The 13^{th} International Workshop on Genetic Improvement (GI @ ICSE 2024)

William B. Langdon
UCL, London, UK
w.langdon@cs.ucl.ac.uk

Gabin An
KAIST, Korea
agb94@kaist.ac.kr

Aymeric Blot
Universite de Rennes
aymeric.blot@univ-rennes.fr

Vesna Nowack
Imperial College, UK
vnowack@imperial.ac.uk

Justyna Petke
UCL, London, UK
j.petke@ucl.ac.uk

Shin Yoo
KAIST, Korea
shin.yoo@cs.kaist.ac.kr

Oliver Krauss
FH Upper Austria
oliver.krauss@fh-hagenberg.at

Erik M. Fredericks
GVSU, Michigan, USA
frederer@gvsu.edu

Daniel Blackwell
UCL, London, UK
daniel.blackwell.14@ucl.ac.uk

The 13^{th} International Workshop on Genetic Improvement (GI 2024) was co-located with the 46^{th} International Conference on Software Engineering (ICSE 2024) and ran in hybrid mode, physically being located with ICSE in Lisbon and being available worldwide using Zoom. Genetic improvement is the process of using automated search to improve existing software [1], [2]. It has successfully been used to fix bugs [3], transplant functionality from one system to another [4], improve predictions [5], and reduce software's runtime [6], [7], energy [8] and memory [9] consumption. GI research has already won five "Humies" [3], [10]–[13], prestigious cash prizes awarded for demonstrating human-competitive results at difficult-to-automate tasks. However, there remain many opportunities to improve the state-of-the-art. By bringing together GI researchers and GI enthusiasts, the workshop facilitates discussions and so we hope moves the field forward by sharing knowledge and exchanging ideas.

I. Workshop Format and Participation

The 13^{th} International Workshop on Genetic Improvement consisted of a one-day workshop and was held on Tuesday 16^{th} April 2024, the day before the main ICSE conference. The final workshop program and the recordings of the talks will be available online at http://geneticimprovementofsoftware.com/events/icse2024 and in the ICSE 2024 workshop proceedings [14]. The workshop featured a program that included a keynote, a GI tutorial, three research paper talks, three position paper talks and a discussion session.

Keynote.

Prof. Shin Yoo gave the invited keynote presentation. His enlightening talk "Executing One's Way out of the Chinese Room" [15] considered Artificial Intelligence (AI) Large Language Models (LLMs), which is growing more popular faster in software engineering than in other areas of computer science according to his analysis of arXiv data. He suspects that this popularity is due to the fact that LLMs can seemingly understand the semantics of both natural and programming languages, allowing LLMs to generate program source code given natural language requirements. However, Yoo questioned whether LLMs really understands the semantics of the code, using John Searle's "Chinese Room" thought experiment.

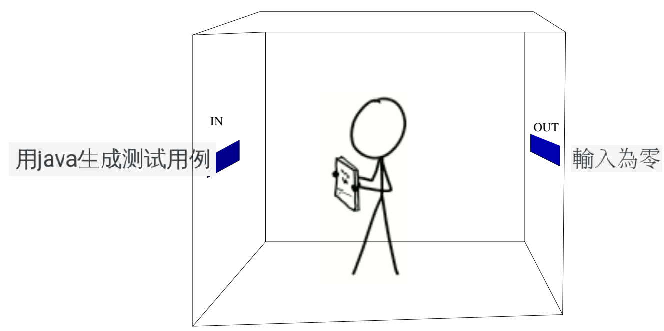

Fig. 1. The "Chinese Room" contains a person who does not speak Chinese. But they have rules to create Chinese output from Chinese input. Externally the whole room appears to understand Chinese. Similarly individual components of AI models do not understand Java but a whole AI LLM can generate Java source code or Java test cases.

Searle invented his "Chinese Room" as a philosophical argument about strong artificial intelligence (AI) [16]. Suppose someone who does not speak Chinese is in a room (Figure 1) with the source code of an AI program that can generate Chinese text in response to Chinese input. When someone outside posts queries written in Chinese into the room, the person inside follows the algorithm and prints the generated Chinese text and passes it to the person outside the room. If the AI program is indeed well written and quickly applied, the room appears to someone outside the room to understand Chinese. Yoo made the analogy with today's LLMs. They can take text input and generate text, such as Java source files and Java test cases [17], without essentially understanding the semantics of the generated code. The lack of understanding stems from the fact that the generated code is what is most likely (according to the distribution in the training corpus) rather than what is correct.

So what can be done? He claimed that program source code is a unique type of text for LLMs to work with, because

Fig. 2. Prof. Shin Yoo [15]

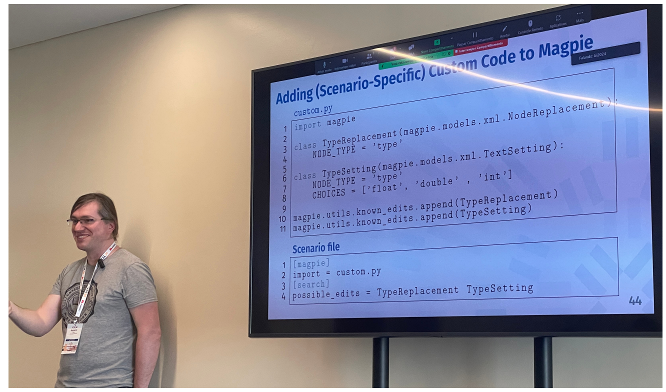

Fig. 3. Dr. Aymeric Blot [24]

Fig. 4. Benjamin J. Craine [28]

code is executable. That is, LLM generated source code output can be automatically compiled, run and tested. If it fails the tests, the faulty LLM output can be automatically rejected. The GI community, as well as the automated software testing community, has a lot of experience in automatically verifying program behaviour using dynamic executions, which is a natural fit to verification of LLM generated source code. Indeed, Yoo showed an example of an LLM agent instructed to reason about buggy code, generating testable scientific hypothesis about the bug and, subsequently generating debugger commands and test inputs to see if the hypothesis is true or not. If the hypothesis is false, this can lead to using the LLM to generate its next (testable) hypothesis; once the bug is located, the LLM agent can be used to generate a fix for the bug [18]–[22].

