nature machine intelligence

PERSPECTIVE

https://doi.org/10.1038/s42256-020-00278-8

A biological perspective on evolutionary computation

Risto Miikkulainen[1,3] ✉ and Stephanie Forrest[2,3] ✉

Evolutionary computation is inspired by the mechanisms of biological evolution. With algorithmic improvements and increasing computing resources, evolutionary computation has discovered creative and innovative solutions to challenging practical problems. This paper evaluates how today's evolutionary computation compares to biological evolution and how it may fall short. A small number of well-accepted characteristics of biological evolution are considered: openendedness, major transitions in organizational structure, neutrality and genetic drift, multi-objectivity, complex genotype-to-phenotype mappings and co-evolution. Evolutionary computation exhibits many of these to some extent but more can be achieved by scaling up with available computing and by emulating biology more carefully. In particular, evolutionary computation diverges from biological evolution in three key respects: it is based on small populations and strong selection; it typically uses direct genotype-to-phenotype mappings; and it does not achieve major organizational transitions. These shortcomings suggest a roadmap for future evolutionary computation research, and point to gaps in our understanding of how biology discovers major transitions. Advances in these areas can lead to evolutionary computation that approaches the complexity and flexibility of biology, and can serve as an executable model of biological processes.

The 2018 Nobel Prize in Chemistry, awarded to Frances Arnold, underscored a remarkable trend in engineering: it is now feasible, and even necessary, to employ automated methods to augment human design and creativity. Today's designs have increased in scale and complexity to the point that humans can no longer comprehend the design space or consider all reasonable possibilities. Breaking through this design barrier is an important challenge that cuts across many areas of engineering.

Arnold used directed evolution to design enzymes with improved and new functions. Directed evolution creates variation through random mutation, and variations are selected and amplified according to a specified design goal (for example, catalysing useful reactions). Human experts define the problem, and automated evolution performs the search, often finding solutions that perform better than those designed by human experts.

Evolutionary computation (EC) aims to harness this process in a computational framework. EC is distinctly different from machine learning approaches such as deep learning, which learn predictive models of phenomena for which correct answers are known. By contrast, EC creates new solutions by iteratively applying mutation, recombination and selection to populations of digital individuals (Fig. 1). These methods, known variously as genetic algorithms[1–3], genetic programming[4] and evolutionary strategies[5], have been applied to a variety of problems requiring engineering and scientific creativity[6–9]. Given the computational power and data that have recently become available—millions of times more than just two decades ago—it is now practical to simulate real-world processes and evolve solutions to engineered systems that interact with them. Examples include: simulating and designing growth recipes for agriculture that are counter-intuitive but outperform humans[10], designing improved treatments for diseases and injuries[11,12], controlling robots and vehicles where human-designed controls are ineffective[13,14] and creating improved designs for machines and chemical processes[15,16]. Beyond these practical applications, EC has been used to create music and art, some of which has been accepted to human exhibitions[17–20]. It has also been applied to software itself, repairing bugs[21,22], designing neural network architectures[23,24], improving compilers[25] and finding energy-efficient versions of programs[26]. This progress in several domains, together with the design complexities faced by engineering and software, suggests that evolutionary methods are poised to augment human design processes in many applications.

Although EC has produced solutions for highly complex systems, they pale in comparison to the robustness, versatility and adaptability of solutions produced by biological evolution, raising the following questions: How does EC today compare with biological evolution? Are there aspects of biological evolution still missing that could make EC more powerful? Conversely, can some of the challenges facing EC point to gaps in our understanding of biological evolution?

To address these questions, this paper compares EC with biological evolution along a small number of dimensions that are generally accepted as essential hallmarks and mechanisms of evolution. Not all such dimensions are included (compare with refs. [27,28]); the discussion focuses on those that lead to most relevant insights into EC today.

Comparing EC with biological evolution

The elements underlying biological evolution have been elucidated and refined over the 160 years since Darwin's theory was proposed. There is general agreement today that natural selection, mutation, recombination and random drift are the key drivers of evolution[29]. In addition, there are several hallmarks that characterize the evolutionary process and mechanisms that enable it. Of all these dimensions, this paper focuses on six that provide the most insight for EC: openendedness, major transitions in organizational structure, neutrality and random drift, multi-objectivity, complex genotype-to-phenotype mappings, and co-evolution.

[1]The University of Texas at Austin and Cognizant Technology Solutions, Austin, TX, USA. [2]Arizona State University and the Santa Fe Institute, Tempe, AZ, USA. [3]These authors contributed equally: Risto Miikkulainen, Stephanie Forrest. ✉e-mail: risto@cs.utexas.edu; steph@asu.edu

