Genetic Improvement @ ICSE 2020

William B. Langdon

Westley Weimer
University of Michigan, USA

Justyna Petke
University College, London, UK

Erik Fredericks
Oakland University, USA

Seongmin Lee
KAIST, Korea

Emily Winter
Lancaster University, UK

Michail Basios
TurinTech, London, UK

Myra B. Cohen
Iowa State University, USA

Aymeric Blot
University College, London, UK

Markus Wagner
Adelaide University, Australia

Bobby R. Bruce
UC, Davis, USA

Shin Yoo
KAIST, Korea

Simos Gerasimou
University of York, UK

Oliver Krauss
AIST, Austria

Yu Huang
University of Michigan, USA

Michael Gerten
Iowa State University, USA

DOI: 10.1145/3417564.3417575
https://doi.org/10.1145/3417564.3417575

Stand on the Shoulders of Giants and Evolve

Abstract

Following Prof. Mark Harman of Facebook's keynote and formal presentations (which are recorded in the proceedings) there was a wide ranging discussion at the eighth international Genetic Improvement workshop, GI-2020 @ ICSE (held as part of the International Conference on Software Engineering on Friday 3rd July 2020). Topics included industry take up, human factors, explainabiloity (explainability, justifyability, exploitability) and GI benchmarks. We also contrast various recent online approaches (e.g. SBST 2020) to holding virtual computer science conferences and workshops via the WWW on the Internet without face to face interaction. Finally we speculate on how the Coronavirus Covid-19 Pandemic will affect research next year and into the future.

1 What is Genetic Improvement

Genetic Improvement is a branch of Artificial Intelligence (AI) and Software Engineering which applies optimisation to improve existing programs. It is always possible to compare the new code with the existing code (effectively treating the program as its own specification) allowing GI to make measureable improvements to today's software. Improvements may be functional (e.g., does the new code have fewer bugs? does it have a new feature? does it give more accurate answers?) or non-functional (e.g. does it have better battery life? is it more reliable?)

2 GI @ ICSE 2020 via Zoom

The eighth Genetic Improvement workshop (GI 2020) was held as part of the forty-second International Conference on Software Engineering (ICSE 2020) during the Corona virus Covid-19 pandemic. Even as late as the close of ICSE workshop submissions, it was intended to hold ICSE in Seoul, the capital of South Korea, from May 23 thru May 29, 2020. However as the Pandemic bit, it became clear that May 2020 was not feasible. Initially it was decided to delay the conference (and hence the workshops) until October 2020. This was later overturned as more experience with virtualising conferences and holding them on the Internet was gained. Hence, the 42nd ICSE was held as a virtual electronic conference in early July 2020. Some events were cancelled. However it was decided that the GI @ ICSE 2020 workshop would became an Internet only event and was held using Zoom.us on Friday 3 July 2020 13:00-16:20 UTC `http://geneticimprovementofsoftware.com/gi2020icse.html` (Notice the Corvid imposed importance of the time zone.)

Fifty people pre-registered for the Genetic Improvement workshop. On the day participation varied with about 35 people "attending" via Zoom at any one time and a further 40 or so watching on a live YouTube channel (recording `https://youtu.be/GsNKCifm44A`). Also Yu Huang @YuHuang_yh ensured that Twitter carried highlights on `#gi_icse_2020`. As far as possible, the workshop kept the traditional (physical) format, starting with an invited keynote given by Prof. Mark Harman, who described the use of SBSE [1] and Genetic Improvement [2][3][4][5][6][7][8][9][10][11], within Facebook and future plans, including social testing [12] and Facebook calls for research proposals. This was followed by formal (albeit electronic only) presentations of papers, which are to be published in the ACM digital library [12, 13, 14, 15, 16, 17]. (The keynote and all of the presentations were recorded and are available via YouTube `https://youtu.be/GsNKCifm44A`). The formal presentations were followed by free-form discussions across seventeen time zones (again recorded).