The keynote slides are available via http://gpbib.cs.ucl.ac.uk/gi2024/gi_2024_slides/yoo_gi2024_keynote.pdf

Tutorial.

Dr. Aymeric Blot (Figure 3) gave an extensive review of his Magpie GI system [23], [24], tracing its development from PyGGI 2.0 [25]. Magpie builds upon the capabilities of PyGGI, showcasing compatibility with any programming language and proficiency in enhancing both functional and non-functional aspects of software. However, Magpie also introduces novel features such as an improved user interface, the addition of parameter configuration to complement program source code manipulation, and the support of a much wider range of local search, genetic programming [26], and validation algorithms. Magpie is free and open source, accessible at https://github.com/bloa/magpie, and provides both a hack-friendly and a user-friendly interface to the world of automated software improvement. During the tutorial, participants gained insights into the framework's structure, philosophy, and key components, whilst also engaging with practical examples. Dr. Blot finished by describing future development plans. The tutorial slides are available via http://www.cs.ucl.ac.uk/staff/a.blot/files/blot_gi@icse_2024_slides.pdf

Paper presentations. This year, the GI workshop received seven paper submissions in total: three research paper submissions and four position paper submissions. Each received three independent reviews from the workshop's programme committee (Section V). One position paper was rejected and six papers were presented at the workshop. The authors of the accepted research papers had 20 minutes for the presentation and 10 for questions. The authors of the accepted position papers had 10 minutes for the presentation and 5 minutes for questions (e.g. Figures 4 and 5).

Participation The workshop attracted a total of 45 registrations, some of whom participated online via zoom (Figure 6 shows some of the in person and some of the zoom participants).

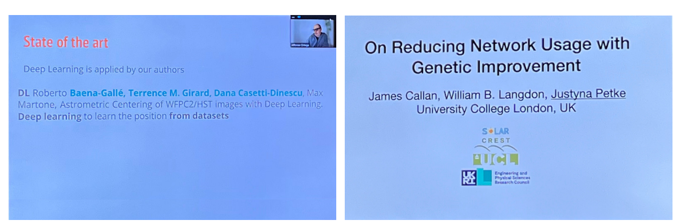

Fig. 5. Zoom presentations [27] [29]

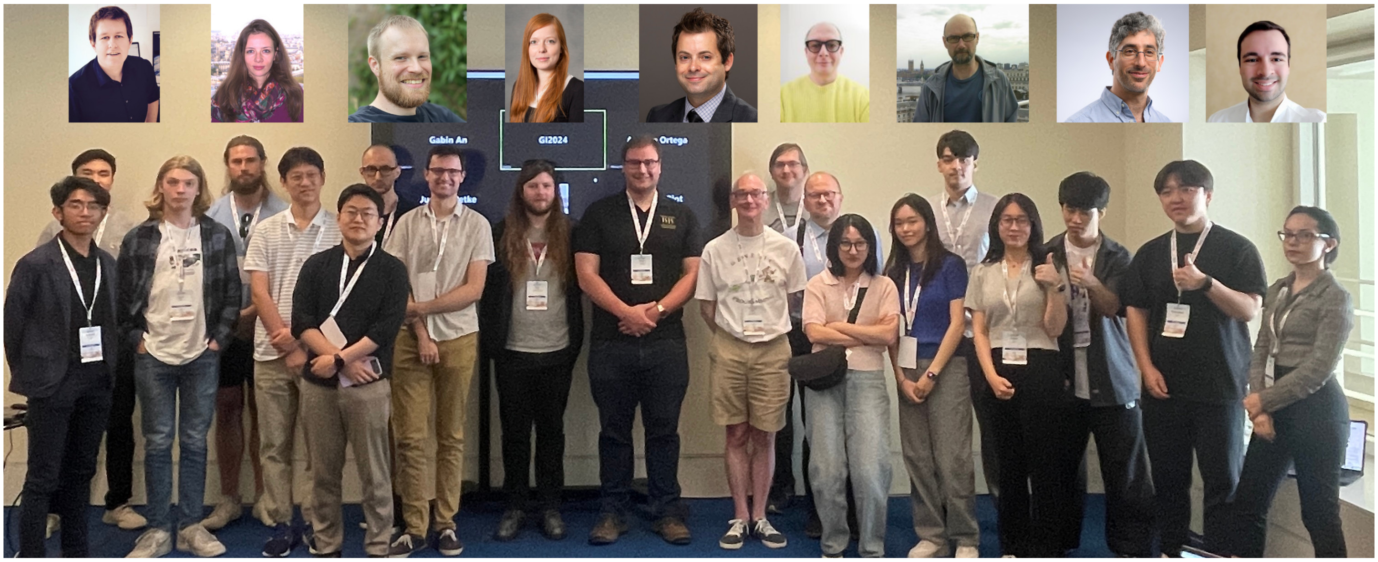

Fig. 6. Some of GI @ ICSE 2024 workshop participants. Top (via zoom): Dominik Sobania, Vesna Nowack, Oliver Krauss, Justyna Petke, Erik M. Fredericks, Alfonso Ortega de la Puente, David Clark, Achiya Elyasaf, Luigi Rovito. In Lisbon: Fathony Achmad, Louis Milliken, Dan Blackwell, Shin Yoo, Sungmin Kang, Max Hort, Benjamin J. Craine, Kevin Leach, W. B. Langdon, Aymeric Blot, Zsolt Nemeth, Juyeon Yoon, Somin Kim, Banseok Woo, Gabin An, Doam Lee, Hyeonseok Lee, Ilaria Pia La Torre