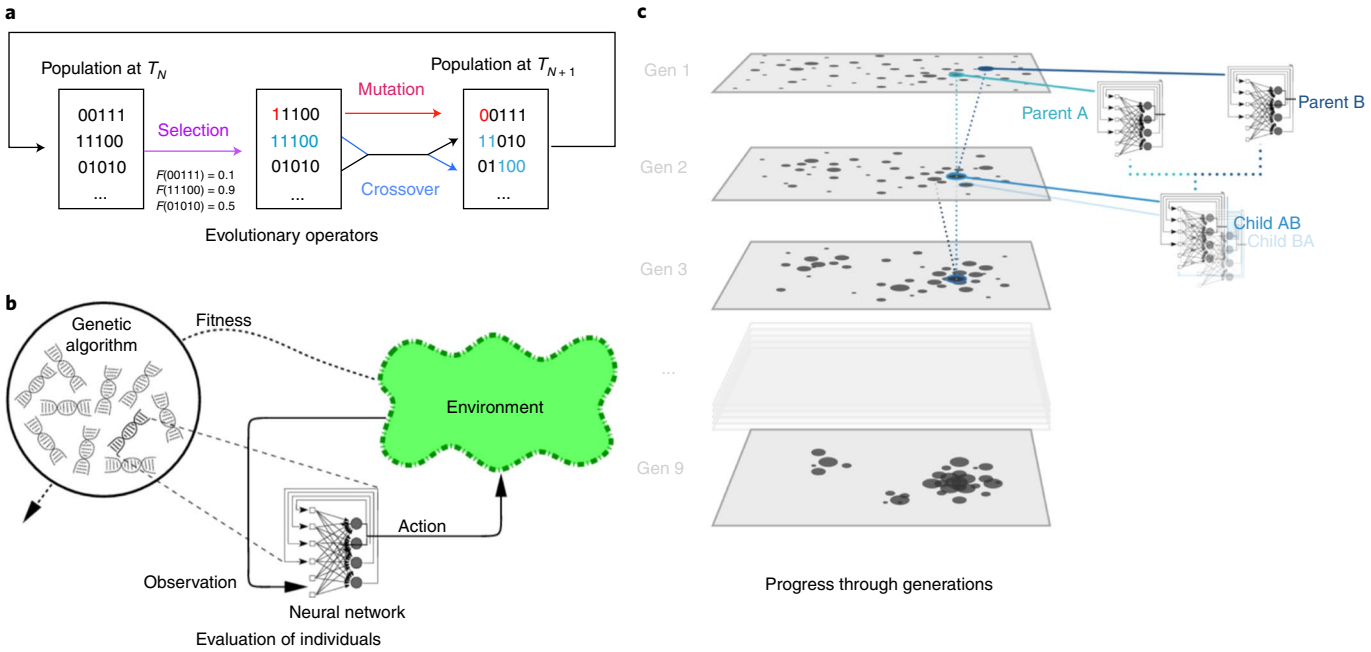

Fig. 1 | EC mechanisms. a, Genotypes of individuals in the population are encoded as strings or trees of values, for example, binary numbers that encode the weight values of a neural network. High-fitness individuals are selected as parents for the next generation, their encodings are recombined, and individual values are mutated. The process repeats until a stopping criterion is met. **b**, To evaluate an individual's fitness, its genotype is decoded into a phenotype, such as a neural network, and it is evaluated in an environment. For instance, the neural network might observe the environment and control the actions of an agent in the environment. The individual's performance is interpreted as the fitness for that genotype in the population. Note that the correct actions do not need to be known, as would be required for deep learning; an overall measure of fitness is sufficient, enabling the discovery of new solutions rather than imitating known ones. **c**, The population is initialized to sample the space of possible individuals broadly. The space is represented as a rectangle in this visualization, individuals are dots, and the size of the dot corresponds to their fitness. Subsequent generations are created as shown in **a**. Over multiple generations, the population converges to high-fitness regions of the space.

There are other dimensions in which EC and biology differ. For instance, sexual recombination is important in biology and its implementation in EC (via the crossover operator) does not capture all of the biological richness that is implied by dominant/recessive genes and diploid structures. However, attempts to add such mechanisms to EC have not been particularly successful or generalizable, and they do not appear to be a promising source of insight for EC at this time. Table 1 summarizes these findings. For each dimension, Table 1 reports how it is manifested in biology and the extent to which it is manifested in EC, with three computational examples. The rightmost two columns highlight what we consider to be the greatest challenge and the greatest opportunity to move EC forward in each dimension. The subsections below discuss each dimension in detail.

Openendedness. Over time, evolution has often produced organisms with increasing complexity, reflecting the openended nature of the evolutionary process[30]. In computational terms, innovations in such a process become essential building blocks for future evolution, leading to nonstationary fitness landscapes and indefinitely long searches rather than rapid convergence[31,32]. A canonical example is the citric acid cycle for producing energy, which is an essential building block of all aerobic cells and plays a central role in many biochemical pathways. Once evolution produced the citric acid cycle, many diverse complex metabolic pathways were subsequently enabled[33]. Niche construction is another example, where organisms alter their environment in a way that affects natural selection, creating feedback to the evolutionary process.

An example of openendedness in EC is NEAT[34], which generates neural network architectures from a minimal network, that is, one node connecting inputs to outputs, iteratively adding nodes and links tailored to the specific problem. Several artificial life simulations have demonstrated increasingly complex organisms and behaviour, such as Tierra, Avida and Division Blocks[35–37]. Similarly in genetic programming (GP)[4] a program's size (genome) increases during a single run, and successful mutations can be used as building blocks for subsequent mutations and recombinations. Software repair, for example, GenProg[21,22], is similarly openended because the size of the program can increase during a single run.

In engineering, increasing complexity is not always desirable, yet openended evolution can lead to unexpected and useful solutions[17]. Conversely, unpredictability and surprising discoveries do not always indicate openendedness. Sometimes, they are simply clever ways of exploiting the ecology imposed by the fitness function, for example, when an evolved bug repair exploits a weakness in its test suite rather than addressing the underlying problem. Such distinctions do not exist in nature.

Because engineering aims to solve particular problems, openended approaches are rare today. However, there is no reason why EC cannot become more openended, if properly scaled, and openendedness may indeed be required for producing more creative and complex designs.