In addition to Facebook, industrial particpants included TurinTech (London UK) and GrammaTech (Bethesda MD and Ithaca NY, USA). Academic and student particpants came from universities across five continents. (An edited record, including genetic improvement tools and resources, of last year's workshop was published in the ACM SIGSOFT's Software Engineering Notes [18] therefore we do not repeat that information here.)

Although the full discussion, as recorded via Bobby Bruce's screen, is available on YouTube, the following sections condense more than an hour's flowing discussion into just a few of the topics covered.

3 Human Factors and GI

There are now several implementations of genetic improvement in use [19, 20, 21, 22] (e.g. Janus Manager [23, 24] and Facebook's SapFix [25, 26, 27]) so it would be the right time to perform human studies to gain insights from developers who have already used GI on what else would be useful. Notice all genetic improvement systems in use require human acceptance of automatically generated source code changes. Indeed in both Janus Manager (Python) and SapFix (Hack, Java) developer acceptance is explicitly part of the continuous integration development protocol.

There is a great need to understand in more depth the human factors surrounding the application and implementation of genetic improvement techniques within industry.

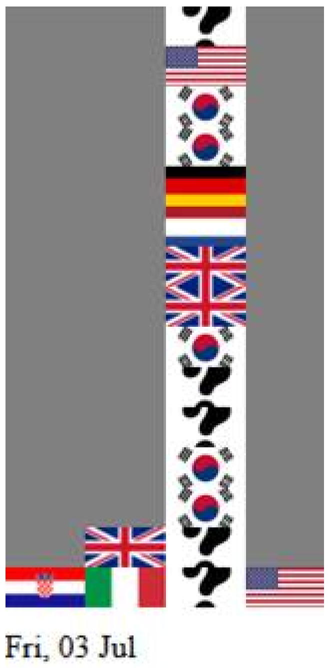

Figure 1: Surge in downloads via the GP bibliography during the GI workshop (14:00–17:21 BST). (Each day divided into four 6 hour periods.)

This should go beyond the consideration of the usability of GI techniques to also examine how such techniques might affect software developers' workflow and work satisfaction. A broader focus would enable a richer understanding of the potential barriers to GI techniques' adoption in industry, as well as the opportunities presented.

Concentrating on usability (often within the context of a controlled experiment) may also obscure the applicability of genetic improvement techniques in real-life settings, and what kind of specific techniques software developers would find useful in their work. However, usefulness alone is not enough and it must also be complemented with trustworthiness, since developers need to be able to trust GI techniques in order to use them with confidence.

It is also vital to understand the broader impact of genetic improvement from a human perspective, such as considering what kind of tasks GI techniques might both remove from and add to software developers' workflow and what this might mean for the future of software engineering work.

4 Explainabiloity, Explain, Justify, Exploit

Explain, justify and exploit in GI go hand in hand. If how the algorithm works, and the resulting patches can be explained, their use can be justified to a larger community and to industry at large. This in turn increases adoption of GI end opens it for exploitation.

4.1 Explainable Genetic Improvement

A very engaging discussion on explainable GI ensued during the workshop. Explainability is an important topic in the Artificial Intelligence (AI) literature now [28]. It provides users with the reasoning about decisions made during the machine learning, search and optimisation process. It was suggested, that if we can provide explainability along with our patches, this would go a long way in helping users gain confidence in the proposed program modifications. This fits in well with our earlier discussion of trust on the human side of GI. Some argued that GI systems have *explainability built in* as the algorithms explore parts of a search space and that it should be possible to capture some of the reasoning (why some patches are selected or not) during the search. The discussion centered on two aspects of explainability: structural and semantic. Attendees felt that the semantic explainability is harder and it would not be trivial to build in. Two potential issues are that (1) we are guiding the search towards the positive part of the search space which could bias our explanations, and (2) since part of explainability is providing a minimally explainable set this could lead to overfitting. Both of these issues would need to be overcome. All agreed that research into explainability of GI is an interesting and exciting direction.

Through the discussion, we shared some research methodologies for explainable GI. One approach is to understand the GI process. We already know the goals of the GI objective (fitness) function and which code changes (mutations) have been applied to the program. Based on the record of the changes, we hope to be able to explain how the control flow of the program has changed and how performance (fitness) has changed after applying the change operators. While the approach of *understanding* the GI is a passive strategy to achieve explainable GI, we can also actively *control* what GI can do. We may use modification operators that are designed to explain their changes. Therefore, each change can provide more useful information about itself.