Fig. 7. Bill Langdon and Aymeric Blot. Best paper "Deep Mutations have Little Impact" [30].

Fig. 8. Zsolt Nemeth and Aymeric Blot. Best position paper "Ecosystem Curation in Genetic Improvement for Emergent Software Systems" [31].

Fig. 9. Kevin Leach and Aymeric Blot. Best presentation award for "Genetic Improvement for DNN Security" [32].

Awards. Traditionally at the GI workshop, the best paper awards are given to the researchers for their outstanding contributions to the GI field. This year, we granted three awards, where the best presentation award was decided by a vote from the participants of the workshop, while the other two were given based on reviews:

Best research paper award: "Deep Mutations have Little Impact" by William B. Langdon and David Clark [30] (Figure 7).

Best position paper award: "Ecosystem Curation in Genetic Improvement for Emergent Software Systems" by Zsolt Nemeth, Penn Faulkner Rainford and Barry Porter [31] (Figure 8).

Best presentation award: was won by Kevin Leach for "Genetic Improvement for DNN Security" [32] (Figure 9).

II. Discussion/Future Topics

A. Extension of Recent Work

A few of the authors present lobbied for people to extend their ideas. For example, Langdon [30] said he intended to investigate how much impact mutations have depending upon their depth in more C++ programs but said more examples, ideally in other languages, were needed and hoped others would consider reporting nesting depth when studying their own mutations [33]. Blackwell [34] suggested fuzz testing tools might help and some existing source code analysis tools, such as parsers might be an interesting source of deeply nested code benchmarks or targets for GI (see, for example, https://github.com/google/fuzzbench/tree/master/benchmarks). Also Blot reminded the audience that Magpie [24] is an open source project and he would welcome both collaborators wishing to extend Magpie and users of it. He also offered ready technical assistance to new users of Magpie.

B. Understandable Automatic Changes

There was an animated discussion about the need or otherwise for explainable GI (cf. Explainable AI) [35]–[39]. Initial work by Wes Weimer's group [40] suggested more than ten years ago that undergraduate students had a prejudice against automatically generated patches. However automatic program repair has moved on and it is now in routine use by professional software engineers in a few major software companies. For example, some automatically generated fixes within Meta are added into their continuous integration (CI) development system and so subject to review like other source code changes [41]–[46]. It is therefore essential they be comprehensible to human developers charged with maintaining the software. However, it was agreed that many of the published human studies on the acceptability of machine generated source code changes, had been carried out before the launch of ChatGPT in the fall of 2022, and so the now widespread knowledge of large language models (LLMs) might have changed software engineers' views on the use of artificially generated patches. Some argued that there may be scope for LLMs to generate natural language text (NLP) to explain the patch to the code reviewers. Should the LLM be tailored to the code reviewers? Could this tailoring be specific to company? E.g. should the text used in Meta be different from that generated for Bloomberg code reviewers [47]? It was pointed out that the following day (17 May) award winning work carried out at Bloomberg would be presented in the ICSE Software Engineering in Practice track [48]. Others asked about the current state-of-the-art: is there a limit to the size of code patches that professional software engineers are prepared to accept? Also, since the patch is written in the developer's language (e.g. Java), will patch explanations help?

Although some of the answers can be found in the human studies conducted in Bloomberg [47]–[49], we need to further understand how software engineers want to interact with LLM-based tools for code generation and what is needed for these tools to be widely adopted.

As noted by Erik Fredericks after Dr. Yoo's keynote, one possible direction could be to combine GitHub Copilot and a GI patching process.

Sungmin Kang pointed to existing open source communities aimed at using LLMs in software engineering, in particular https://github.com/OpenDevin/OpenDevin.

C. Genetic Improvement for Specification Repair

During the discussion, Vesna Nowack highlighted the possible application of GI in requirements engineering. Maintaining up-to-date specification is crucial for software development. However, due to the changes in the environment or user requirements, specification might become outdated or inaccurate, leading to misunderstandings in the development process or making it challenging to realise the intended system. Some proposed techniques [50], [51] successfully repair specification written in the Alloy declarative language. Similarly, being guided by the counterexamples generated by existing tools, GI could be applied to generate, refine and repair specification in Alloy or another language, such as Spectra [52].

D. Doing the Impossible

The previous section has already mentioned using genetic improvement to improve specifications and, although so far used only in testing [53], Prof. Yoo, in his keynote (Section I, see especially slides 38–40) proposed GI systems where an LLM makes mutations and a second LLM scores them as part of the GI fitness function. So allowing GI to operate on any type of software, not just programs.

