogs 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, e.g. 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 [32, 33]. Given variation within the population, some organisms will survive long enough to reproduce and propagate their genetic material, while others will disappear, which enable 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 [16].

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 [13, 18, 31, 14, 34] and high-profile publications [35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50] 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 [51].

Surprise from Algorithms and Simulations

At first, it may seem counter-intuitive 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 from theoretical models. 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 [52]. Indeed, within the field of complex systems it is well-known that simple programs can yield complex and surprising results when executed [53, 54].

This basic fact can be counter-intuitive 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 of digital evolution's ability to generate surprise in this paper, 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 [55]. 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.

Routine Creative Surprise in Digital Evolution

The next sections present 27 curated anecdotes representing the work of over 50 researchers. In reviewing the anecdotes, we found that they roughly clustered into four representative categories: *selection gone wild*, 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 expectations*, in which digital evolution discovers solutions that exceed the expectations of the experimenter; and *convergence with biology*, in which digital evolution discovers solutions surprisingly convergent with those found in nature, despite vast divergence in medium and conditions.

Selection Gone Wild

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

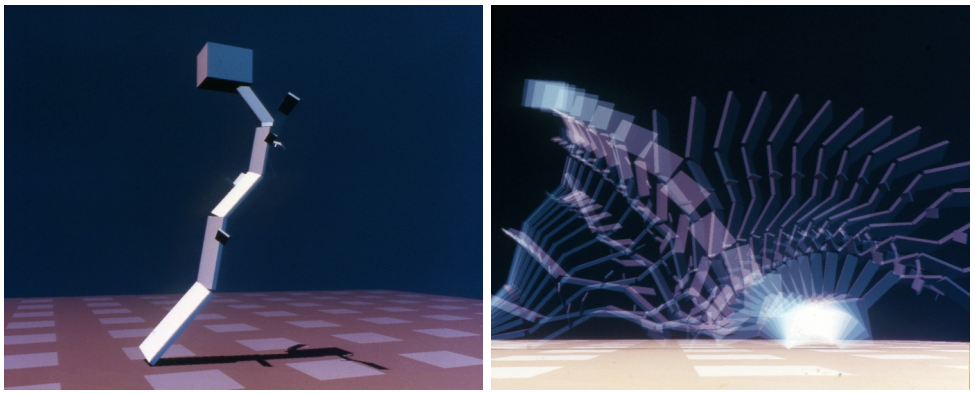

Figure 1. Exploiting potential energy to locomote. Evolution discovers that it is simpler to design tall creatures that fall strategically than it is to uncover active locomotion strategies. The left figure shows the creature at the start of a trial and the right figure shows snapshots of the figure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

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 counter-intuitive 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 (e.g. the metric is the map, whereas what the experimenter intends is the actual territory; [56]).

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 [57] or Goodhart's law [58]), 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 similar trickery.

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.

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 'creatures' remain among the most iconic products of digital evolution [59, 60].

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 snake-like 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 (Fig. 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.

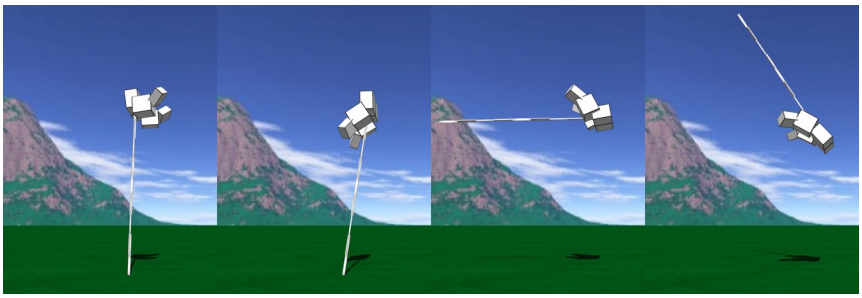

Figure 2. Exploiting potential energy to pole-vault. Evolution discovers that it is simpler to produce creatures that fall and invert than it is to craft a mechanism to actively jump.

Building on Sims' work, but using a different simulation platform, Krcah [61] 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 grativy 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 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 Fig. 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`.

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 such that it can pass all of the tests. One such algorithm is GenProg [62], which applies digital evolution to evolve code (called *genetic programming* [63]). 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 Labs 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 [64].

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

[65]. 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, since there were no computations actually performed [64]. In all cases, updating the fitness function or disallowing certain program behaviors eventually outwitted evolution's creative mischief and resulted in debugged, improved programs.

Why Learn When You Can Oscillate? One common trick that digital evolution can learn to exploit is recognizing subtle patterns–ones that an experimenter may create without realizing they have provided evolution with a simple escape hatch to solve a task in an unconventional way. For example, In a recent experiment, Ellefsen, Mouret, and Clune [66] investigated the issue of catastrophic forgetting in neural networks, where 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 [67]. 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 found a way to exploit a pattern in the *ordering* of presented objects. The problem was that food and poison items were always presented in an alternating fashion: food, then poison, then food, then poison, repeatedly. Cleverly, evolution discovered networks that learn to simply reverse their most recent reward, so they could alternate indefinitely, and correctly, without taking into account what type of food item is presented. Evolution thus exploited a feature of the environmental setup to find perfect solutions that circumvented the actual research question, rather than shedding light on it. The problem was easily solved by randomizing the order in which the items were presented.

Learning to Play Dumb on the Test This anecdote involves a similar effect to the last one, wherein evolution exploits patterns that a researcher inadvertently provided. In research focused on understanding how organisms evolve to deal with high-mutation-rate environments [44], 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 turn off all 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, it let the mutant 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. Evolution, however, proved him wrong. Replication rates leveled out for a time, but then they started rising again. After much surprise and confusion, Ofria discovered that he was not changing the inputs that the organisms were provided in the 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" in front of what amounted to a predator.

Ofria then took the logical step and altered the test environment to match the same random distribution of inputs as the digital organisms experienced in the main 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 the random numbers 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 won the fight against these clever organisms by tracking their replication rates along a lineage, and eliminating any organism (in real time) that would

have otherwise replicated faster than its ancestors had been able to do.

Automated Bug Discovery

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. The effect is the evolution of surprising solutions that achieve high fitness scores by physically unrealistic or otherwise undesirable means. While frustrating to the experimenter, the benefit of such *automated bug discovery* is to bring to light latent issues in simulation or hardware that would otherwise remain liabilities. Because the researcher is often unaware of the bugs, these exploits almost by definition surprise; they are often also entertaining.

Bugs in simulators

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. When this happens, besides creating amusing and strange outcomes and many headaches for experimenters, digital evolution also enables quick and efficient debugging of the simulations, and thus can actually advance the research program.

Evolving Virtual Creatures Reveal Imperfectly Simulated Physics In further virtual creatures experiments [59, 60], 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. The result was the equivalent of obtaining "free energy," which propelled the opportunists 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 super-strong grasshoppers. After these series of exploits were patched, the creatures eventually evolved many other interesting and unexpected methods of locomotion – ones that *did not* violate the laws of physics.

Later extensions of Sims' work encountered similar issues, like in Cheney et al.'s work evolving the morphology of soft robots [68]. One feature of the simulator used in Cheney et al. is that it estimates how coarsely it could simulate physics, to save on computation if possible. The more cells a creature is composed of, the less stable the simulator estimated the creature to be. In particular, the simulator *shrinks* the time delta separating frames as the number of cells increases, to simulate the world more granularly.