A more pragmatic approach could be to build on standard methods used for debugging, including control flow and data flow graphs, which would aid developers, if not to accept the patch then understand its impact on the code base. Perhaps even if the fix is not the right one, if developers understand its influence then they could still accept it as a temporary fix, whilst they think of a permanent one. Also, if we have a reasonable measure of readability, GI could use it as one of the objectives in a multi-objective search.

4.2 Justify

The discussion turned towards the need for popular, easy-replicable use cases of GI[1]. It was suggested that in general people do not understand genetic algorithms and that perhaps this makes it a challenge to justify use of GI to a broader audience.

During the discussion it was also noted that some form of metric could be introduced for automatically generated patches. Such a metric could also impact use of patches, similar to how upvotes work in forums such as Stack Overflow. In production environments, such as continuous integration (CI), we certainly see that managers ensure that effectiveness metrics, such as fraction of auto-patches accepted into production, are collected [25, 27].

4.3 Exploit

The question of exploiting GI was felt to be much the same as how to get uptake of GI by industry (see Sections 5 and 6).

5 Industry's View of GI

5.1 Artificial Intelligence is very popular across Industry

AI is one of the hottest topics in industry at the moment. Usually businesses that use the term AI refer to the technology that will help them automate their current processes (e.g., chat bots, automatic document text extraction, speech to text), better understand their data (machine learning clustering models) and take automatic prediction decisions (supervised learning models). Thus, there is a very big increase in hiring data scientists who can apply the newest machine learning techniques. Knowledge of machine learning libraries such as Keras, Tensorflow and Pytorch[2] has become essential knowledge in most interviews. Practically, in industry the term AI refers to machine learning related problems.

5.2 Machine Learning has dominated the AI space

The popularity of AI means that any problem that is in the optimisation space is tackled usually as a machine learning problem. For instance, a lot of industrial problems that have to do with parameter tuning for maximising or minimising an objective function, are treated like machine learning prediction problems, whilst they could be solved, many times, more efficiently using Genetic Algorithms and other optimisation techniques. However, few people in industry are familiar with terms such as genetic algorithms, multi-objective optimisation and genetic improvement. This may be due to limited industrial use so far or not being highly advertised or due to the lack of widely available and popular open source libraries that are used by current data scientists.

5.3 AI on data vs. GI on code adoption

Most AI techniques in industry are applied on data, whereas most GI techniques focus on code (recent GI exceptions include [32, 33]). In many organisations, having access to data

[1]Last year, in Section 3 [18], we listed 22 tools, also [11, 29, 30, 31] were published after the GI @ ICSE 2019 workshop.

[2]See `https://keras.io/`, `https://www.tensorflow.org/`, and `https://pytorch.org/`.

and exporting them for applying machine learning libraries is quite a straightforward and replicable process. However, applying GI techniques on existing internal code bases can be a bit more tricky because of the different programming languages used (different teams use different languages inside the same company), the different programming tools, and the variety of project code structures.

Additionally, many code bases lack proper testing and performance benchmarks, upon which many GI techniques rely on. For instance, when we wanted to apply Artemis [34], a GI tool that does automatic code optimisation using better data structures on a performance critical component of a system, we realised that the project did not contain a proper performance benchmark and relying on the test cases was not a realistic behaviour of the application, thus making the optimisations impractical. Additionally, building a performance benchmark for the specific project would take a long time, and thus was considered a lower priority on the project manager list of features.

To summarise, the most common issues when applying GI techniques on existing industrial code bases are:

1. Lack of proper testing and benchmarks on the code bases. Industrial applications are usually much more complex and with many dependencies than the applications used by the research community to apply GI.

2. Lack of popular, easy-replicable successful use cases where GI has been applied. Industry likes ready to use tools that can help them make money.

3. GI tools are mostly understood by only a few people inside an organisation, who are usually technical experts. Technical people have less access to budget and investment in new technologies.