Major transitions. Major transitions[38–40] are one example of how biological evolution achieves openendedness. These evolutionary jumps, or transitions to a new level of organization, occurred when life progressed from replicating molecules to chromosomes, and further to cells, to plastids, to multicellularity, to eusocial societies, and to societies with language and culture. In these transitions, individual units formed groups, differentiated into distinct and

Table 1 | A comparison of biological and computational evolution across several characteristic dimensions

Dimension	In biology	In computation	Example computational approaches	Computational challenge	Opportunity
Openendedness	Apparent	Incipient	Avida, GenProg, Division Blocks[21,35,37]	Nonstationary fitness	Scale
Major transitions	Transformative	Minor	EVC, SEAM, Model-S[41–43]	Encapsulation	Develop biology
Neutrality and drift	Essential	Minimal	Novelty Search, MGA, GEVO[54,55,60]	Selection pressure	Emulate biology
Multi-objectivity	Implicit	Explicit	NSGA, Lexicase, Quality Diversity [72,73,78]	Dimensionality	Scale
Geno/pheno mappings	High DoF	Low DoF	CE, HyperNEAT, Deep GA[87–89]	Effective representations	Emulate biology
Co-evolution	Pervasive	Constrained	CoDeepNEAT, EUREQA, POET[23,102,103]	Generality	Scale

cooperative roles, and lost their ability to replicate independently. Even though the information necessary for life is still represented in the molecular code, the way that information is organized, transmitted and selected changes with each transition, leading to a transformative new level of individuality.

Even with an explicit focus on openendedness, EC has not yet produced jumps that could be considered major transitions. Several researchers have developed methods to encourage transitions to new organizational structures, but so far they are limited in scope to mathematical functions (for example, ADFs in GP[4]) and abstract mathematical games (for example, Model-S in the Game of Life[41] and SEAM in HIFF[42]). By contrast, a major transition in EC might occur, for instance, if evolving agents discovered how to create roads and vehicles that allowed them to move longer distances at higher speeds. Further evolution might then discover behaviours that take advantage of such mobility.

When transitions to a higher level of organization are engineered by humans, EC can construct increasing and possibly even openended complexity. For example, neural networks have been evolved that control virtual creatures (EVCs) to run, turn and hit[43]. These networks can then be encapsulated and used as modules to evolve pursuit, evasion, and attack. Such modular neural networks can be further encapsulated to evolve fight-or-flight behaviour. While such a process incorporates major transitions in network organization, each transition is engineered rather than evolved.

Computational mechanisms that discover such transitions as part of an evolutionary process would be a powerful extension of current EC. They would enable automatic discovery of complex structures and exhibit a more compelling form of openendedness than what we have today. This dimension remains a major challenge and future opportunity for EC research (see 'Opportunities and challenges').

Neutrality and genetic drift. Neutrality refers to genetic changes that leave an organism's reproductive fitness unchanged, and genetic drift refers to changes in gene frequencies that arise from chance rather than selection. Both are well studied in evolutionary biology, and thought to be required for natural populations to evolve and innovate[29,44–46].

Specialized computer simulations have been used to model neutrality in biology[35,36,47,48], and traditional selection-based EC algorithms have taken advantage of neutrality in particular problems[49–53]. For example, novelty search[54] is a recent technique in EC that eschews fitness entirely, grading solutions only on their novelty with respect to earlier samples, and then assessing whether or not a solution was found at the end of the process. Similarly, the mixing genetic algorithm (MGA)[55], tailored to travelling salesman problem applications, does not use selection, allowing it to preserve diversity and avoid premature convergence.

Although neutrality rarely emerges endogenously in EC, it is surprisingly common in human-produced software. For example, random mutations of source code are neutral with respect to tested functionality as much as 30–40% of the time, even when mutations are restricted to regions of code executed by the fitness function[56–58]. These results are informative for algorithms that repair software automatically[59]. Similarly, the GEVO code optimizer uses neutral mutations of LLVM code to improve execution times[60].

Random drift provides raw material for evolutionary innovation under weak selection. Such selection allows populations to retain mutations that are neutral or even weakly deleterious, which in turn can seed innovations that increase fitness dramatically in subsequent generations, even though it may take a long time[61–64]. By contrast, most EC algorithms today use selection that is strong compared to biology. Neutral and weakly deleterious mutations are eliminated almost immediately, except for accidental hitchhikers, and the population tends to converge quickly on a local optimum. This design decision, driven by efficiency considerations, is a notable departure from many evolutionary processes.

Incorporating neutrality, weak selection and longer time scales is thus a promising opportunity for future EC research.

Multi-objectivity. Biological fitness is complex. Many competing elements determine the number of offspring that an individual ultimately produces. Spending resources to find food competes with attracting a mate and protecting the young. To survive and reproduce, individuals must perform adequately on all such dimensions. Thus, while fitness drives biological selection directly, the tradeoffs that determine reproductive success are indirect and multi-dimensional.

In some cases, EC systems have a single high-level computational objective, such as survival in an ecological simulation, winning or losing a game, or the amount of money earned in financial markets[65–67]. In these settings, evolution manages the necessary dimensions and tradeoffs among them implicitly. In many settings, however, high-level fitness provides too weak a signal for selection, and EC fails to make progress towards a solution.

Multi-objective ECs address this problem by operating one level below the high-level objective. They specify and measure the various dimensions directly, such as performance, cost, accuracy, solution complexity or appearance[68–70]. The solutions form Pareto fronts, that is, collections of different trade-offs, and mathematical techniques (such as NSGA-II) are used to find them[71]. Engineers can then use external criteria to select a particular set of trade-offs, in some cases combining hundreds or thousands of objectives that represent a large array of constraints or tests (for example,

NSGA-III[72], lexicase selection[73]). Although popular, such explicit multi-objectivity departs from biology: it is a tool for expressing principles of human design.

In biology, multidimensional fitness can facilitate speciation, that is, different strategies for thriving in a complex environment. Such environments contain multiple niches and different species without explicit multiple objectives[74,75]. Speciation is useful in EC as well, enhancing diversity and protecting innovation[34,76]. However, speciation does not arise in EC endogeneously without supporting mechanisms. A step in this direction is quality diversity, where novelty is included as a secondary objective. The approach rewards diversity explicitly, and useful niches and stepping stones are more likely to be discovered without prescribing them as explicit objectives[77,78].

Thus, artificial multi-objectivity can serve as a proxy for complex environments and niches that emerge through time. There is an opportunity, however, for EC to apply multi-objectivity more aggressively and implicitly by adopting more natural high-level fitness functions and encouraging niche formation.