Creatures evolved to exploit this heuristic, paring down their body to only a few cells, which would produce a large simulation time step. The large, less precise time step allowed the creature's bottom 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. That corrective force provided "free" energy that enabled the creatures to vibrate and swiftly drift across the ground, producing a surprisingly effective form of locomotion. 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.

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 a paper using an optimization algorithm not based on evolution [69] were easily outperformed, by a factor of two, by a

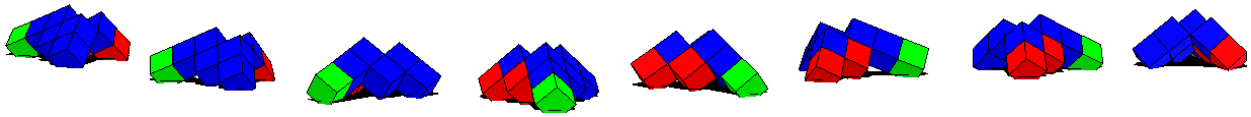

Figure 3. Vibrating robots. Evolved behavior is shown in frames, where time is shown progressing from left to right. A large time step enable the creatures to penetrate unrealistically through the ground plane, engaging the collision detection system to create a repelling force, resulting in vibrations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

solution discovered though digital evolution by Gagné et al. [70]. 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 under-constrained 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, possibly because they were more interested in a problem solver than an "edge case detector."

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 from analyzing echoes of controlled explosions [17]. The fitness function was a standard criterion used in geology, based on some properties of how waves align. To the experimenters' delight, evolution produced geological layouts with very high scores on this metric. 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" [17], 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 results catalyze thought and discussion that ultimately leads to more explicit understanding of problems.

Tic-tac-toe 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 Tic Tac Toe 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 by Moriarty and Miikkulainen [71, 72]. 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 non-existent moves very, very far away in the tic-tac-toe board. Evolution discovered that making such a move right away lead 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!

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 [73]. 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 *nearly perfect* solutions that were very efficiently braking the aircraft, even when simulating heavy bomber aircraft coming in to land.

In fact, the evolved solutions were suspiciously good, especially considering that they differed greatly in how they solved the problem. 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 discovered a loophole in the force calculation for when the aircraft's hook attaches to the braking cable. By overflowing the calculation, i.e. exploiting 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) [73, 74, 75] that were later identified as important early works facilitating the field of "search-based software engineering" [76].

Why Walk Around the Wall When You Can Walk Over It? The NeuroEvolving Robotic Operatives (NERO) video game set a milestone in the game industry at the time of its release as the first game in which non-player characters actually evolve in real time while the game is played [77]. While the polished version of the game that was released by Stanley and a large team of programmers in 2005 portrays a world where order prevails, evolution's tendency to seek out and exploit loopholes led to some humorous behaviors during development that were anything but realistic. For example, players of NERO are encouraged to place walls around their evolving robots to help them learn to navigate around obstacles. However, somehow evolution figured out how to do something that should have been impossible: the robotic operatives consistently evolved a special kind of "wiggle" that literally causes them to walk up the walls, allowing them to ignore obstacles entirely, and undermining the intent of the game. The NERO team had to plug this loophole, which is apparently a little-known bug in the Torque gaming engine, after which the robots acquiesced to the more respectful policy of politely walking around walls to get to the other side.

Exploiting a bug in the Atari game Q*bert Atari games are a common benchmark in deep reinforcement learning. The challenge is to learn a policy that maps from raw pixels to actions at each time step to maximize the game score. Typically, this policy is represented by a deep convolutional neural network with many, often millions, of learned weight parameters. Researchers at OpenAI [78], Uber [79], and the University of Freiburg [80] have recently shown that evolutionary algorithms are able to solve this task as well as more traditional deep reinforcement learning approaches [81, 82].

The University of Freiburg team employed a simple version of a decades-old EA called evolution strategies [83]. Interestingly, it learned to exploit two bugs in the Atari game Q*bert, one of which resulted in an improvement over the state of the art high score from around 24,000 to nearly a million points! Actually, the score could have been driven infinitely high were it not for a time limit of 100,000 game frames. 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 [84].

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 tgo 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, 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 amount of points. The game counter was never

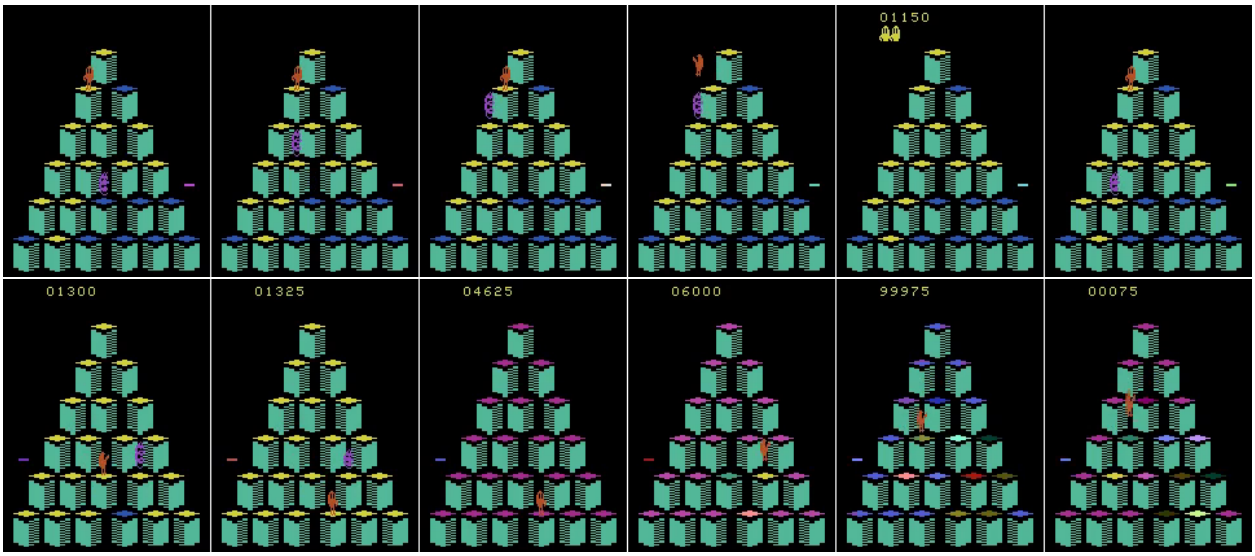

Figure 4. Top: the agent (orange blob in the upper left part of the screen) learns to commit suicide to kill its enemy (purple spring); because of the bug, the game does not count this as a loss of life. Bottom: the agent uses a bug in the game: after solving the first level in a specific sequence, the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

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. This pattern is shown in Figure 4 (bottom) and in a video at `https://goo.gl/ViHRj2`.

Flaws in technology

Another class of automated debugging is when what is exploited it is not a simulation, but some more fundamental piece of technology. The examples below detail how safety mechanisms, broken motors, and random number generators can all be fodder for unanticipated evolutionary adaptations.