4. Further need to integrate GI as part of the continuous integration process. If GI is inside a development tool, most engineers will not focus on the exact techniques used, and will not be familiar with the term GI.

6 Researchers and Industry

In addition to TurinTech (Sections 5–5.3), there were two other companies represented in the registered audience: Facebook and GrammaTech. The question of how GI researchers could work with them (or other companies) was raised. A Facebook call for research proposals was announced during the keynote. For examples, would industry accept partnership agreements? Perhaps these could lead to easy ways to try out a new technique (e.g., stack-based GI, presented by Dr. Blot [13]) on Facebook's SapFix[27]?

In order to develop GI techniques that are beneficial to industry, greater collaboration with industry may be needed. However, there are several potential barriers to this. Firstly, some companies may be wary of sharing code and documentation with researchers. (This can often be overcome with suitable non-disclosure agreements, NDAs.) Secondly, the time investment needed for industry to work with researchers might be off-putting. Several possible solutions were discussed, including clearly articulated partnership agreements (traditional university ethics procedures may not be necessarily adequate [35]) and adopting agile-inspired frameworks in order to carry out research in a timely and light-touch fashion [36].

Dr. Winter [17] spoke of the need for more longitudinal and ethnographic research in industry, so as to more fully understand the organisational and human context of GI techniques' use, and potential barriers to their adoption.

More immediately, a GI plug-in or simply better documentation will encourage way more widespread use of GI.

7 Benchmarks

A call for industry cooperation to create realistic genetic improvement benchmarks was raised. Some argued that benchmarks which have characteristics of real (industrial) applications are needed to allow researchers to build new techniques that will work in practice. If researchers use only small benchmarks, which are open source, their techniques may overfit and/or not scale to real systems. Another benefit of benchmarks is to provide an easy entry point for newcomers to GI. Some of the attendees from industry at the workshop pushed back saying that everyone already has access to realistic systems since many of the large open source applications today are representative of the types of systems in industry. However, they also noted that to make research techniques viable for industry they need to be built into existing continuous integration (CI) build systems. They suggested GitHub open source repository pull requests be provided for genetic improvement patches to find out if they are really good and useful. Some suggested that scalability, especially in data-driven systems, is still a real challenge for genetic improvement. Others argued that some GI tools scale well, but that this may be a factor of whether or not there is good fault (bug) localization behind it.

Benchmarking is a challenge that even the optimisation community continues to struggle with [37]. Indeed the fact that the ten-year impact award at GECCO 2020 was awarded to a paper on benchmarking organisation [38], highlights that the challenge is still unsolved. If real world systems taken from industry might provide realistic examples, they come with their inherent cost and complexity. Furthermore, when selecting representatives examples: which project to consider, which programming language to target, what feature in general to take into consideration, are recurrent questions in the genetic improvement community for which there is no clear consensus. One idea, in order to obtain much simpler and cheaper problems instances for GI approaches, would be to take a step back and from the theory point of view focus on the core components and characteristics of GI problems [14]. Such theory-anchored characterisations, while not exactly filling the need of actual complex real-world benchmarks, would have several advantages:

Table 1: Contrasting Approaches and Tool Usage between GI and SBST ICSE 2020 workshops

	Genetic Improvement	Search-Based Software Testing
Meeting Software	Zoom	Microsoft Teams
Livestream	YouTube	Twitch
Presentation Format	Live presentation	Live keynote + pre-recorded author videos

1. better rationalisation of what makes GI work,
2. minimal examples highlighting specific features for practitioners to improve the state-of-the-art on, and
3. an opportunity for people from more traditional optimisation fields to encounter GI problems and share their expertise.

Finally, to further strengthen the connection between the software engineering roots of GI and the knowledge of the optimisation community, GI might provide benchmarks for optimisation competitions (e.g., held at the CEC or GECCO conferences).

