Genetic Programming and Evolvable Machines (2023) 24:22
https://doi.org/10.1007/s10710-023-09470-2

COMMENTARY

New directions in fitness evaluation: commentary on Langdon's JAWS30

Colin G. Johnson[1]

Accepted: 13 October 2023 / Published online: 22 November 2023

Langdon's paper emphasises the key role of fitness in GP, yet notes issues with current approaches to fitness: "In GP, as in most optimisation problems, most of the computation effort is spent on evaluating how good the proposed solutions are". The paper goes on to discuss a number of ways of tackling this bottleneck through the use of surrogate and learned fitness functions. In this commentary, I would like to suggest other future directions for fitness evaluation.

It is perhaps surprising that a combination of search and evaluation has proven effective across such a wide range of domains. Of course, fitness functions are domain specific, so fitness is not a universal, domain-independent metric. It may seem strange that we do not, typically, benefit from bespoke search approaches for particular problems. But—to borrow the terminology of Wigner [8] about the use of mathematics as a model of the physical world—this combination of a generic search technique and fitness evaluation is "unreasonably effective".

In this commentary, I would like to explore a number of future directions for fitness evaluation in GP and other evolutionary computation approaches. Firstly, let us continue the theme of abstraction. In a typical GP system, domain knowledge is included in two places: in the representation of solutions, and in the fitness function. Compared with other evolutionary systems, GP already abstracts some representational issues: in theory, the use of Turing-complete code as a representation provides a generic search space, without the need for problem-specific operators. Similar points can be made for other machine learning approaches, such as neural networks, which use a very flexible function representation to allow a vast amount of problems to be tackled using the same generic representation.

Thirtieth Anniversary of Genetic Programming: On the Programming of Computers by Means of Natural Selection.

This comment refers to the article available online at https://doi.org/10.1007/s10710-023-09467-x.

✉ Colin G. Johnson
Colin.Johnson@nottingham.ac.uk

[1] School of Computer Science, University of Nottingham, Nottingham, UK

Is there any scope for the development of a *universal* fitness function, one that could be applied with minimal domain knowledge? Li et al. [6]. argue that information-based metrics called *normalised information distances* have the potential to act as "universal distances". An example of such a distance is *normalised compression distance*, which consists of taking two datasets, and calculating how compressible they are separately and when combined. If the two datasets contain related data, then the combined set should compress to a smaller size than the two separate sets; if the data is unrelated, then the compressed size of the combined dataset will be similar to the sum of the two separate compressed datasets. This has been demonstrated to act as a metric for similarity across many domains: "genomics, virology, languages, literature, music, handwritten digits, astronomy, […], executables, Java programs" [6]. Could such a measure be used as the basis for as a generic fitness function across domains?

An alternative approach to the same aim is touched on in Langdon's paper: how to make "best use of previously gained knowledge" by learning fitness from examples. Given a problem, can we *learn* a fitness function from examples of the problem using so-called *autodidactic iteration* [1], where supervised learning is used to abstract an evaluation function from examples. This has been applied as a way of automatically creating a set of intermediate reward values in reinforcement learning [1] and has been combined with genetic search in an early attempt to generate a fitness function by learning [3].

Another approach might be to leverage the use of large language models as a fitness evaluator. This has particular value in areas where the fitness function has a more unstructured, linguistic, or subjective character. For example, a recent paper by Sawicki et al. [7] uses a fine-tuned version GPT-3.5 as a means of evaluating the style of text—in particular, how closely a particular piece of poetic text is to that of a particular author. Goes et al. [2] uses a similar approach to rank the quality of jokes. This is an interesting direction—whilst LLMs have been used extensively to generate material in a particular style, these new approaches demonstrate that they can also be used as evaluators, returning a ranking or score in response to a prompt. Again, the issues of evaluation time highlighted in Langdon's paper are potentially an issue with the use of LLMs for fitness evaluation, but it would be interesting to compare LLM-based methods against traditional methods of fitness computation.

As a final reflection, we might ask whether the traditional concept of the fitness function will continue to be used in its current form in GP—and more broadly, whether there are alternatives to the the idea of fitness/loss/error/objective functions in learning and optimisation. Krawiec [4] has argued that the idea of fitness functions needs to be generalised into the idea of *search drivers*–"'partial, heuristic, transient pseudo-objectives that form multifaceted characterizations of candidate solutions" [5]. Can we move away from a single idea of a fitness function measuring all aspects of a problem at once, to a more dynamically constructed idea of evaluation throughout a run?

In particular, one promising future direction would be to pay more attention to the *relevance* of a program fragment to the problem being tackled, rather than evaluating whole programs. If we could discover ways of evaluating fragments in isolation, rather than always evaluating whole solutions, this would address the issue of

"computation effort" that Langdon emphasises as a key problem, and also exploit the "embarrassingly parallel" nature of GP—not just evaluating vast numbers of programs in parallel for their fitness, but evaluating vast numbers of program fragments in parallel for their relevance to the problem. Again, ideas from information theory may offer a way of evaluating this *problem relevance* of fragments of code.

Funding Open Access funding enabled and organized by CAUL and its Member Institutions.

References

1. F. Agostinelli, S. McAleer, A. Shmakov, P. Baldi, Solving the Rubik's cube with deep reinforcement learning and search. Nature Mach. Intel. **1**, 356–363 (2019). https://doi.org/10.1038/s42256-019-0070-z
2. F. Goes et al., Is GPT-4 good enough to evaluate jokes? In *14th International Conference for Computational Creativity*, (Waterloo, Canada, 2023)
3. C.G. Johnson, Solving the Rubik's cube with stepwise deep learning. Expert. Syst. **38**(3), e12665 (2021). https://doi.org/10.1111/exsy.12665
4. Krzysztof Krawiec, *Behavioural Program Synthesis with Genetic Programming*, (Springer, 2016)
5. K. Krawiec, Opening the black box: alternative search drivers for genetic programming and test-based problems. MENDEL **23**(1), 1–6 (2017). https://doi.org/10.13164/mendel.2017.1.001
6. M. Li, X. Chen, X. Li, B. Ma, P. Vitanyi, The similarity metric. IEEE Trans. Inf. Theory **50**(12), 3250–3264 (2004). https://doi.org/10.1109/TIT.2004.838101
7. P. Sawicki et al., On the power of special-purpose GPT models to create and evaluate new poetry in old styles, in *International Conference on Computational Creativity*, (2023)
8. E.P. Wigner, The unreasonable effectiveness of mathematics in the natural sciences. Commun. Pure Appl. Math. **13**, 1–14 (1960)