In general, the first, the mutator, need not be an LLM. Any system which makes a reasonable percentage of "sensible" changes might be tried. For example, if the text file has a reasonable grammar, we might use Grammatical Evolution [54] to make changes via a BNF grammar. Alternatively, there are several more sophisticated grammars used with genetic programming which might be tried [55, page 53].

Although interactive evolutionary computation (IEC) [56] where one or more humans interact with the evolutionary computing system has been successfully used, particularly in art [57], [58] and games [59], it suffers from "user fatigue", which severely limits the number of fitness evaluations. However, the second LLM (the scorer), since it gives an automatic way of performing fitness evaluation, opens up many possibilities for GI. Note, the fitness function (e.g. the second LLM) need not be perfect. Evolutionary computing systems, such as GI, typically tolerate a lot of fitness noise, as long as on average the fitness function guides the search in the right direction. Often there are multiple ways, not just LLMs, to make stochastic changes. Potentially what they now give us, are ways to measure, or at least rank, automatically generated changes. Thus, we might consider applying GI to any part of software engineering where now LLMs give us at least a semi-automatic way of choosing on average the better part of a population of proposed mutations. We may hope overtime, as repeated generations of mutation and crossover pile beneficial mutations on top of other beneficial mutations, to solve problems previously considered impossible.

III. Workshop Outcomes

Following the success of last year's Genetic Improvement special issue of the Automated Software Engineering journal, some authors of accepted papers have been invited to submit their extended work to a second GI ASE special issue. (Special Issue Editors: Oliver Krauss, Vesna Nowack and Justyna Petke https://geneticimprovementofsoftware.com/events/ase2024.)

As with earlier workshops [60]–[63], there will be a short write up in the ACM SIGSOFT SEN newsletter (this document).

We hope to hold the GI workshop again next year.

IV. GI 2024 Workshop Organisers

Gabin An

Aymeric Blot

Vesna Nowack

Oliver Krauss

Justyna Petke

V. GI 2024 Programme Committee

Each submission received three independent reviews from the workshop's programme committee (see Figure 11).

In addition to providing feedback to the authors and deciding which submissions to accept, the best paper awards (Figures 7 and 8) were decided by the organisers using the reviewers' comments. Whilst the best presentation was chosen by the audience on the day in Lisbon and on Zoom.us (Figure 9).

Acknowledgement

We would like to thank: our ICSE 2024 student volunteer, photographers Somin Kim and Hyeonseok Lee and Sungmin Kang and Sisi Li.

Sponsored by A Field Guide to Genetic Programming [55].

Fig. 10. *Pasteis de Belem*'s pastel de nata were a great success.

References

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

[2] 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.

[3] Westley Weimer, ThanhVu Nguyen, Claire Le Goues, and Stephanie Forrest. Automatically finding patches using genetic programming. In Stephen Fickas, editor, *International Conference on Software Engineering (ICSE) 2009*, pages 364–374, Vancouver, May 16-24 2009. Winner ACM SIGSOFT Distinguished Paper Award. Gold medal at 2009 HUMIES. Ten-Year Most Influential Paper [65]. doi:10.1109/ICSE.2009.5070536.

[4] Alexandru Marginean, Earl T. Barr, Mark Harman, and Yue Jia. Automated transplantation of call graph and layout features into Kate. In Yvan Labiche and Marcio Barros, editors, *SSBSE*, volume 9275 of *LNCS*, pages 262–268, Bergamo, Italy, September 5-7 2015. Springer. doi:10.1007/978-3-319-22183-0_21.

[5] 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. doi:10.1007/978-3-319-77553-1_14.

[6] Andrea Arcuri, David Robert White, John Clark, and Xin Yao. Multi-objective improvement of software using co-evolution and smart seeding. In Xiaodong Li, Michael Kirley, Mengjie Zhang, David G. Green, Victor Ciesielski, Hussein A. Abbass, Zbigniew Michalewicz, Tim Hendtlass, Kalyanmoy Deb, Kay Chen Tan, Jürgen Branke, and Yuhui Shi, editors, *Proceedings of the 7th International Conference on Simulated Evolution And Learning (SEAL '08)*, volume 5361 of *Lecture Notes in Computer Science*, pages 61–70, Melbourne, Australia, December 7-10 2008. Springer. doi:10.1007/978-3-540-89694-4_7.

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

[8] Bobby R. Bruce. *The Blind Software Engineer: Improving the Non-Functional Properties of Software by Means of Genetic Improvement*. PhD thesis, Computer Science, University College, London, UK, 12 July 2018. URL: http://www.cs.ucl.ac.uk/staff/W.Langdon/ftp/papers/bruce_bobby_r_thesis.pdf.

[9] Jose L. Risco-Martin, J. Manuel Colmenar, J. Ignacio Hidalgo, Juan Lanchares, and Josefa Diaz. A methodology to automatically optimize dynamic memory managers applying grammatical evolution. *Journal of Systems and Software*, 91:109–123, 2014. doi:10.1016/j.jss.2013.12.044.

[10] 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. doi:10.1109/TSE.2017.2702606.

[11] Earl T. Barr, Mark Harman, Yue Jia, Alexandru Marginean, and Justyna Petke. Automated software transplantation. In Tao Xie and Michal Young, editors, *International Symposium on Software Testing and Analysis, ISSTA 2015*, pages 257–269, Baltimore, Maryland, USA, 14-17 July 2015. ACM. ACM SIGSOFT Distinguished Paper Award. doi:10.1145/2771783.2771796.