8 Next Year: What if we still have a Pandemic

8.1 Workshop Virtualization

A new issue that all international gatherings had to manage this year was remote interaction and content delivery for participants located in varying time zones. While ICSE2020 used a time band model, both GI @ ICSE 2020 and the Search-Based Software Testing (SBST 2020) workshops had single synchronous sessions using an online meeting tool and accompanying livestream. Table 1 summarises the two approaches.

In terms of "success", both workshops were enjoyable to attend and available technologies minimized problems during both events. The key difference between the two formats was that GI was more presentation-focused and SBST was more discussion-focused. For SBST, additional effort was placed on the chairs to ensure that there is no "dead air" (i.e., questions needed to be prepared in advance). Whereas GI was able to leverage the authors as well as the audience to fuel the interaction. SBST also provided a social event in the form of an enjoyable live pub quiz. Unlike other international events, the ICSE workshops were *not* under heavy preasure to remain within their scheduled time slot or under the threat of Zoom being closed whilst discussion continued.

Regardless of the differences, both GI and SBST were both successful in their own right. Both had a record number of registered participants. Monitoring attendance rates in the future will determine whether this is a result of lower attendance fees, being available via the world wide web (WWW), or both. However this initial metric makes a strong case for allowing virtual participation via the Internet in future.

8.2 Do What the Virtual World Does Best

Perhaps we should not try to replicate the conference experience but work out what is best done in the virtual world and more importantly how to connect via it to give faster discussion (rather than the current yearly or bi-annual cycle).

The use of Slack to continue conversations after scheduled talks/sessions had finished was something that worked well at other conferences and enabled more informal interaction.

References

[1] Mark Harman and Bryan F. Jones. Search based software engineering. *Information and Software Technology*, 43(14):833–839, December 2001. URL: `http://dx.doi.org/10.1016/S0950-5849(01)00189-6`.

[2] David R. White, Andrea Arcuri, and John A. Clark. Evolutionary improvement of programs. *IEEE Transactions on Evolutionary Computation*, 15(4):515–538, August 2011. URL: `http://dx.doi.org/10.1109/TEVC.2010.2083669`.

[3] B. J. Alexander and M. J. Gratton. Constructing an optimisation phase using grammatical evolution. In Andy Tyrrell, editor, *2009 IEEE Congress on Evolutionary Computation*, pages 1209–1216, Trondheim, Norway, 18-21 May 2009. IEEE Computational Intelligence Society, IEEE Press. URL: `http://dx.doi.org/10.1109/CEC.2009.4983083`.

[4] W. B. Langdon. Genetic improvement of programs. In Radomil Matousek, editor, *18th International Conference on Soft Computing, MENDEL 2012*, Brno, Czech Republic, 27-29 June 2012. Brno University of Technology. Invited keynote. URL: `http://www.cs.ucl.ac.uk/staff/W.Langdon/ftp/papers/Langdon_2012_mendel.pdf`.

[5] William B. Langdon. Genetically improved software. In Amir H. Gandomi, Amir H. Alavi, and Conor Ryan, editors, *Handbook of Genetic Programming Applications*, chapter 8, pages 181–220. Springer, 2015. URL: `http://dx.doi.org/10.1007/978-3-319-20883-1_8`.

[6] William B. Langdon and Mark Harman. Optimising existing software with genetic programming. *IEEE Transactions on Evolutionary Computation*, 19(1):118–135, February 2015. URL: `http://dx.doi.org/10.1109/TEVC.2013.2281544`.

[7] Justyna Petke, Mark Harman, William B. Langdon, and Westley Weimer. Using genetic improvement and code transplants to specialise a C++ program to a problem class. In Miguel Nicolau, Krzysztof Krawiec, Malcolm I. Heywood, Mauro Castelli, Pablo Garcia-Sanchez, Juan J. Merelo, Victor M. Rivas Santos, and Kevin Sim, editors, *17th European Conference on Genetic Programming*, volume 8599 of *LNCS*, pages 137–149, Granada, Spain, 23-25 April 2014. Springer. URL: `http://dx.doi.org/10.1007/978-3-662-44303-3_12`.

[8] Justyna Petke. Constraints: The future of combinatorial interaction testing. In *2015 IEEE/ACM 8th International Workshop on Search-Based Software Testing*, pages 17–18, Florence, May 2015. URL: `http://dx.doi.org/doi:10.1109/SBST.2015.11`.