Genotype-to-phenotype mappings. Most EC systems encode solutions directly in the genome. For instance, in function optimization, the genotype might represent parameter values, and for neural networks, the genotype might be a concatenation of connection weights. Mutation and recombination can thus modify the components of the solution directly. This approach succeeds in many problems, and it is easy to understand and visualize.

Such direct encodings do not scale to biological levels of complexity, and in biology the genotype-to-phenotype mapping is much more complex and indirect. Epistasis is common, and gene expression is modulated, for example, by genetic regulatory networks[79] and by epigenetic traits[80]. Developmental pathways are modulated by interactions among cells and between the cells and the environment[81], providing another level of indirection. Selection operates at the phenotypic level, which is constructed in an interactive, multifaceted process with high degrees of freedom (DoF).

In EC, epistasis manifests as nonlinear interactions among genes, and the recombination operator allows successful gene combinations to be preserved in successive generations. Simple genetic regulatory networks have led to evolvable solutions, that is, those that can adapt rapidly to environmental changes[82–84]. Several types of epigenetic mechanism have been proposed, leading to transgenerational effects similar to those seen in nature[85]. Many projects have investigated generative and developmental approaches, that is, those that include a constructive or adaptive process operating on the genetic encoding. For instance, the structure of a neural network can be evolved, and its weights learned through gradient descent[86]; a neural network can be evolved to assign the weights of another neural network that actually performs the task (for example, HyperNEAT[87]); the genome may contain a set of grammatical rules that are applied sequentially to generate a solution (for example, cellular encoding or CE[88]); and a compressed representation of a deep neural network can be evolved and expanded systematically to form the full network (for example, Deep GA[89]). In GP, evaluations can be seen as a primitive form of development: programs are evaluated with an interpreter[90], or by compiling and executing them in a run-time environment[21].

Despite these advances, most of these systems use mappings that are unlike biology, or abstract away much of it. The implementations have few degrees of freedom, and the epistatic interactions among the genes are highly constrained. Indirect encodings often do not improve performance over direct encodings[91–93]. Developing more successful indirect encodings will likely require thinking more carefully about their biological counterparts.

Co-evolution. Biological systems exist in dynamic environments where multiple species co-evolve simultaneously and both competition and cooperation are pervasive[94]. Some EC systems exhibit a similar dynamic, for example, in game playing, robot navigation and multi-agent problem solving. In such domains, co-evolution can be effective: two or more populations evolve simultaneously, and fitness is defined by how well the individuals compete or cooperate with individuals in the other population(s)[95].

Competitive co-evolution has rich dynamics that researchers have characterized in terms of the equilibria that can evolve[96,97]. This dynamic is particularly effective in game playing, where it establishes a natural progression of more challenging opponents—a method dating back to Samuel's Checker playing program in the 1950s[98] and used more recently in AlphaZero, for example[99]. Competitive co-evolution also applies to asymmetric situations such as co-evolving solutions and objectives[100], sorting networks and test cases[101], locomotion and obstacle courses (POET[102]), equations that explain progressively more experimental data while remaining simple (EUREQA[103]), and predator/prey systems that develop progressively complex behaviours[104]. More recently, generative adversarial network (GAN) architectures have adopted this strategy, whether trained through EC or gradient descent[105,106].

In cooperative co-evolution, multiple populations also evolve simultaneously, but the solution is formed by combining the populations[107]. For instance, CoDeepNEAT evolves modules and high-level blueprints of deep neural networks in multiple populations in parallel[23]. Similarly, subroutines can be evolved to form complete programs when combined[108]. Fitness is assessed on the combined system and propagates to the components—allowing evolution to discover components that work well together. This approach produces more complex solutions than would be possible if they were evolved directly.

Therefore, co-evolution is appealing because it establishes an openended dynamic that, in principle, can indefinitely discover progressively more complex solutions. Cooperative co-evolution can lead to more complex representations, and competitive co-evolution can establish new challenges. In current EC, however, interactions between populations are highly constrained and scripted, and the relationships and representations do not change. With more computing resources and more flexible environments it may be possible to generalize this approach, and thus take better advantage of it.

Opportunities and challenges

Research in EC has matured in the nearly 60 years since these algorithms were first proposed, and today's EC at times demonstrates surprisingly creative and innovative problem solving. However, most observers would judge today's digitally evolved systems as lacking the sophistication, flexibility and plasticity of biological organisms and ecosystems. Evolved neural networks are still far from the complexity of their biological counterparts; evolved cyberdefence systems do not come close to the capabilities of the natural immune system; no evolved agents are capable of even simple geometric reasoning[109] that comes naturally to humans.

While evolutionary biology has progressed tremendously during these 60 years, the development of EC methods has followed its own trajectory, and often diverged from biology (as summarized in Table 1). In terms of openendedness, multi-objectivity and co-evolution, EC is on a promising path where scaling the computation, representations, and environments is likely to lead to systems that resemble biological complexity more closely. However, EC falls short in other areas: neutrality and random drift, using complex genotype-to-phenotype mappings with rich environmental interactions, and achieving major transitions in organization. Biological mechanisms may need to be further elucidated and incorporated into EC before we see major transitions evolving computationally. However, biology does suggest how the first two might be addressed.

First, regarding neutrality and random drift, EC researchers have studied computational mechanisms that promote diversity and

exploration[54,73,110], but they are ad hoc corrections to small populations operating under strong selection pressure. In a similar vein, many EC methods rely on mutation more than recombination[5,111], even though it is viewed as a key mechanism in biological evolution, preserving epistatic interactions among genes. By contrast, diversity and exploration in naturally evolving systems often originate from relatively weak selection on large populations. Even unlikely innovations, such as changes in organizational structure, can appear when the process runs long enough. These directions are underexplored in EC but constitute a major opportunity for future research.