[12] Michail Basios, Lingbo Li, Fan Wu, Leslie Kanthan, and Earl T. Barr. Darwinian data structure selection. In Gary T. Leavens, Alessandro Garcia, and Corina S. Pasareanu, editors, *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, 4-9 Nov 2018. ACM. doi:10.1145/3236024.3236043.

[13] Joel Kuepper, Andres Erbsen, Jason Gross, Owen Conoly, Chuyue Sun, Samuel Tian, David Wu, Adam Chlipala, Chitchanok Chuengsatiansup, Daniel Genkin, Markus Wagner, and Yuval Yarom. CryptOpt: Verified compilation with randomized program search for cryptographic

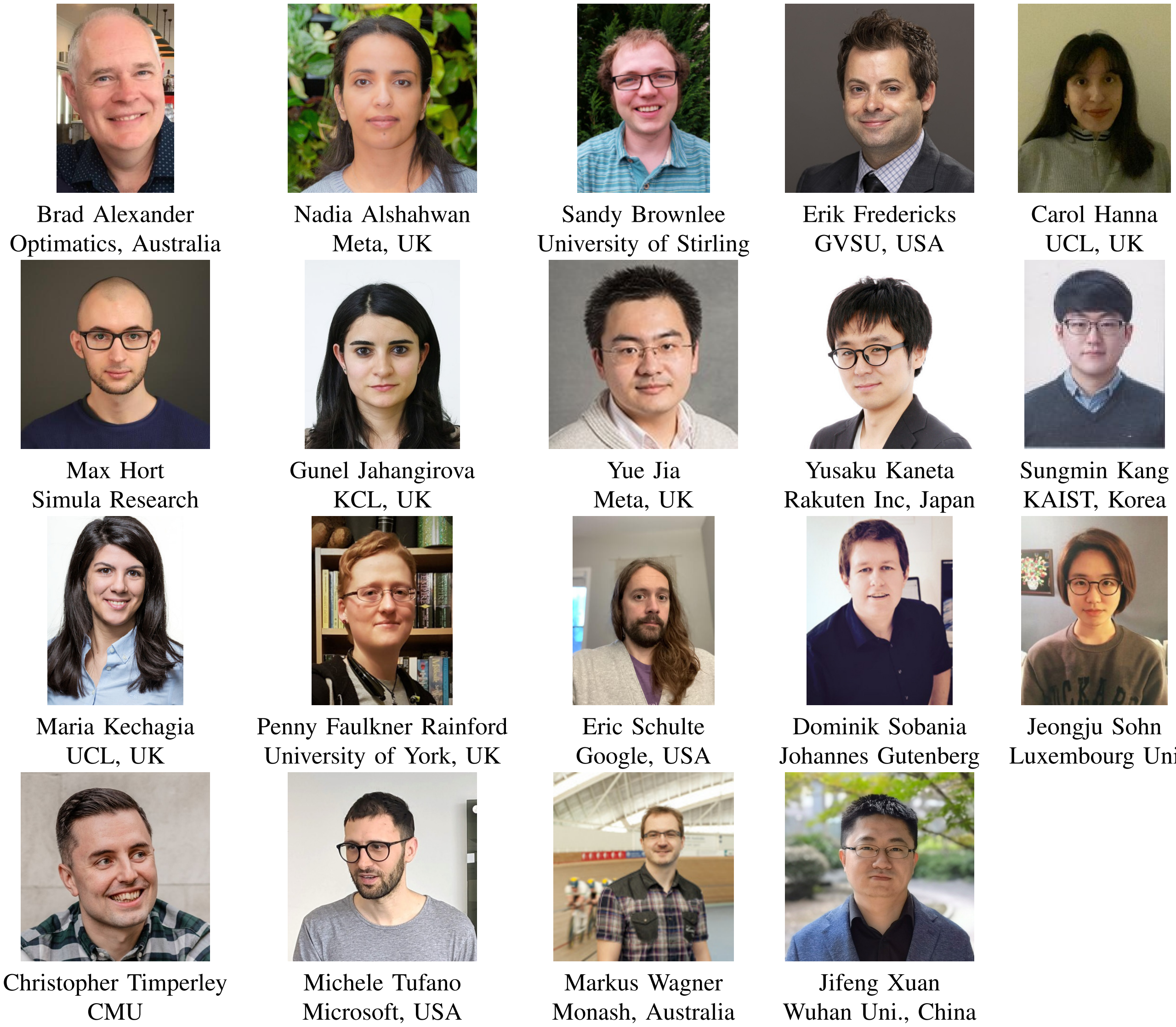

Fig. 11. GI @ ICSE 2024 Reviewers

primitives. In Nate Foster, editor, *44th ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI 2023*, page article no. 158, Orlando, Florida, 17-21 June 2023. Association for Computing Machinery. Gold winner 2023 HUMIES, PLDI Distinguished Paper. doi:10.1145/3591272.

[14] Gabin An, Aymeric Blot, Vesna Nowack, Oliver Krauss, and Justyna Petke, editors. *13th International Workshop on Genetic Improvement @ICSE 2024*, Lisbon, 16 April 2023. ACM. URL: http://gpbib.cs.ucl.ac.uk/gi2024/an_2024_GI.pdf.

[15] Shin Yoo. Executing one's way out of the chinese room. In Gabin An, Aymeric Blot, Vesna Nowack, Oliver Krauss, and Justyna Petke, editors, *13th International Workshop on Genetic Improvement @ICSE 2024*, page iv, Lisbon, 16 April 2024. ACM. Invited Keynote. URL: http://gpbib.cs.ucl.ac.uk/gi2024/an_2024_GI.pdf.