[9] Justyna Petke, Mark Harman, William B. Langdon, and Westley Weimer. Specialising software for different downstream applications using genetic improvement and code transplantation. *IEEE Transactions on Software Engineering*, 44(6):574–594, June 2018. URL: `http://dx.doi.org/10.1109/TSE.2017.2702606`.

[10] Justyna Petke, Saemundur O. Haraldsson, Mark Harman, William B. Langdon, David R. White, and John R. Woodward. Genetic improvement of software: a comprehensive survey. *IEEE Transactions on Evolutionary Computation*, 22(3):415–432, June 2018. URL: `http://dx.doi.org/doi:10.1109/TEVC.2017.2693219`.

[11] Samundur O. Haraldsson, John R. Woodward, and Markus Wagner. Genetic improvement: Taking real-world source code and improving it using genetic programming. In Richard Allmendinger et al., editors, *Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion*, GECCO '20, page 801–831, internet, July 8-12 2020. Association for Computing Machinery. GI tutorial. URL: `http://dx.doi.org/10.1145/3377929.3389885`.

[12] John Ahlgren, Maria Eugenia Berezin, Kinga Bojarczuk, Elena Dulskyte, Inna Dvortsova, Johann George, Natalija Gucevska, Mark Harman, Ralf Laemmel, Erik Meijer, Silvia Sapora, and Justin Spahr-Summers. WES: Agent-based user interaction simulation on real infrastructure. In Shin Yoo, Justyna Petke, Westley Weimer, and Bobby R. Bruce, editors, *GI @ ICSE 2020*, pages 276–284, internet, 3 July 2020. ACM. Invited Keynote. URL: `http://dx.doi.org/10.1145/3387940.3392089`.

[13] Aymeric Blot and Justyna Petke. Stack-based genetic improvement. In Shin Yoo, Justyna Petke, Westley Weimer, and Bobby R. Bruce, editors, *GI @ ICSE 2020*, pages 289–290, internet, 3 July 2020. ACM. URL: `http://dx.doi.org/10.1145/3387940.3392174`.

[14] Aymeric Blot and Justyna Petke. Synthetic benchmarks for genetic improvement. In Shin Yoo, Justyna Petke, Westley Weimer, and Bobby R. Bruce, editors, *GI @ ICSE 2020*, pages 287–288, internet, 3 July 2020. ACM. URL: `http://dx.doi.org/10.1145/3387940.3392175`.

[15] Santanu Kumar Dash, Fan Wu, Michail Basios, Lingbo Li, and Leslie Kanthan. Checkers: Multi-modal darwinian API optimisation. In Shin Yoo, Justyna Petke, Westley Weimer, and Bobby R. Bruce, editors, *GI @ ICSE 2020*, pages 291–292, internet, 3 July 2020. ACM. URL: `http://dx.doi.org/10.1145/3387940.3392173`.

[16] Oliver Krauss, Hanspeter Moessenboeck, and Michael Affenzeller. Towards knowledge guided genetic improvement. In Shin Yoo, Justyna Petke, Westley Weimer, and Bobby R. Bruce, editors, *GI @ ICSE 2020*, pages 293–294, internet, 3 July 2020. ACM. URL: `http://dx.doi.org/10.1145/3387940.3392172`.

[17] Emily Winter, David Bowes, Steve Counsell, Tracy Hall, Saemundur Haraldsson, Vesna Nowack, and John Woodward. Human factors in the study of automatic software repair: Future directions for research with industry. In Shin Yoo, Justyna Petke, Westley Weimer, and Bobby R. Bruce, editors, *GI @ ICSE 2020*, pages 285–286, internet, 3 July 2020. ACM. URL: `http://dx.doi.org/10.1145/3387940.3392176`.