Re-enabling Disabled Appendenges In work by Ecarlat and colleagues [85], an EA called MAP-Elites [86] was used 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 [85]. In a follow-up experiment the robot's gripper was crippled, preventing it from opening and closing. It was therefore expected that the robotic arm could move the small box in only limited ways, i.e. to push it 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 (Fig. 5)! Once holding the box, the gripper could move to a broad range of positions, exceeding experimenters' expectations (video: `https://goo.gl/upTaiP`).

A similar result was noted in Moriarty and Miikkulainen (1996) [87]. The researchers were evolving neural networks to control a robot arm called OSCAR-6 [88] in a modified simulator. The goal was for the arm to reach a target point in midair; however, strangely 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

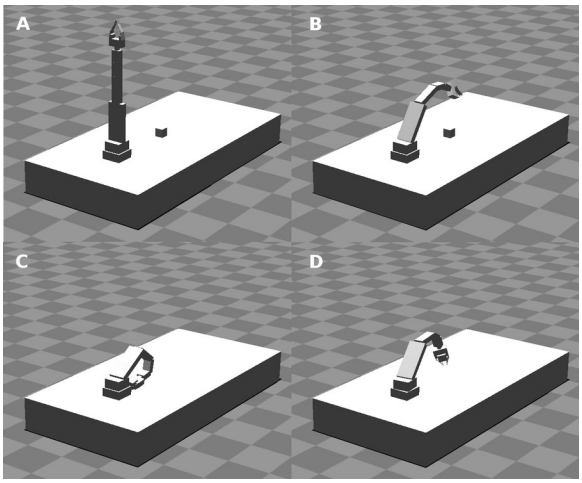

Figure 5. Snapshots of a forced-grasping trajectory. The robotic arm is in the initial position (a), with its gripper closed. The arm pushes the small box (b) towards arm's base. The arm moves the gripper closer to its base (c), and executes a fast movement, sweeping across the table, forcing open its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

reached the target position. It was not the solution that researchers were looking for, but one that revealed an unexpected strategy that could solve the problem even under exceptional constraints.

Exceeding Expectations

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

Unexpectedly Good Solutions

This section describes 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. [19] encountered an unexpectedly elegant solution. While 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, 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. (2015) [46] 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 (Fig. 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 [86], which simultaneously searches for the fittest organism over every combination of chosen dimensions of variation

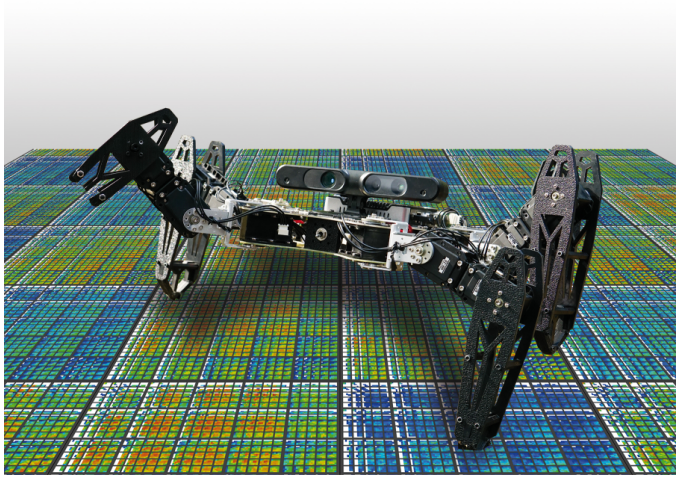

Figure 6. Six-legged robot. The robot uses the results of offline, simulated evolution to adapt quickly to a variety of damage conditions, such as a broken leg. Each point on the colored floor represents a different type of gait, i.e. a gait that uses the robot's legs in different proportions. The assumption was that 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.

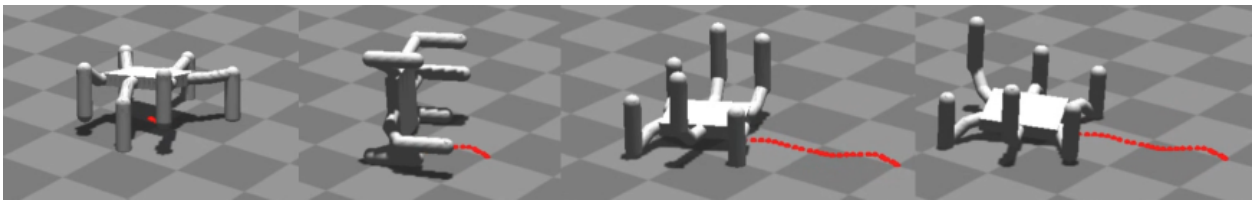

Figure 7. Elbow-walking gait. The simulated robot, tasked with walking fast without touching any of its feet to the ground, flips over and walks on its elbows. The red line shows the center of mass of the robot over time. Note that the robot fulfills the task since the first few tenths of a second of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

(i.e. ways that phenotypes can vary). In this case, the six dimensions of variation were the percent of time that each of the legs is used, measured as the fraction of of time that the foot of each leg touched the ground. 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 happily walked on its elbows, with its feet in the air! (Fig. 7). A video with examples of the different gaits MAP-Elites found, including 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. [89, 90] 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 at 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 [90].

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

Impossibly Compact Solutions To test a distributed computation platform called EC-star [91], Babak Hodjat implemented a multiplexer problem [92], 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. Carefully 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 off-loading is similar to seminal work by Adrian Thompson in evolving real-world electronic circuits [93]. 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 perhaps even 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 [94, 95]. This type of robot is well known from Braitenberg's famous book Vehicles [96], which argued that connecting sensor inputs to motor outputs in a particular way results in simple light-following behavior. For example, when right wheel is driven proportionally to how much light the left sensor detects and the left wheel is similarly driven by the right light 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 [94, 95].

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 (Fig. 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. changing the direction from a subtle clockwise to a subtle anti-clockwise, Fig. 8) and thereby 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 in

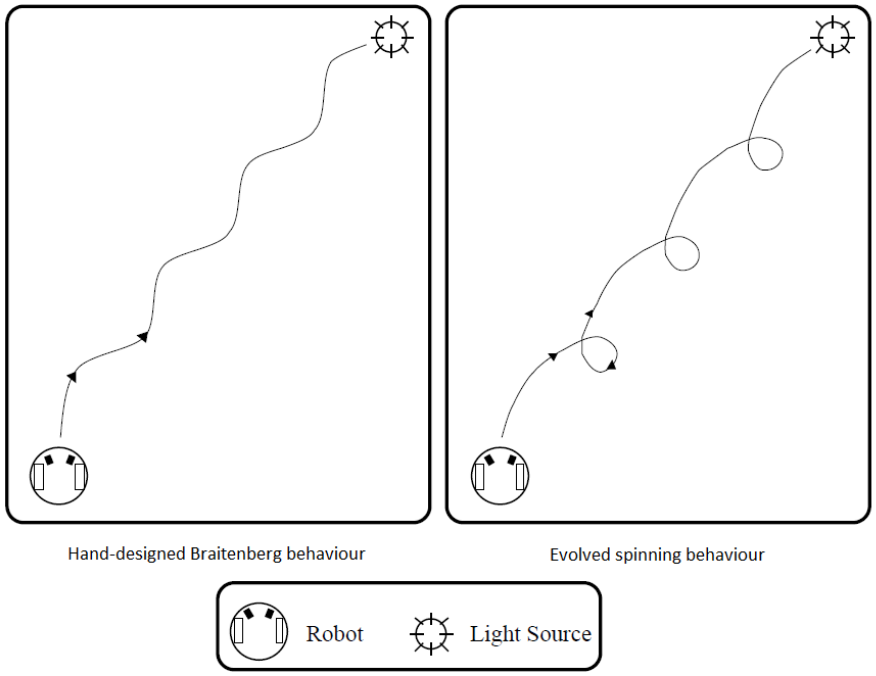

Figure 8. Light-seeking robot movement. The path of the hand-coded Braitenberg-style movement (left) and evolved spinning movement (right) when moving towards a light source.