[16] John R. Searle. Minds, brains, and programs. *Behavioral and Brain Sciences*, 3(3):417—424, 1980. doi:10.1017/S0140525X00005756.

[17] Aidan Dakhama, Karine Even-Mendoza, W. B. Langdon, Hector Menendez Benito, and Justyna Petke. SearchGEM5: Towards reliable gem5 with search based software testing and large language models. In Paolo Arcaini, Tao Yue, and Erik Fredericks, editors, *SSBSE 2023: Challenge Track*, volume 14415 of *LNCS*, pages 60–166, San Francisco, USA, 8 December 2023. Springer. Winner best Challenge Track paper. doi:10.1007/978-3-031-48796-5_14.

[18] William B. Langdon, Shin Yoo, and Mark Harman. Inferring automatic test oracles. In Juan P. Galeotti and Justyna Petke, editors, *Search-Based Software Testing*, pages 5–6, Buenos Aires, Argentina, 22-23 May 2017. doi:10.1109/SBST.2017.1.

[19] Sungmin Kang, Juyeon Yoon, and Shin Yoo. Large language models are few-shot testers: Exploring LLM-based general bug reproduction. In *45th IEEE/ACM International Conference on Software Engineering, ICSE 2023, Melbourne, Australia, May 14-20, 2023*, pages 2312–2323. IEEE, 2023. doi:10.1109/ICSE48619.2023.00194.

[20] Sungmin Kang, Gabin An, and Shin Yoo. A quantitative and qualitative evaluation of LLM-based explainable fault localization. In *Proceedings of the 32nd International Conference on the Foundations of Software Engineering*, Porto de Galinhas, Brazil, 15-19 July 2024. To appear. URL: https://coinse.github.io/publications/pdfs/Kang2024ay.pdf.

[21] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL: https://openreview.net/pdf?id=1PL1NIMMrw.

[22] Robert Feldt. Genetic programming as an explorative tool in early software development phases. In Conor Ryan and Jim Buckley, editors, *Proceedings of the 1st International Workshop on Soft Computing Applied to Software Engineering*, pages 11–20, University of Limerick, Ireland, 12-14 April 1999. Limerick University Press. URL: http://www.cs.ucl.ac.uk/staff/W.Langdon/ftp/papers/scase_1999/feldt_1999_GPxtxsdp.pdf.

[23] Aymeric Blot and Justyna Petke. MAGPIE: Machine automated general performance improvement via evolution of software. arXiv, 4 August 2022. doi:10.48550/arxiv.2208.02811.

[24] Aymeric Blot. Automated software performance improvement with Magpie. In Gabin An, Aymeric Blot, Vesna Nowack, Oliver Krauss, and Justyna Petke, editors, *13th International Workshop on Genetic Improvement @ICSE 2024*, page v, Lisbon, 16 April 2024. ACM. Invited tutorial. URL: http://gpbib.cs.ucl.ac.uk/gi2024/an_2024_GI.pdf.

[25] 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. doi:10.1145/3338906.3341184.

[26] John R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA, USA, 1992. URL: http://mitpress.mit.edu/books/genetic-programming.

[27] Ricardo Sarmiento, Marina de la Cruz, Alfonso Ortega, Roberto Baena-Galle, Terrence Girard, Dana Casetti-Dinescu, and Alejandro Cervantes. Grammar evolution and symbolic regression for astrometric centering of Hubble space telescope images. In Gabin An, Aymeric Blot, Vesna Nowack, Oliver Krauss, and Justyna Petke, editors, *13th International Workshop on Genetic Improvement @ICSE 2024*, Lisbon, 16 April 2024. ACM. URL: http://gpbib.cs.ucl.ac.uk/gi2024/ICSE_24_GIGE_astronomy_cr_vldtd.pdf.

[28] Benjamin J. Craine, Penn Faulkner Rainford, and Barry Porter. Human guidance approaches for the genetic improvement of software. In Gabin An, Aymeric Blot, Vesna Nowack, Oliver Krauss, and Justyna Petke, editors, *13th International Workshop on Genetic Improvement @ICSE 2024*, Lisbon, 16 April 2024. ACM. URL: http://gpbib.cs.ucl.ac.uk/gi2024/Craine_2024_GI.pdf.

[29] James Callan, William B. Langdon, and Justyna Petke. On reducing network usage with genetic improvement. In Gabin An, Aymeric Blot, Vesna Nowack, Oliver Krauss, and Justyna Petke, editors, *13th International Workshop on Genetic Improvement @ICSE 2024*, Lisbon, 16 April 2024. ACM. URL: http://www.cs.ucl.ac.uk/staff/W.Langdon/ftp/papers/callan_2024_GI.pdf.

[30] William B. Langdon and David Clark. Deep mutations have little impact. In Gabin An, Aymeric Blot, Vesna Nowack, Oliver Krauss, and Justyna Petke, editors, *13th International Workshop on Genetic Improvement @ICSE 2024*, Lisbon, 16 April 2024. ACM. Best paper. URL: http://gpbib.cs.ucl.ac.uk/gi2024/langdon_2024_GI.pdf.

[31] Zsolt Nemeth, Penn Faulkner Rainford, and Barry Porter. Ecosystem curation in genetic improvement for emergent software systems. In Gabin An, Aymeric Blot, Vesna Nowack, Oliver Krauss, and Justyna Petke, editors, *13th International Workshop on Genetic Improvement @ICSE 2024*, Lisbon, 16 April 2024. ACM. Best position paper. URL: http://gpbib.cs.ucl.ac.uk/gi2024/GI_ICSE2024_Nemeth.pdf.