Second, regarding genotype-to-phenotype mappings, EC methods today succeed when the human designer can identify a genetic encoding that maps directly to phenotypic behaviour. By contrast, biological evolution uses complex indirect mappings. To take advantage of this process in EC will require designs in which genomes do not necessarily contain all of the information required to construct a solution, and instead information emerges naturally through interactions with the environment[112,113].

Third, as discussed in 'Comparing EC with biological evolution', we have not yet seen major transitions in EC, that is, evolved representations at new organizational levels than what was provided in the initial encoding. In EC such innovations are currently achievable only by manually creating transition mechanisms[43], that is, if the different levels are prespecified, representations constructed for each of them, and evolution is allowed to proceed sequentially or in parallel at multiple levels. In such a setting, collaborative co-evolution can create cooperative structures that work together[23,69], and competitive co-evolution can dynamically produce new challenges at different levels[102,114]. But they do not drive actual transitions.

By contrast, major transitions in biology occurred by manipulating one underlying representation, the amino acid sequences[38–40]. The way that information is organized, transmitted between elements and individuals, and translated into physical structures and behaviours, changes fundamentally in these transitions. However, biological theory does not yet provide adequate guidance on how such transitions can be achieved mechanistically. Biological theory describes what these transitions are, but not how they happened, leaving many questions unanswered. Does selection operate at multiple levels? Is the high-level organization discovered gradually, or does it require multiple simultaneous innovations? Are there phases where multiple levels coexist, cooperate and compete? What mechanisms lead to component specialization and loss of independent replication? Is there a similar dynamic in each of these transitions or is it different in each case? This lack of guidance suggests that there may be gaps in evolutionary theory, which are important to fill not just for its own sake, but for EC as well.

If we can meet these challenges, EC is poised to become a core innovation engine for complex engineered systems. As EC improves, it may capture enough relevant biology to serve as a useful model of biological evolution. The enormous computational power available today allows us to simulate large populations and deep time, thereby testing theories about biological evolution in a way that has not been possible until now.

Conclusion

Over half a century of EC has shown the promise in harnessing the principles of biological evolution. Modern evolutionary theory, coupled with a millionfold increase in computational power, offers an opportunity to simulate evolutionary mechanisms at a new scale with increased fidelity. Of the six essential dimensions discussed in this paper, EC has a good start on openendedness, multi-objectivity and co-evolution, although more can be achieved by scaling up. Neutrality and genotype-to-phenotype mappings are opportunities where existing biological insights could be utilized to produce dramatic improvements. By contrast, major transitions may not emerge in EC until they are better understood in biology and computational approaches are adjusted accordingly. With such progress, EC can play a leading role in machine creativity, support breakthroughs in engineering and provide insight into evolutionary theory.

Received: 15 September 2020; Accepted: 1 December 2020;
Published online: 18 January 2021

References

1. Holland, J. H. Outline for a logical theory of adaptive systems. *J. ACM* **9**, 297–314 (1962).
2. Holland, J. H. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence* (Univ. Michigan Press, 1975).
3. Forrest, S. Genetic algorithms: principles of natural selection applied to computation. *Science* **261**, 872–878 (1993).
4. Koza, J. R. *Genetic Programming* (MIT Press, 1992).
5. Beyer, H.-G. & Schwefel, H.-P. Evolution strategies: a comprehensive introduction. *Nat. Comput.* **1**, 3–52 (2002).
6. Dasgupta, D. & Michalewicz, Z. (eds). *Evolutionary Algorithms in Engineering Applications* (Springer, 1997).
7. Greiner, D., Periaux, J., Quagliarella, D., Magalhaes-Mendes, J. & Galván, B. Evolutionary algorithms and metaheuristics: applications in engineering design and optimization. *Math. Probl. Eng.* **2018**, 2793762 (2018).
8. Miettinen, K. & Neittaanmaki, P. *Evolutionary Algorithms in Engineering and Computer Science: Recent Advances in Genetic Algorithms, Evolution Strategies, Evolutionary Programming, GE* (Wiley, 1999).
9. Parmee, I. C. *Evolutionary and Adaptive Computing in Engineering Design* (Springer, 2001).
10. Johnson, A. J. et al. Flavor-cyber-agriculture: optimization of plant metabolites in an open-source control environment through surrogate modeling. *PLoS ONE* **14**, e0213918 (2019).
11. Ling, S. H. & Lam, H. K. Evolutionary algorithms in health technologies. *Algorithms* **12**, 202 (2019).
12. Wang, H., Jin, Y. & Jansen, J. Data-driven surrogate-assisted multi-objective evolutionary optimization of a trauma system. *IEEE Trans. Evol. Comput.* **20**, 939–952 (2016).
13. Bongard, J. Evolutionary robotics. *Commun. ACM* **56**, 74–85 (2013).
14. Cheney, N., Bongard, J., SunSpiral, V. & Lipson, H. Scalable co-optimization of morphology and control in embodied machines. *J. R. Soc. Interface* **15**, 20170937 (2018).
15. Hornby, G. S., Lohn, J. D. & Linden, D. S. Computer-automated evolution of an X-band antenna for NASA's space technology 5 mission. *Evol. Comput.* **19**, 1–23 (2011).
16. van Eck Conradie, A., Miikkulainen, R. & Aldrich, C. Adaptive control utilising neural swarming. In *Proceedings of the Genetic and Evolutionary Computation Conference* 60–67 (2002).
17. Lehman, J. et al. The surprising creativity of digital evolution: a collection of anecdotes from the evolutionary computation and artificial life research communities. *Artif. Life* **26**, 274–306 (2020).
18. Miranda, E. R. & Biles, J. A. (eds) *Evolutionary Computer Music* (Springer, 2006).
19. Romero, J. & Machado, P. (eds) *The Art of Artificial Evolution: A Handbook on Evolutionary Art and Music* (Springer, 2007).
20. Secretan, J. et al. Picbreeder: a case study in collaborative evolutionary exploration of design space. *Evol. Comput.* **19**, 345–371 (2011).
21. Le Goues, C., Nguyen, T., Forrest, S. & Weimer, W. GenProg: a generic method for automated software repair. *Trans. Software Eng.* **38**, 54–72 (2012).
22. Le Goues, C., Dewey-Vogt, M., Forrest, S. & Weimer, W. A systematic study of automated program repair: fixing 55 out of 105 bugs for $8 each. In *International Conference on Software Engineering* (IEEE, 2012).
23. Miikkulainen, R. et al. Evolving deep neural networks. In *Artificial Intelligence in the Age of Neural Networks and Brain Computing* (eds Morabito, C. F. et al.) Ch. 15, 293–312 (Elsevier, 2020).
24. Real, E., Aggarwal, A., Huang, Y. & Le, Q. V. Regularized evolution for image classifier architecture search. In *Proceedings of the AAAI Conference on Artificial Intelligence* 4780–4789 (AAAI, 2019).
25. Stephenson, M., Amarasinghe, S., Martin, M. & O'Reilly, U.-M. Meta optimization: improving compiler heuristics with machine learning. *SIGPLAN Not.* **38**, 77–90 (2003).
26. Schulte, E., Dorn, J., Harding, S., Forrest, S. & Weimer, W. Post-compiler software optimization for reducing energy. In *Architectural Support for Programming Languages and Operating Systems* 639–652 (2014).
27. Banzhaf, W. et al. From artificial evolution to computational evolution: a research agenda. *Nat. Rev. Genet.* **7**, 729–735 (2006).
28. Bedau, M. A. et al. Open problems in artificial life. *Artif. Life* **6**, 363–376 (2000).