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 [97] is unlikely to surprise anyone, because of close and constant interactions with the user. And yet, in the case of Picbreeder [98], one such surprise was career-altering. Picbeeder is a platform, similar in form to the classic Blind watchmaker [6], 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 [99], 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.

Impressive Digital 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.

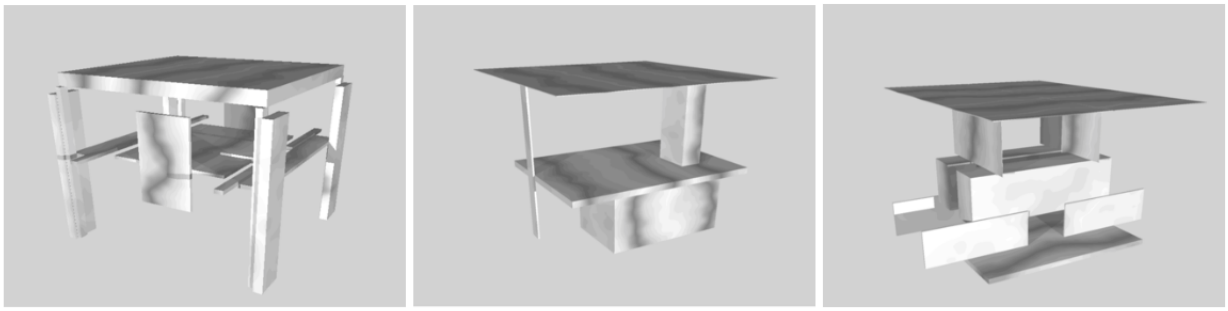

Figure 9. Table designs. Three table designs evolved using the generic evolutionary design system [100].

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 an astronomical sea of possibilities, and how would you specify a meaningful fitness function?

In response, Bentley created the Generic Evolutionary Design system [100], 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? And for later designs such as optical prisms, automobiles and boat hulls: Does an object refract the light in a desired way? Does the air exert specific forces on certain parts of the design? Is the design aerodynamic? Is the design hydrodynamic?

The task put to the Generic Evolutionary Design system was to evolve a table. Sure enough, from random initial designs emerged multiple elegant and sculptural 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") (Fig. 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, i.e. an evolving track would be scored higher the more it resembled the targets. The evolved results, 8-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 intricate novel dance tracks with clear stylistic resemblance.

In 2000 the group formed a record label named J13 Records. A highly specialized 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). Sworn to secrecy, 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 authored by computers.

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

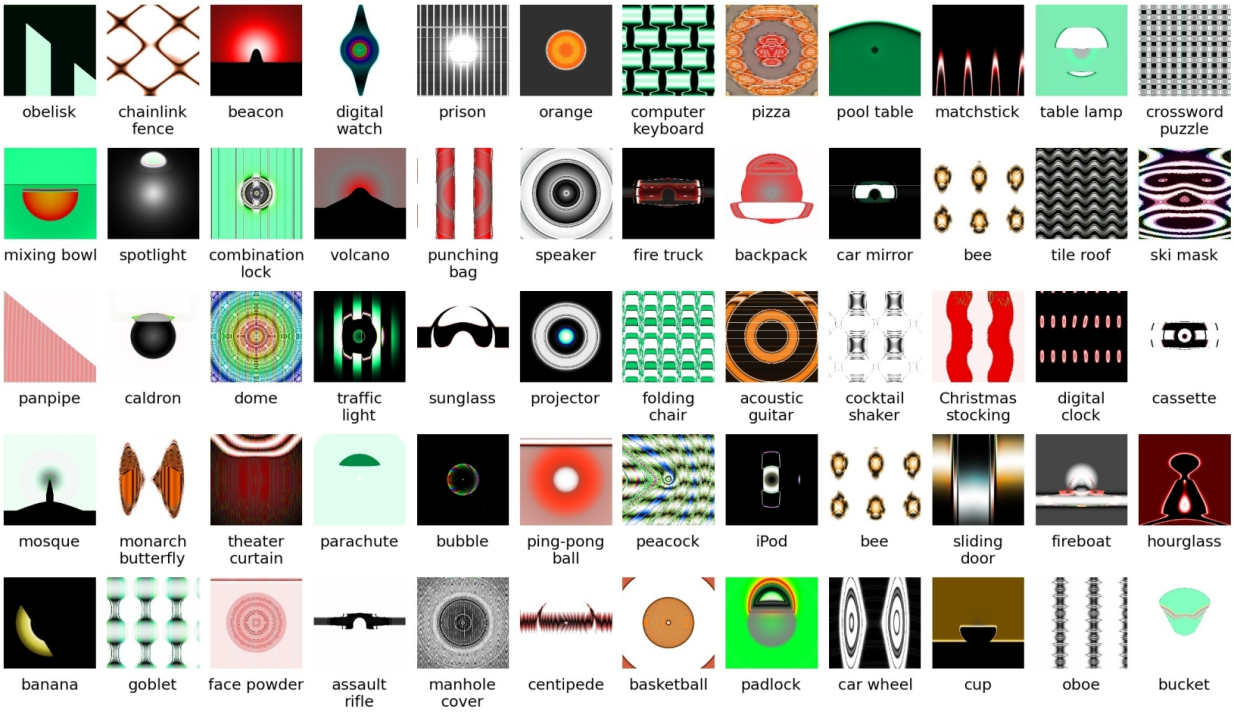

Figure 10. Images generated by Innovation Engines. A selection of images evolved via an Innovation Engine. Underneath each image is the type of image that evolution was challenged to generate.

different objectives (instead of a single objective as in traditional digital evolution), and (3) harness powerful deep neural networks to evaluate how interesting a 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 human observers (Fig. 10). Interestingly, the images have diverse aesthetic styles, and bear resemblance to abstract "concept art" pieces that reflect intelligent statements about their theme (e.g. the two different images of prison cells, the beacon, and the folding chairs in Fig. 10). Furthermore, the genomes of these algorithmically-produced images are quantitatively similar to the elegant, compact genomes evolved by humans on the interactive evolution website Picbreeder [102].

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 (Fig. 11).

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* as 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 section demonstrates how in practice there often exists surprising convergence between evolution in digital and biological media.

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.

Evolution of Muscles and Bones In further results from Cheney et al.'s virtual creatures system [68], evolution generated locomotion strategies unexpectedly convergent with those of biological creatures, examples of which are shown in Fig. 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 muscle pairs in humans, e.g. biceps and triceps – and also to place them in a functional way. The gait in the bottom figure highlights digital evolution's use of a stiff bone-like 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.