[18] William B. Langdon, Westley Weimer, Christopher Timperley, Oliver Krauss, Zhen Yu Ding, Yiwei Lyu, Nicolas Chausseau, Eric Schulte, Shin Hwei Tan, Kevin Leach, Yu Huang, and Gabin An. The state and future of genetic improvement. *SIGSOFT Software Engineering Notes*, 44(3):25–29, July 2019. URL: `http://dx.doi.org/10.1145/3356773.3356801`.

[19] William B. Langdon, Brian Yee Hong Lam, Justyna Petke, and Mark Harman. Improving CUDA DNA analysis software with genetic programming. In Sara Silva et al., editors, *GECCO '15: Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, pages 1063–1070, Madrid, 11-15 July 2015. ACM. URL: `http://dx.doi.org/10.1145/2739480.2754652`.

[20] William B. Langdon, Albert Vilella, Brian Yee Hong Lam, Justyna Petke, and Mark Harman. Benchmarking genetically improved BarraCUDA on epigenetic methylation NGS datasets and nVidia GPUs. In Justyna Petke, Westley Weimer, and David R. White, editors, *Genetic Improvement 2016 Workshop*, pages 1131–1132, Denver, July 20-24 2016. ACM. URL: `http://dx.doi.org/10.1145/2908961.2931687`.

[21] William B. Langdon, Brian Yee Hong Lam, Marc Modat, Justyna Petke, and Mark Harman. Genetic improvement of GPU software. *Genetic Programming and Evolvable Machines*, 18(1):5–44, March 2017. URL: `http://dx.doi.org/10.1007/s10710-016-9273-9`.

[22] William B. Langdon and Ronny Lorenz. Improving SSE parallel code with grow and graft genetic programming. In Justyna Petke, David R. White, W. B. Langdon, and Westley Weimer, editors, *GI-2017*, pages 1537–1538, Berlin, 15-19 July 2017. ACM. URL: `http://dx.doi.org/10.1145/3067695.3082524`.

[23] Saemundur O. Haraldsson, John R. Woodward, Alexander E. I. Brownlee, and Kristin Siggeirsdottir. Fixing bugs in your sleep: How genetic improvement became an overnight success. In Justyna Petke, David R. White, W. B. Langdon, and Westley Weimer, editors, *GI-2017*, pages 1513–1520, Berlin, 15-19 July 2017. ACM. Best paper. URL: `http://dx.doi.org/10.1145/3067695.3082517`.

[24] Saemundur Oskar Haraldsson. *Genetic Improvement of Software: From Program Landscapes to the Automatic Improvement of a Live System.* PhD thesis, Institute of Computing Science and Mathematics, University of Stirling, UK, May 2017. URL: `http://hdl.handle.net/1893/26007`.

[25] Nadia Alshahwan. Industrial experience of genetic improvement in Facebook. In Justyna Petke, Shin Hwei Tan, William B. Langdon, and Westley Weimer, editors, *GI-2019, ICSE workshops proceedings*, page 1, Montreal, 28 May 2019. IEEE. Invited Keynote. URL: `http://dx.doi.org/10.1109/GI.2019.00010`.

[26] Yue Jia, Ke Mao, and Mark Harman. Finding and fixing software bugs automatically with SapFix and Sapienz. Posted on Sep 13, 2018 to AI Research, Developer Tools, Open Source, Production Engineering, 13 September 2018. URL: `https://bit.ly/3hR2gpy`.

[27] Alexandru Marginean, Johannes Bader, Satish Chandra, Mark Harman, Yue Jia, Ke Mao, Alexander Mols, and Andrew Scott. SapFix: Automated end-to-end repair at scale. In Joanne M. Atlee and Tevfik Bultan, editors, *41st International Conference on Software Engineering*, pages 269–278, Montreal, 25-31 May 2019. ACM. URL: `http://dx.doi.org/10.1109/ICSE-SEIP.2019.00039`.

[28] David Gunning and David W. Aha. DARPA's explainable artificial intelligence (XAI) program. *AI Magazine*, 40(2):44–58, June 2019. URL: `http://dx.doi.org/10.1609/aimag.v40i2.2850`.