[32] Hunter Baxter, Yu Huang, and Kevin Leach. Genetic improvement for DNN security. In Gabin An, Aymeric Blot, Vesna Nowack, Oliver Krauss, and Justyna Petke, editors, *13th International Workshop on Genetic Improvement @ICSE 2024*, Lisbon, 16 April 2024. ACM. Best Presentation. URL: http://gpbib.cs.ucl.ac.uk/gi2024/Genetic_Improvement_for_DNN_Security.pdf.

[33] Justyna Petke, David Clark, and William B. Langdon. Software robustness: A survey, a theory, and some prospects. In Paris Avgeriou and Dongmei Zhang, editors, *ESEC/FSE 2021, Ideas, Visions and Reflections*, pages 1475–1478, Athens, Greece, 23-28 August 2021. ACM. doi:10.1145/3468264.3473133.

[34] Daniel Blackwell and David Clark. Prescientfuzz: A more effective exploration approach for grey-box fuzzing. arXiv 2404.18887, 29 April 2024. URL: https://arxiv.org/abs/2404.18887.

[35] William B. Langdon and Gabriela Ochoa. Genetic improvement: A key challenge for evolutionary computation. In Yun Li, editor, *Key Challenges and Future Directions of Evolutionary Computation*, pages 3068–3075, Vancouver, 25-29 July 2016. IEEE. doi:10.1109/CEC.2016.7744177.

[36] Sarah L. Thomson, Jason Adair, Alexander E. I. Brownlee, and Daan van den Berg. From fitness landscapes to explainable AI and back. In Giovanni Iacca, David Walker, Alexander Brownlee, Stefano Cagnoni, John McCall, and Jaume Bacardit, editors, *Evolutionary Computation and Explainable AI*, GECCO '23, pages 1663–1667, Lisbon, Portugal, 15-19 July 2023. Association for Computing Machinery. doi:10.1145/3583133.3596395.

[37] Dominik Sobania, Alina Geiger, James Callan, Alexander E. I. Brownlee, Carol Hanna, Rebecca Moussa, Mar Zamorano Lopez, Justyna Petke, and Federica Sarro. Evaluating explanations for software patches generated by large language models. In Paolo Arcaini, Tao Yue, and Erik Fredericks, editors, *SSBSE 2023: Challenge Track*, volume 14415 of *LNCS*, pages 147–152, San Francisco, USA, 8 Dec 2023. Springer. doi:10.1007/978-3-031-48796-5_12.

[38] Yuan Yuan and Wolfgang Banzhaf. Iterative genetic improvement: Scaling stochastic program synthesis. *Artificial Intelligence*, 322:103962, Sept 2023. doi:10.1016/J.ARTINT.2023.103962.

[39] Giovanni Pinna, Damiano Ravalico, Luigi Rovito, Luca Manzoni, and Andrea De Lorenzo. Enhancing large language models-based code generation by leveraging genetic improvement. In Mario Giacobini, Bing Xue, and Luca Manzoni, editors, *EuroGP 2024: Proceedings of the 27th European Conference on Genetic Programming*, volume 14631 of *LNCS*, pages 108–124, Aberystwyth, 3-5 April 2024. Springer. doi:10.1007/978-3-031-56957-9_7.

[40] Zachary P. Fry, Bryan Landau, and Westley Weimer. A human study of patch maintainability. In Zhendong Su, editor, *Proceedings of the 2012 International Symposium on Software Testing and Analysis, ISSTA 2012*, pages 177–187, Minneapolis, MN, USA, 15-20 July 2012. ACM. doi:10.1145/2338965.2336775.

[41] 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. doi:10.1109/GI.2019.00010.

[42] 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. doi:10.1109/ICSE-SEIP.2019.00039.

[43] 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. doi:10.1145/3387940.3392089.

[44] Mark Harman. Scaling genetic improvement and automated program repair. In Maria Kechagia, Shin Hwei Tan, Sergey Mechtaev, and Lin Tan, editors, *International Workshop on Automated Program Repair (APR'22)*, Internet, 19 May 2022. ACM. Invited keynote. doi:10.1145/3524459.3527353.

[45] Nadia Alshahwan, Mark Harman, and Alexandru Marginean. Software testing research challenges: An industrial perspective. In Sreedevi Sampath, editor, *16th IEEE International Conference on Software Testing, Verification and Validation (ICST 2023)*, pages 1–10, Dublin, Ireland, 16-20 April 2023. Keynote. doi:10.1109/ICST57152.2023.00008.

[46] Nadia Alshahwan, Mark Harman, Inna Harper, Alexandru Marginean, Shubho Sengupta, and Eddy Wang. Assured LLM-based software engineering, 15 April 2024. InteNSE 2024 Keynote. arXiv:2402.04380.

[47] Serkan Kirbas, Etienne Windels, Olayori McBello, Kevin Kells, Matthew Pagano, Rafal Szalanski, Vesna Nowack, Emily Winter, Steve Counsell, David Bowes, Tracy Hall, Saemundur Haraldsson, and John Woodward. On the introduction of automatic program repair in Bloomberg. *IEEE Software*, 38(4):43–51, July-August 2021. doi:10.1109/MS.2021.3071086.