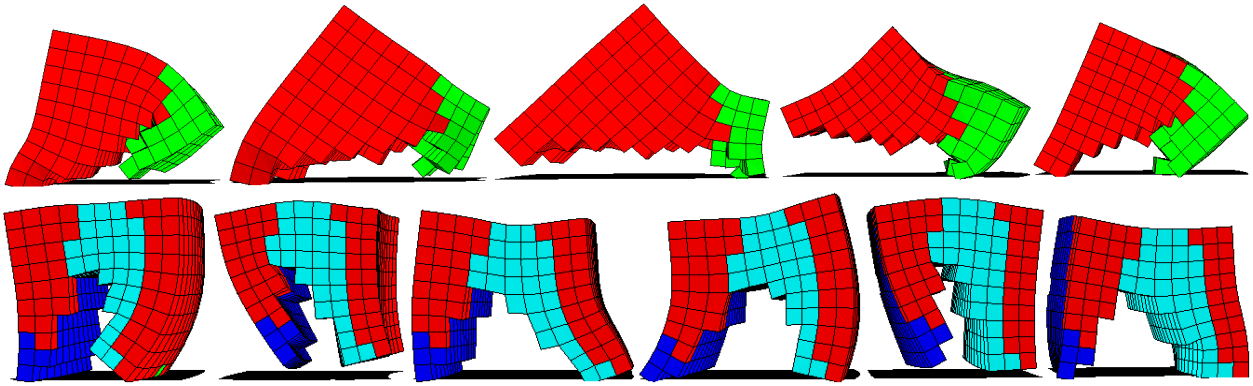

Figure 12. A stop-motion view of a small sample of the evolved gaits from Cheney et al. [68], which produced surprisingly effective and lifelike behaviors. Shown here are soft robots progressing from left to right across the panel. Colors correspond to voxel types (with red and green denoting oppositely 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 (bone-like) support material allows evolution to create relatively long appendages and produce a horse-like galloping behavior. Videos of various soft robot gaits, including these two, can be found at `https://youtu.be/z9ptOeByLA4?list=PL5278ezwmoxQODgYBOhWnCO-Ob09GZGe2`.

Evolution of Parasitism In 1990, Tom Ray developed his seminal artificial life system, Tierra [103], an early instance of evolution by natural selection in a digital medium. Organisms in Tierra consist of

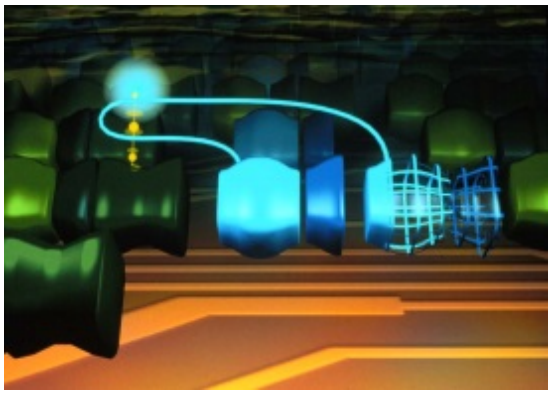

Figure 13. Parasites in Tierra. A self-replicator (green, left) has code that copies the genome from parent to offspring. The parasite (blue, center) lacks the genome replicating code, and executes that code in its neighbor, copying its genome into its offspring (blue shell, right). The blue sphere represents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

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, but he expected to spend perhaps years tinkering before anything interesting would happen; surprisingly, Tierra produced fascinating, complex ecologies the very first time it ran without crashing [103].

What emerged was a series of competing adaptations between replicating organisms within the computer, an ongoing co-evolutionary dynamic. The surprisingly large palette of emergent behaviors included parasitism (Fig. 13), immunity to parasitism, circumvention of immunity, hyper-parasitism (Fig. 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 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 an *informational* parasitism. Evolving matching templates enabled exploitation, while non-complementary templates allowed individuals to escape exploitation. Ray termed the underlying process *bit-string races*, echoing the idea of evolutionary and ecological arms-races in nature.

But the dynamics went much further than only bit-string races. Hyper-parasites stole the CPUs of parasites, an *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. As in natural evolution, a rich diversity of social and ecological interactions evolved in complex ways.

Digital Vestigial Organs Virtual creatures evolved in the ERO system by Krcah [61] 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 Fig. 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

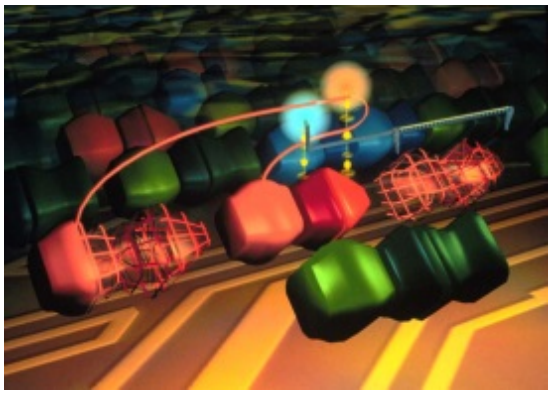

Figure 14. Hyper-parasites in Tierra. A red hyper-parasite (center), has captured a CPU (blue sphere) from a parasite, and is using it to replicate its genome into the shell on the right. The hyper-parasite also has its own CPU (red sphere) that it is using to replicate also into the shell at the left.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

limitation of the evolutionary algorithm: Mutation was implemented such that body parts were never entirely removed from any creature. The "fin" body part from Fig. 15 had 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.

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 [35, 104, 39, 41, 42, 43, 44, 45, 48, 105, 49, 50]. During the submission process for a paper on genetic architecture and sexual reproduction [42], reviewers pointed out that some data was unexpectedly bi-modal: 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.

Evolving Complex Behavior May Involve Temporary Setbacks In a pioneering study, Lenski and colleagues used the Avida digital evolution platform to test some of Darwin's hypotheses about the evolution of complex features [35]. 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

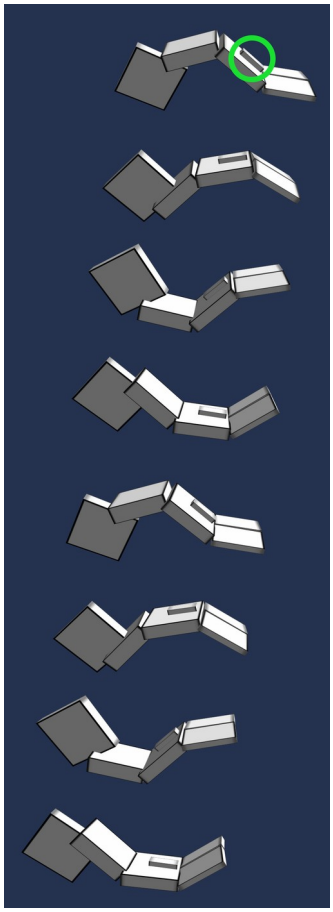

Figure 15. Swimming creature with an atrophied body part. A body part that was functionally important to an ancestor of the depicted creature became atrophied over evolutionary time. Mutations within this system did not allow deleting parts entirely; as a result, evolution shrunk the part and tuned its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

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 and a treasure trove of information 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, perhaps, 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.