[29] Alexander E. I. Brownlee, Justyna Petke, Brad Alexander, Earl T. Barr, Markus Wagner, and David R. White. Gin: genetic improvement research made easy. In Manuel Lopez-Ibanez et al., editors, *GECCO '19: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 985–993, Prague, Czech Republic, 13-17 July 2019. ACM. URL: `http://dx.doi.org/10.1145/3321707.3321841`.

[30] Justyna Petke and Alexander Brownlee. Software improvement with Gin: a case study. In Shiva Nejati and Gregory Gay, editors, *SSBSE 2019*, volume 11664 of *LNCS*, pages 183–189, Tallinn, Estonia, 31 August - 1 September 2019. Springer. URL: `http://dx.doi.org/10.1007/978-3-030-27455-9_14`.

[31] Gabin An, Aymeric Blot, Justyna Petke, and Shin Yoo. PyGGI 2.0: Language independent genetic improvement framework. In Sven Apel and Alessandra Russo, editors, *Proceedings of the 27th Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering ESEC/FSE 2019)*, pages 1100–1104, Tallinn, Estonia, August 26–30 2019. ACM. URL: `http://dx.doi.org/10.1145/3338906.3341184`.

[32] William B. Langdon, Justyna Petke, and Ronny Lorenz. Evolving better RNAfold structure prediction. In Mauro Castelli, Lukas Sekanina, and Mengjie Zhang, editors, *EuroGP 2018: Proceedings of the 21st European Conference on Genetic Programming*, volume 10781 of *LNCS*, pages 220–236, Parma, Italy, 4-6 April 2018. Springer Verlag. URL: `http://dx.doi.org/10.1007/978-3-319-77553-1_14`.

[33] William B. Langdon and Justyna Petke. Evolving better software parameters. In Thelma Elita Colanzi and Phil McMinn, editors, *SSBSE 2018 Hot off the Press Track*, volume 11036 of *LNCS*, pages 363–369, Montpellier, France, 8-9 September 2018. Springer. URL: `http://dx.doi.org/10.1007/978-3-319-99241-9_22`.

[34] Michail Basios, Lingbo Li, Fan Wu, Leslie Kanthan, and Earl T. Barr. Darwinian data structure selection. In *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/FSE 2018*, pages 118–128, Lake Buena Vista, FL, USA, 2018. ACM. URL: `http://dx.doi.org/10.1145/3236024.3236043`.

[35] Robert Rybnicek and Roland Koenigsgruber. What makes industry–university collaboration succeed? a systematic review of the literature. *Journal of Business Economics*, 89:221–250, 2019. URL: `https://doi.org/10.1007/s11573-018-0916-6`.

[36] Helen Sharp, Laura Plonka, Katie Taylor, and Peggy Gregory. Overcoming challenges in collaboration between research and practice: The agile research network. In *Proceedings of the 1st International Workshop on Software Engineering Research and Industrial Practices, SER&IPs 2014*, pages 10–13, Hyderabad, India, 2014. Association for Computing Machinery. URL: `http://dx.doi.org/10.1145/2593850.2593859`.

[37] Thomas Bartz-Beielstein, Carola Doerr, Jakob Bossek, Sowmya Chandrasekaran, Tome Eftimov, Andreas Fischbach, Pascal Kerschke, Manuel Lopez-Ibanez, Katherine M. Malan, Jason H. Moore, Boris Naujoks, Patryk Orzechowski, Vanessa Volz, Markus Wagner, and Thomas Weise. Benchmarking in optimization: Best practice and open issues. ArXiv, 8 July 2020. URL: `https://arxiv.org/abs/2007.03488`, `arXiv:2007.03488`.

[38] Nikolaus Hansen, Anne Auger, Raymond Ros, Steffen Finck, and Petr Pošík. Comparing Results of 31 Algorithms from the Black-Box Optimization Benchmarking BBOB-2009. In Anne Auger, Hans-Georg Beyer, Nikolaus Hansen, Steffen Finck, Raymond Ros, and Petr Posik, editors, *Black box optimization benchmarking 2010 (BBOB 2010)*, pages 1689–1696, Portland, Oregon, USA, 7-11 July 2010. ACM. URL: `http://dx.doi.org/10.1145/1830761.1830790`.