29. Lynch, M. The frailty of adaptive hypotheses for the origins of organismal complexity. *Proc. Natl Acad. Sci. USA* **104**, 8597–8604 (2007).
30. Liow, L. H., Valen, L. & Stenseth, N. C. Red queen: from populations to taxa and communities. *Trends Ecol. Evol.* **26**, 349–358 (2011).
31. Banzhaf, W. et al. Defining and simulating open-ended novelty: requirements, guidelines, and challenges. *Theor. Biosci.* **135**, 131–161 (2016).
32. Stanley, K. O. Why open-endedness matters. *Artif. Life* **25**, 232–235 (2019).
33. Smith, E. & Morowitz, H. J. Universality in intermediary metabolism. *Proc. Natl Acad. Sci. USA* **101**, 13168–13173 (2004).
34. Stanley, K. O. & Miikkulainen, R. Evolving neural networks through augmenting topologies. *Evol. Comput.* **10**, 99–127 (2002).
35. Lenski, R. E., Ofria, C., Collier, T. C. & Adami, C. Genome complexity, robustness, and genetic interactions in digital organisms. *Nature* **400**, 661–664 (1999).
36. Ray, T. S. An approach to the synthesis of life. In *Artificial Life II* (eds Langton, C. G. et al.) 371–408 (Addison-Wesley, 1991).
37. Spector, L., Klein, J. & Feinstein, M. Division blocks and the open-ended evolution of development, form, and behavior. In *Proceedings of the Genetic and Evolutionary Computation Conference* (2007).
38. Maynard Smith, J. & Szathmary, E. *The Major Transitions in Evolution* (Oxford Univ. Press, 1997).
39. Szathmáry, E. Toward major evolutionary transitions theory 2.0. *Proc. Natl Acad. Sci. USA* **112**, 10104–10111 (2015).
40. West, S. A., Fisher, R. M., Gardner, A. & Kiers, E. T. Major evolutionary transitions in individuality. *Proc. Natl Acad. Sci. USA* **112**, 10112–10119 (2015).
41. Watson, R. A. & Pollack, J. B. A computational model of symbiotic composition in evolutionary transitions. *Biosystems* **69**, 187–209 (2003).
42. Turney, P. D. Symbiosis promotes fitness improvements in the game of life. *Artif. Life* **26**, 338–365 (2020).
43. Lessin, D., Fussell, D. & Miikkulainen, R. Open-ended behavioral complexity for evolved virtual creatures. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)* (2013).
44. Bershtein, S., Segal, M., Bekerman, R., Tokuriki, N. & Tawfik, D. Robustness-epistasis link shapes the fitness landscape of a randomly drifting protein. *Nature* **444**, 929–932 (2006).
45. Kimura, M. *The Neutral Theory of Molecular Evolution* (Cambridge Univ. Press, 1985).
46. Wagner, A. et al. *Robustness and Evolvability in Living Systems* (Princeton Univ. Press, 2005).
47. Draghi, J., Parsons, T., Wagner, G. & Plotkin, J. Mutational robustness can facilitate adaptation. *Nature* **463**, 353–355 (2010).
48. LaBar, T. & Adami, C. Different evolutionary paths to complexity for small and large populations of digital organisms. *PLoS Comput. Biol.* **12**, e1005066 (2016).
49. Banzhaf, W. & Leier, A. Evolution on neutral networks in genetic programming. In *Genetic Programming Theory and Practice III* (eds Yu, T. et al.) 207–221 (Springer, 2006).
50. Milano, N. & Nolfi, S. Robustness to faults promotes evolvability: insights from evolving digital circuits. *PLoS ONE* **11**, e0158627 (2016).
51. Smith, T., Husbands, P. & O'Shea, M. Neutral networks and evolvability with complex genotype-phenotype mapping. In *Advances in Artificial Life* 272–281 (2001).
52. Spector, L. & Robinson, A. Genetic programming and autoconstructive evolution with the push programming language. *Genet. Program. Evolvable Mach.* **3**, 7–40 (2002).
53. Yu, T. & Miller, J. F. Through the interaction of neutral and adaptive mutations, evolutionary search finds a way. *Artif. Life* **12**, 525–551 (2006).
54. Stanley, K. O. & Lehman, J. *Why Greatness Cannot Be Planned: The Myth of the Objective* (Springer, 2015).
55. Varadarajan, S. & Whitley, D. The massively parallel mixing genetic algorithm for the traveling salesman problem. In *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO '19* 872–879 (ACM, 2019).
56. Harrand, N., Allier, S., Rodriguez-Cancio, M., Monperrus, M. & Baudry, B. A journey among Java neutral program variants. *Genet. Program. Evolvable Mach.* **20**, 531–580 (2019).
57. Schulte, E., Fry, Z. P., Fast, E., Weimer, W. & Forrest, S. Software mutational robustness. *Genet. Program. Evolvable Mach.* **15**, 281–312 (2014).
58. Veerapen, N., Daolio, F. & Ochoa, G. Modelling genetic improvement landscapes with local optima networks. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion* 1543–1548 (2017).
59. Renzullo, J., Weimer, W., Moses, M., and Forrest, S. Neutrality and epistasis in program space. In *ICSE Genetic Improvement Workshop* (2018).
60. Liou, J.-Y., Wang, X., Forrest, S. & Wu, C.-J. GEVO: GPU code optimization using evolutionary computation. *ACM Trans. Archit. Code Optimiz.* **17**, 33 (2020).
61. Cowperthwaite, M. C., Bull, J. J. & Meyers, L. A. From bad to good: fitness reversals and the ascent of deleterious mutations. *PLoS Comput. Biol.* **2**, e141 (2006).