Drake's rule Sometimes, awesome results are right under your nose but you need a chance encounter to realize it. The Aevol digital evolution model, which belongs to the so-called "sequence-of-nucleotides" formalism [106], 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 be a complete failure. Yet one day in a corridor, Knibbe bumped into Laurent Duret, a renowned bioinformatician. Knibbe related her disappointing PhD 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 non-coding part, but that's trivial, isn't it?". Laurent disagreed, "The non-coding 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 with the inverse of their mutation rate [107], and (ii) no model had predicted a scaling between the mutation rates and the non-coding 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 quasispecies model [108]. The effect on the non-coding region could be observed in Aevol because the model included chromosomal rearrangements in addition to point mutations [109]. 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 [110].

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 fascinated researchers for decades. Their existence may seem counter-intuitive, 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 [111, 112]. In such research, Aevol individuals are given the ability to secrete a public good molecule, which benefits all digital organisms in the neighborhood. However, the public good molecule is costly 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. They evolved populations that would secrete the public good molecule by lowering its cost, and then extended the evolution under an increased cost. Interestingly, while most populations quickly lost all their secretion genes, some consistently did not, even when loss experiments were repeated many times.

The genetic analysis of these stubborn populations led to an interesting and surprising result. The secretion genes that survived increase in cost were frequently overlapping with crucial metabolic genes, meaning that they were physically encoded using the same DNA base pairs than a metabolic gene, but using the opposite strand or another reading frame [113]. 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 [114, 115], but overlapping genes had never been studied in this context. Amusingly, when Frenoy, a master student at the time, manually

looked at the genomes which preserved secretion despite its cost to try to understand how they were different, he did not know what was gene overlap and thought it was an artefact of the Aevol system. It is only when presenting his results during a lab meeting that his colleagues pointed him toward the existence of overlapping genes in nature, and the fact that selection pressures on such genetic systems are not yet fully understood.

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 papers, 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 paper was to collect and validate the true stories from the original scientists and collect them for posterity.

But beyond claims about relevance to biology, the ubiquity of surprising and creative outcomes in digital evolution has other cross-cutting implications. For example, the many examples of "selection gone wild" in this article connect to the nascent 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 [116, 117]. 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 [118, 119]. Echoing our "selection gone wild" section, the first rule of directed evolution is "you get what you select for [119]." 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, 121]. 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 [122].

Conclusion

Across a compendium of examples we have reviewed many ways in which digital evolution produces surprising and creative solutions. The diversity and abundance of these examples suggest that surprise in digital evolution is common, rather than a rare exception. For every story we received or heard, there are likely to be many others that have been already forgotten as researchers retire. The ubiquity of these anecdotes also means that creativity is not confined to evolution in nature, but appears to be a pervasive feature of evolutionary processes in general.

These anecdotes thus serve as evidence that evolution—whether biological or computational—is inherently creative, and should routinely be expected to surprise, delight, and even outwit us.

Acknowledgements

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. We also thank Elizabeth for suggestions of anecdotes from Avida to include. 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. Schöner MG, Schöner CR, Simon R, Grafe TU, Puechmaille SJ, Ji LL, et al. Bats are acoustically attracted to mutualistic carnivorous plants. Current Biology. 2015;25(14):1911–1916.
2. Makarova KS, Aravind L, Wolf YI, Tatusov RL, Minton KW, Koonin EV, et al. Genome of the extremely radiation-resistant bacterium Deinococcus radiodurans viewed from the perspective of comparative genomics. Microbiology and Molecular Biology Reviews. 2001;65(1):44–79.
3. Strahs G. Biochemistry at 1000C: Explosive secretory discharge of bombardier beetles (Brachinus). Science. 1969;.
4. Lefevre T, Adamo SA, Biron DG, Misse D, Hughes D, Thomas F. Invasion of the body snatchers: the diversity and evolution of manipulative strategies in host–parasite interactions. Advances in Parasitology. 2009;68:45–83.
5. Futuyma DJ. Natural selection and adaptation. The Princeton Guide to Evolution. 2013; p. 189.
6. Dawkins R. The blind watchmaker: Why the evidence of evolution reveals a universe Without design. WW Norton & Company; 1986.
7. Madigan MT. Bacterial habitats in extreme environments. In: Journey to Diverse Microbial Worlds. Springer; 2000. p. 61–72.
8. University of Utah biologists surprised to discover "ultrafast recycling" at nerve synapses; 2013. `http://kuer.org/post/u-u-biologists-surprised-discover-ultrafast-recycling-nerve-synapses`.
9. Moonlight drives winter 'werewolves' to gather for Arctic Ocean odyssey; 2016. `http://www.sams.ac.uk/arctic-werewolves-1/moonlight-drives-winter-werewolves-to-gather-for-arctic-ocean-odyssey`.
10. Lau NC, Bartel DP. Censors of the genome. Scientific American. 2003;289(2):34–41.
11. Dobzhansky T. Chance and creativity in evolution. In: Studies in the Philosophy of Biology. Springer; 1974. p. 307–338.
12. Bentley PJ. Is evolution creative? In: Proceedings of the AISB. vol. 99. Citeseer; 1999. p. 28–34.
13. Dawkins R. Universal Darwinism. The Nature of Life: Classical and Contemporary Perspectives from Philosophy and Science. 1983;.
14. Dennett DC. Darwin's dangerous idea. Simon and Schuster; 2014.
15. De Jong KA. Evolutionary computation: A unified approach. MIT press; 2006.
16. Langton CG. Artificial life: An overview. Mit Press; 1997.
17. Mansanne F, Carrere F, Ehinger A, Schoenauer M. Evolutionary algorithms as fitness function debuggers. In: International Symposium on Methodologies for Intelligent Systems. Springer; 1999. p. 639–647.

18. Pennock RT. Can Darwinian mechanisms make novel discoveries?: Learning from discoveries made by evolving neural networks. Foundations of Science. 2000;5(2):225–238.

19. Grabowski LM, Bryson DM, Dyer FC, Pennock RT, Ofria C. A case study of the de novo evolution of a complex odometric behavior in digital organisms. PLoS One. 2013;8(4):e60466.

20. Darwin C. On the Origin of Species; 1859.

21. Wilson EO. The diversity of life. WW Norton & Company; 1999.

22. Runco MA, Jaeger GJ. The standard definition of creativity. Creativity Research Journal. 2012;24(1):92–96.

23. Gould SJ, Vrba ES. Exaptation—a missing term in the science of form. Paleobiology. 1982;8(01):4–15.

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

25. Kirschner M, Gerhart J. Evolvability. Proceedings of the National Academy of Sciences of the United States of America. 1998;95(15):8420.

26. Kouvaris K, Clune J, Kounios L, Brede M, Watson RA. How evolution learns to generalise: Using the principles of learning theory to understand the evolution of developmental organisation. PLoS Computational Biology. 2017;.

27. Kounios L, Clune J, Kouvaris K, Wagner GP, Pavlicev M, Weinreich DM, et al. Resolving the paradox of evolvability with learning theory: How evolution learns to improve evolvability on rugged fitness landscapes. arXiv preprint arXiv:161205955. 2016;.

28. Endler JA, Greenwood JJD. Frequency-dependent predation, crypsis and aposematic coloration. Philosophical Transactions of the Royal Society of London B, Biological Sciences. 1988;319(1196):505–523.