62. LaBar, T. & Adami, C. Evolution of drift robustness in small populations. *Nat. Commun.* **8**, 1012 (2017).
63. Levin, B. R., Perrot, V. & Walker, N. Compensatory mutations, antibiotic resistance and the population genetics of adaptive evolution in bacteria. *Genetics* **154**, 985–997 (2000).
64. Moore, F. B.-G., Rozen, D. E. & Lenski, R. E. Pervasive compensatory adaptation in *Escherichia coli*. *Proc. R. Soc. Lon. B* **267**, 515–522 (2000).
65. Fogel, D. B. *Blondie24: Playing at the Edge of AI* (Kaufmann, 2001).
66. Grasm, R., Golestani, A., Hendry, A. P. & Cristescu, M. E. Speciation without pre-defined fitness functions. *PLoS ONE* **10**, e0137838 (2015).
67. Hu, Y. et al. Application of evolutionary computation for rule discovery in stock algorithmic trading: a literature review. *Appl. Soft Comput.* **36**, 534–551 (2015).
68. Coello Coello, C. A., Van Veldhuizen, D. A. & Lamont, G. B. *Evolutionary Algorithms for Solving Multi-Objective Problems* (Springer, 2007).
69. Liang, J., et al. Evolutionary neural AutoML for deep learning. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2019)* 401–409 (2019).
70. Schwaab, J. et al. Improving the performance of genetic algorithms for land-use allocation problems. *Int. J. Geogr. Inf. Sci.* **32**, 907–930 (2018).
71. Deb, K., Pratab, A., Agrawal, S. & Meyarivan, T. A fast and elitist multi-objective genetic algorithm: NSGA-II. *IEEE Trans. Evol. Comput.* **6**, 181–197 (2002).
72. Deb, K. & Jain, H. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: solving problems with box constraints. *IEEE Trans. Evol. Comput.* **18**, 577–601 (2014).
73. LaCava, W., Helmuth, T., Spector, L. & Moore, J. H. A probabilistic and multi-objective analysis of lexicase selection and ϵ -lexicase selection. *Evol. Comput.* **27**, 377–402 (2019).
74. Anceschi, N. et al. Neutral and niche forces as drivers of species selection. *J. Theor. Biol.* **483**, 109969 (2019).
75. Dieckmann, U. & Doebeli, M. On the origin of species by sympatric speciation. *Nature* **400**, 354–357 (1999).
76. Mahfoud, S. W. *Niching Methods for Genetic Algorithms*. PhD thesis, Univ. Illinois at Urbana-Champaign (1995).
77. Meyerson, E., Lehman, J. & Miikkulainen, R. Learning behavior characterizations for novelty search. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)* 149–156 (2016).
78. Pugh, J. K., Soros, L. B., Szerlip, P. A. & Stanley, K. O. Confronting the challenge of quality diversity. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation* 967–974 (2015).
79. Wang, Y. Gene regulatory networks. In *Encyclopedia of Systems Biology* (eds Dubitzky, W. et al.) 801–805 (Springer, 2013).
80. Lind, M. & Spagopoulou, F. Evolutionary consequences of epigenetic inheritance. *Heredity* **121**, 205–209 (2018).
81. Muller, G. B. Evo-devo Extending the evolutionary synthesis. *Nat. Rev. Genet.* **8**, 943–949 (2007).
82. Bentley, P. J. Evolving fractal gene regulatory networks for robot control. In *Advances in Artificial Life* (eds Banzhaf, W. et al.) 753–762 (Springer, 2003).
83. Payne, J. L., Moore, J. H. & Wagner, A. Robustness, evolvability, and the logic of genetic regulation. *Artif. Life* **20**, 111–126 (2014).
84. Reisinger, J. & Miikkulainen, R. Acquiring evolvability through adaptive representations. In *Proceeedings of the Genetic and Evolutionary Computation Conference* 1045–1052 (2007).
85. Wang, Q. et al. Epigenetic game theory: how to compute the epigenetic control of maternal-to-zygotic transition. *Phys. Life Rev.* **20**, 126–137 (2017).
86. Stanley, K. O., Clune, J., Lehman, J. & Miikkulainen, R. Designing neural networks through evolutionary algorithms. *Nat. Mach. Intell.* **1**, 24–35 (2019).
87. Stanley, K. O., D'Ambrosio, D. B. & Gauci, J. A hypercube-based encoding for evolving large-scale neural networks. *Artif. Life* **15**, 185–212 (2009).
88. Gruau, F. & Whitley, D. Adding learning to the cellular development of neural networks: evolution and the Baldwin effect. *Evol. Comput.* **1**, 213–233 (1993).
89. Such, F. P. et al. Deep neuroevolution: genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning. In *NeurIPS Deep Reinforcement Learning Workshop* (2017).
90. Banzhaf, W., Francone, F. D., Keller, R. E. & Nordin, P. *Genetic Programming: An Introduction: On the Automatic Evolution of Computer Programs and Its Applications* (Kaufmann, 1998).
91. Gomez, F., Schmidhuber, J. & Miikkulainen, R. Accelerated neural evolution through cooperatively coevolved synapses. *J. Mach. Learn. Res.* **9**, 937–965 (2008).
92. Helms, L. & Clune, J. Improving hybrid: how to best combine indirect and direct encoding in evolutionary algorithms. *PLoS ONE* **12**, e0174635 (2017).
93. Schrum, J., Gillespie, L. E. & Gonzalez, G. R. Comparing direct and indirect encodings using both raw and hand-designed features in tetris. In *Proceedings of the Genetic and Evolutionary Computation Conference* 179–186 (ACM, 2017).