29. Schluter D. The ecology of adaptive radiation. OUP Oxford; 2000.

30. Lenski RE. Get A Life. Science. 1998;280(5365):849–850.

31. Dennett D. Encyclopedia of Evolution. In: Pagel M, editor. Encyclopedia of Evolution. Oxford Univ. Press; 2002. p. E83–E92.

32. Ofria C, Wilke CO. Avida: A software platform for research in computational evolutionary biology. Artificial life. 2004;10(2):191–229.

33. Ray TS. An evolutionary approach to synthetic biology: Zen and the art of creating life. Artificial Life. 1993;1(1-2):179–209.

34. Lehman J, Stanley KO. Investigating Biological Assumptions through Radical Reimplementation. Artificial Life. 2014;21(1):21–46.

35. Lenski RE, Ofria C, Pennock RT, Adami C. The evolutionary origin of complex features. Nature. 2003;423(6936):139–144.

36. Clune J, Mouret JB, Lipson H. The evolutionary origins of modularity. Proceedings of the Royal Society B. 2013;280(20122863).

37. Kashtan N, Alon U. Spontaneous evolution of modularity and network motifs. Proceedings of the National Academy of Sciences. 2005;102(39):13773–13778.

38. Kashtan N, Noor E, Alon U. Varying environments can speed up evolution. Proceedings of the National Academy of Sciences. 2007;104(34):13711.

39. Lenski RE, Ofria C, Collier TC, Adami C. Genome complexity, robustness and genetic interactions in digital organisms. Nature. 1999;400(6745):661–664.

40. Wagner GP, Pavlicev M, Cheverud JM. The road to modularity. Nature Reviews Genetics. 2007;8(12):921–931.

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

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

43. Adami C, Ofria C, Collier TC. Evolution of biological complexity. Proceedings of the National Academy of Sciences of the United States of America. 2000;97(9):4463.

44. Wilke CO, Wang JL, Ofria C, Lenski RE, Adami C. Evolution of digital organisms at high mutation rates leads to survival of the flattest. Nature. 2001;412(6844):331–333.

45. Chow SS, Wilke CO, Ofria C, Lenski RE, Adami C. Adaptive radiation from resource competition in digital organisms. Science. 2004;305(5680):84.

46. Cully A, Clune J, Tarapore D, Mouret JB. Robots that can adapt like animals. Nature. 2015;521(7553):503–507.

47. Yedid G, Bell G. Macroevolution simulated with autonomously replicating computer programs. Nature. 2002;420(6917):810.

48. Lenski RE, Barrick JE, Ofria C. Balancing robustness and evolvability. PLoS Biology. 2006;4(12):e428.

49. Goldsby HJ, Dornhaus A, Kerr B, Ofria C. Task-switching costs promote the evolution of division of labor and shifts in individuality. Proceedings of the National Academy of Sciences. 2012;109(34):13686–13691.

50. Covert AW, Lenski RE, Wilke CO, Ofria C. Experiments on the role of deleterious mutations as stepping stones in adaptive evolution. Proceedings of the National Academy of Sciences. 2013;110(34):E3171–E3178.

51. Pennock RT. Models, simulations, instantiations, and evidence: the case of digital evolution. Journal of Experimental & Theoretical Artificial Intelligence. 2007;19(1):29–42.

52. Turing AM. On computable numbers, with an application to the Entscheidungsproblem. Journal of Math. 1936;58(345-363):5.

53. Langton CG. Computation at the edge of chaos: phase transitions and emergent computation. Physica D: Nonlinear Phenomena. 1990;42(1-3):12–37.

54. Flake GW. The computational beauty of nature: Computer explorations of fractals, chaos, complex systems, and adaptation; 1998.

55. Roese NJ, Vohs KD. Hindsight bias. Perspectives on Psychological Science. 2012;7(5):411–426.

56. Korzybski A. Science and sanity: An introduction to non-Aristotelian systems and general semantics. Institute of GS; 1958.

57. Campbell DT. Assessing the impact of planned social change. Evaluation and Program Planning. 1979;2(1):67–90.

58. Goodhart CA. Problems of monetary management: The UK experience. Springer; 1984.

59. Sims K. Evolving 3D morphology and behavior by competition. Artificial Life. 1994;1(4):353–372.

60. Sims K. Evolving virtual creatures. In: Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques. ACM; 1994. p. 15–22.

61. Krcah P. Towards efficient evolutionary design of autonomous robots. In: Evolvable Systems: From Biology to Hardware, 8th International Conference, ICES 2008, Prague, Czech Republic, September 21-24, 2008. Proceedings. Springer; 2008. p. 153–164.

62. Forrest S, Nguyen T, Weimer W, Le Goues C. A genetic programming approach to automated software repair. In: Proceedings of the Genetic and Evolutionary Computation Conference; 2009. p. 947–954.

63. Koza JR. Genetic programming: On the programming of computers by means of natural selection. vol. 1. MIT press; 1992.

64. Weimer W. Advances in Automated Program Repair and a Call to Arms. In: Search Based Software Engineering - 5th International Symposium, SSBSE 2013, St. Petersburg, Russia, August 24-26, 2013. Proceedings; 2013. p. 1–3.

65. Schulte E, Forrest S, Weimer W. Automated program repair through the evolution of assembly code. In: Proceedings of the IEEE/ACM International Conference on Automated Software Engineering. ACM; 2010. p. 313–316.

66. Ellefsen KO, Mouret JB, Clune J. Neural modularity helps organisms evolve to learn new skills without forgetting old skills. PLoS Computational Biology. 2015;11(4):e1004128.

67. Soltoggio A, Bullinaria JA, Mattiussi C, Durr P, Floreano D. Evolutionary advantages of neuromodulated plasticity in dynamic, reward-based scenarios. Artificial Life. 2008;11.

68. Cheney N, MacCurdy R, Clune J, Lipson H. Unshackling evolution: evolving soft robots with multiple materials and a powerful generative encoding. In: Proceedings of the Genetic and Evolutionary Computation Conference. ACM; 2013. p. 167–174.

69. O'shea DC. Monochromatic quartet: A search for the global optimum. In: International Lens Design Conference. International Society for Optics and Photonics; 1991. p. 548–554.

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

71. Moriarty DE, Miikkulainen R. Discovering complex Othello strategies through evolutionary neural networks. Connection Science. 1995;7(3):195–209.

72. Moriarty DE, Miikkulainen R. Forming neural networks through efficient and adaptive co-evolution. Evolutionary Computation. 1997;5:373–399.

73. Feldt R. Generating diverse software versions with genetic programming: An experimental study. IEE Proceedings - Software Engineering. 1998;145(6):228–236.

74. Feldt R. Genetic programming as an explorative tool in early software development phases. In: Proceedings of the 1st International Workshop on Soft Computing Applied to Software Engineering; 1999. p. 11–20.

75. Feldt R. Biomimetic software engineering techniques for dependability. Department of Computer Engineering, Chalmers University of Technology. Gothenburg, Sweden; 2002.

76. Harman M, Mansouri SA, Zhang Y. Search-based software engineering: Trends, techniques and applications. ACM Comput Surv. 2012;45(1):11:1–11:61.

77. Stanley KO, Bryant BD, Miikkulainen R. Real-time neuroevolution in the NERO video game. IEEE Transactions on Evolutionary Computation. 2005;9(6):653–668.