94. Nuismer, S. *Introduction to Coevolutionary Theory* (Freeman, 2017).
95. Popovici, E., Bucci, A., Wiegand, P. & De Jong, E. In *Handbook of Natural Computing* (Rozenberg, G. et al.) 987–1033 (Springer, 2010).
96. de Jong, E. D. & Pollack, J. B. Ideal evaluation from coevolution. *Evol. Comput.* **12**, 159–192 (2004).
97. Ficici, S. G. & Pollack, J. B. Pareto optimality in coevolutionary learning. In *Sixth European Conference on Artificial Life* (ed. Kelemen, J.) 316–325 (Springer, 2001).
98. Samuel, A. In *Computers and Thought* (eds Feigenbaum, E. A. & Feldman, J. A.) 210–229 (McGraw-Hill, 1963).
99. Silver, D. et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science* **362**, 1140–1144 (2018).
100. Sipper, M., Moore, J. H. & Urbanowicz, R. J. In *Genetic Programming* (eds Sekanina, L. et al.) 146–161 (Springer, 2019).
101. Hillis, W. D. Co-evolving parasites improve simulated evolution as an optimization procedure. *Physica D* **42**, 228–234 (1990).
102. Wang, R., Lehman, J., Clune, J. & Stanley, K. O. POET: Open-ended coevolution of environments and their optimized solutions. In *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2019, Prague, Czech Republic, July 13-17, 2019* 142–151 (2019).
103. Schmidt, M. & Lipson, H. Distilling free-form natural laws from experimental data. *Science* **324**, 81–85 (2009).
104. Rawal, A., Rajagopalan, P. & Miikkulainen, R. Constructing competitive and cooperative agent behavior using coevolution. In *IEEE Conference on Computational Intelligence and Games (CIG 2010)* (2010).
105. Goodfellow, I. et al. Generative adversarial nets. In *Advances in Neural Information Processing Systems 27* (eds Ghahramani, Z. et al.) 2672–2680. (Curran Associates, 2014).
106. Wang, C., Xu, C., Yao, X. & Tao, D. Evolutionary generative adversarial networks. *IEEE Trans. Evol. Comput.* **23**, 921–934 (2019).
107. Potter, M. A. & Jong, K. A. D. Cooperative coevolution: an architecture for evolving coadapted subcomponents. *Evol. Comput.* **8**, 1–29 (2000).
108. Gerules, G. & Janikow, C. A survey of modularity in genetic programming. In *2016 IEEE Congress on Evolutionary Computation (CEC)* 5034–5043 (2016).
109. Chollet, F. On the measure of intelligence. Preprint at https://arxiv.org/abs/01547 (2019).
110. Goldberg, D. E. & Richardson, J. Genetic algorithms with sharing for multimodal function optimization. In *Proceedings of the Second International Conference on Genetic Algorithms* (1987).
111. Hansen, N. The CMA evolution strategy: a tutorial. Preprint at https://arxiv.org/abs/1604.00772 (2016).
112. Davidson, E. & Erwin, D. Gene regulatory networks and the evolution of animal body plans. *Science* **311**, 796–800 (2006).
113. Hendriks-Jansen, H. *Catching Ourselves in the Act. Situated Activity, Interactive Emergence, and Human Thought* (MIT Press, 1996).
114. Stanley, K. O. & Miikkulainen, R. Competitive coevolution through evolutionary complexification. *J. Artif. Intell. Res.* **21**, 63–100 (2004).

Acknowledgements

We thank R. Axelrod, D. Erwin, A. Graham and M. Mitchell for their suggestions on an earlier version of this paper, and R. Lenski and M. Lynch for many helpful discussions on evolution. Thanks to P. Reiter, F. Gomez and C. Schoolland for the original drawings for Fig. 1a–c. R.M. was partially supported by NSF DBI-0939454, DARPA FA8750-18-C-0103 and HR0011-18-2-0024, and NIH 1U01DC014922; S.F. was partially supported by NSF CCF-1908633 and IOS 2029696, DARPA FA8750-19C-0003 and N6600120C4020, and AFRL FA8750-19-1-0501; both R.M. and S.F. were supported by NSF IIS-2020103.

Competing interests

The authors declare no competing interests.

Additional information

Correspondence should be addressed to R.M. or S.F.

Peer review information *Nature Machine Intelligence* thanks Christoph Adami, Julian Miller and the other, anonymous, reviewer(s) for their contribution to the peer review of this work.