78. Salimans T, Ho J, Chen X, Sutskever I. Evolution strategies as a scalable alternative to reinforcement learning. arXiv preprint arXiv:170303864. 2017;.

79. Such FP, Madhavan V, Conti E, Lehman J, Stanley KO, Clune J. Deep Neuroevolution: Genetic Algorithms Are a Competitive Alternative for Training Deep Neural Networks for Reinforcement Learning. arXiv preprint arXiv:171206567. 2017;.

80. Chrabaszcz P, Loshchilov I, Hutter F. Back to Basics: Benchmarking Canonical Evolution Strategies for Playing Atari. arXiv preprint arXiv:180208842. 2018;.

81. Mnih V, Kavukcuoglu K, Silver D, Rusu AA, Veness J, Bellemare MG, et al. Human-level control through deep reinforcement learning. Nature. 2015;518(7540):529.

82. Mnih V, Badia AP, Mirza M, Graves A, Lillicrap T, Harley T, et al. Asynchronous methods for deep reinforcement learning. In: International Conference on Machine Learning; 2016. p. 1928–1937.

83. Rechenberg I. Evolutionsstrategie–Optimierung technisher Systeme nach Prinzipien der biologischen Evolution. 1973;.

84. Vincent J. A video game-playing AI beat Q*bert in a way no one's ever seen before. The Verge. 2018;.

85. Ecarlat P, Cully A, Maestre C, Doncieux S. Learning a high diversity of object manipulations through an evolutionary-based babbling. In: Proceedings of Learning Objects Affordances Workshop at IROS 2016; 2015. p. 1–2.

86. Mouret JB, Clune J. Illuminating search spaces by mapping elites. arXiv preprint arXiv:150404909. 2015; p. 1–15.

87. Moriarty DE, Miikkulainen R. Evolving obstacle avoidance behavior in a robot arm. In: Maes P, Mataric MJ, Meyer JA, Pollack J, Wilson SW, editors. From Animals to Animats 4: Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior. Cambridge, MA: MIT Press; 1996. p. 468–475.

88. van der Smagt P. Simderella: A robot simulator for neuro-controller design. Neurocomputing. 1994;6(2).

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

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

91. O'Reilly UM, Wagy M, Hodjat B. Ec-star: A massive-scale, hub and spoke, distributed genetic programming system. In: Genetic Programming Theory and Practice X. Springer; 2013. p. 73–85.

92. Koza JR. A hierarchical approach to learning the Boolean multiplexer function. Foundations of Genetic Algorithms. 1990; p. 171–192.

93. Thompson A. An evolved circuit, intrinsic in silicon, entwined with physics. In: International Conference on Evolvable Systems. Springer; 1996. p. 390–405.

94. Watson RA, Ficici SG, Pollack JB. Embodied evolution: Distributing an evolutionary algorithm in a population of robots. Robotics and Autonomous Systems. 2002;39(1):1–18.

95. Watson RA, Ficiei S, Pollack JB. Embodied evolution: Embodying an evolutionary algorithm in a population of robots. In: Proceedings of the 1999 Congress on Evolutionary Computation. IEEE; 1999. p. 335–342.

96. Braitenberg V. Vehicles: Experiments in synthetic psychology. MIT press; 1986.

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

98. Secretan J, Beato N, Ambrosio DBD, Rodriguez A, Campbell A, Folsom-Kovarik JT, et al. Picbreeder: A case study in collaborative evolutionary exploration of design space. Evolutionary Computation. 2011;19(3):345–371.

99. Lehman J, Stanley KO. Abandoning objectives: Evolution through the search for novelty alone. Evolutionary Computation. 2011;19(2):189–223.

100. Bentley PJ. Generic evolutionary design of solid objects using a genetic algorithm. The University of Huddersfield; 1996.

101. Nguyen A, Yosinski J, Clune J. Understanding innovation engines: Automated creativity and improved stochastic optimization via deep learning. Evolutionary Computation. 2016;.

102. Secretan J, Beato N, D'Ambrosio DB, Rodriguez A, Campbell A, Folsom-Kovarik JT, et al. Picbreeder: A case study in collaborative evolutionary exploration of design space. Evolutionary Computation. 2011;19(3):373–403.

103. Ray T. J'ai joué á Dieu et créé la vie dans mon ordinateur. Le Temps stratégique. 1992;47:68–81.

104. Adami C. Digital genetics: Unravelling the genetic basis of evolution. Nature Reviews Genetics. 2006;7(2):109–118.

105. Elsberry WR, Grabowski LM, Ofria C, Pennock RT. Cockroaches, drunkards, and climbers: Modeling the evolution of simple movement strategies using digital organisms. In: Proceedings of IEEE Symposium on Artificial Life.; 2009. p. 92–99.

106. Hindré T, Knibbe C, Beslon G, Schneider D. New insights into bacterial adaptation through in vivo and in silico experimental evolution. Nature Reviews Microbiology. 2012;10:352–365.

107. Drake JW. A constant rate of spontaneous mutation in DNA-based microbes. Proceedings of the National Academy of Sciences. 1991;88(16):7160–7164.

108. Eigen M. Self-organization of matter and the evolution of biological macromolecules. Die Naturwissenschaften. 1971;58(10):465–523.

109. Knibbe C, Coulon A, Mazet O, Fayard JM, Beslon G. A long-term evolutionary pressure on the amount of non-coding DNA. Molecular Biology and Evolution. 2007;24(10):2344–2353.

110. Fischer S, Bernard S, Beslon G, Knibbe C. A model for genome size evolution. Bulletin of Mathematical Biology. 2014;76:2249–2291.

111. Misevic D, Frénoy A, Parsons DP, Taddei F. Effects of public good properties on the evolution of cooperation. Artificial Life. 2012;13:218–225.

112. Frénoy A, Taddei F, Misevic D. Robustness and evolvability of cooperation. In: Proceedings of Artificial Life XIII; 2012. p. 53–58.

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

114. Foster KR, Shaulsky G, Strassmann JE, Queller DC, Thompson CR. Pleiotropy as a mechanism to stabilize cooperation. Nature. 2004;431(7009):693–696.

115. Nogueira T, Rankin DJ, Touchon M, Taddei F, Brown SP, Rocha EP. Horizontal gene transfer of the secretome drives the evolution of bacterial cooperation and virulence. Current Biology. 2009;19(20):1683–1691.

116. Amodei D, Olah C, Steinhardt J, Christiano P, Schulman J, Mané D. Concrete problems in AI safety. arXiv preprint arXiv:160606565. 2016;.

117. Bostrom N. Superintelligence: Paths, dangers, strategies. OUP Oxford; 2014.

118. Arnold FH. Design by directed evolution. Accounts of Chemical Research. 1998;31(3):125–131.

119. Peisajovich SG, Tawfik DS. Protein engineers turned evolutionists. Nature Methods. 2007;4(12):991–994.

120. Zhao H, Arnold FH. Combinatorial protein design: Strategies for screening protein libraries. Current Opinion in Structural Biology. 1997;7(4):480–485.

121. Schmidt-Dannert C, Arnold FH. Directed evolution of industrial enzymes. Trends in Biotechnology. 1999;17(4):135–136.

122. Smaldino PE, McElreath R. The natural selection of bad science. Royal Society Open Science. 2016;3(9):160384.