[48] David Williams, James Callan, Serkan Kirbas, Sergey Mechtaev, Justyna Petke, Thomas Prideaux-Ghee, and Federica Sarro. User-centric deployment of automated program repair at Bloomberg. In *ICSE Software Engineering in Practice*, Lisbon, 14-20 April 2024. Best paper. URL: https://discovery.ucl.ac.uk/id/eprint/10187233.

[49] Emily Rowan Winter, Vesna Nowack, David Bowes, Steve Counsell, Tracy Hall, Sæmundur Haraldsson, John Woodward, Serkan Kirbas,

Etienne Windels, Olayori McBello, Abdurahman Atakishiyev, Kevin Kells, and Matthew Pagano. Towards developer-centered automatic program repair: findings from Bloomberg. In *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ESEC/FSE 2022, pages 1578–588, Singapore, 2022. doi:10.1145/3540250.3558953.

[50] Guolong Zheng, ThanhVu Nguyen, Simón Gutiérrez Brida, Germán Regis, Nazareno Aguirre, Marcelo F. Frias, and Hamid Bagheri. ATR: template-based repair for Alloy specifications. In *Proceedings of the 31st ACM SIGSOFT International Symposium on Software Testing and Analysis*, ISSTA 2022, pages 666–677, 2022. doi:10.1145/3533767.3534369.

[51] Simón Gutiérrez Brida, Germán Regis, Guolong Zheng, Hamid Bagheri, ThanhVu Nguyen, Nazareno Aguirre, and Marcelo Frias. Bounded exhaustive search of Alloy specification repairs. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*, pages 1135–1147, 2021. doi:10.1109/ICSE43902.2021.00105.

[52] Shahar Maoz and Jan Oliver Ringert. Spectra: a specification language for reactive systems. *Software and Systems Modeling*, 20(5):1553–1586, oct 2021. doi:10.1007/s10270-021-00868-z.

[53] Robert Feldt, Sungmin Kang, Juyeon Yoon, and Shin Yoo. Towards autonomous testing agents via conversational large language models. In *38th IEEE/ACM International Conference on Automated Software Engineering, ASE 2023, Luxembourg, September 11-15, 2023*, pages 1688–1693. IEEE, 2023. doi:10.1109/ASE56229.2023.00148.

[54] Michael O'Neill and Conor Ryan. *Grammatical Evolution: Evolutionary Automatic Programming in a Arbitrary Language*, volume 4 of *Genetic programming*. Kluwer Academic Publishers, 2003. doi:10.1007/978-1-4615-0447-4.

[55] Riccardo Poli, William B. Langdon, and Nicholas Freitag McPhee. *A field guide to genetic programming*. Published via http://lulu.com and freely available at http://www.gp-field-guide.org.uk, 2008. (With contributions by J. R. Koza). URL: http://www.gp-field-guide.org.uk.

[56] Hideyuki Takagi. Interactive evolutionary computation: Fusion of the capabilities of EC optimization and human evaluation. *Proceedings of the IEEE*, 89(9):1275–1296, September 2001. Invited Paper. doi:10.1109/5.949485.

[57] Peter Bentley and David Corne, editors. *Creative evolutionary systems*. Morgan Kaufmann, USA, 2002. URL: http://www.amazon.com/Creative-Evolutionary-Kaufmann-Artificial-Intelligence/dp/1558606734.

[58] Juan Romero and Penousal Machado, editors. *The Art of Artificial Evolution: A Handbook on Evolutionary Art and Music*. Natural Computing Series. Springer, 2008. doi:10.1007/978-3-540-72877-1.

[59] Pablo Funes, Elizabeth Sklar, Hugues Juille, and Jordan Pollack. Animal-animat coevolution: Using the animal population as fitness function. In Rolf Pfeifer, Bruce Blumberg, Jean-Arcady Meyer, and Stewart W. Wilson, editors, *From Animals to Animats 5: Proceedings of the Fifth International Conference on Simulation of Adaptive Behavior*, pages 525–533, Zurich, Switzerland, August 17-21 1998. MIT Press. doi:10.7551/mitpress/3119.003.0079.

[60] 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. doi:10.1145/3356773.3356801.

[61] William B. Langdon, Westley Weimer, Justyna Petke, Erik Fredericks, Seongmin Lee, Emily Winter, Michail Basios, Myra B. Cohen, Aymeric Blot, Markus Wagner, Bobby R. Bruce, Shin Yoo, Simos Gerasimou, Oliver Krauss, Yu Huang, and Michael Gerten. Genetic Improvement @ ICSE 2020. *SIGSOFT Software Engineering Notes*, 45(4):24–30, October 2020. doi:10.1145/3417564.3417575.

[62] Alexander E. I. Brownlee. Genetic Improvement @ ICSE 2021: Personal reflection of a workshop participant. *SIGSOFT Software Engineering Notes*, 46(4):28–30, October 2021. doi:10.1145/3485952.3485960.

[63] William B. Langdon, Vesna Nowack, Justyna Petke, Erik M. Fredericks, Gabin An, Aymeric Blot, Markus Wagner, and Hyeonseok Lee. Genetic Improvement @ ICSE 2023. *SIGSOFT Software Engineering Notes*, 48(4):51–59, October 2023. doi:10.1145/3617946.3617956.

[64] W. B. Langdon and Riccardo Poli. Removal of the man-machine interface bottleneck "Do what I ment not what I said". In *Grand Challenges for Computing*, Edinburgh, 24-26 November 2002. Discussion paper. URL: http://www.cs.ucl.ac.uk/staff/W.Langdon/ftp/papers/dwimrn0220.pdf.

[65] Westley Weimer, ThanhVu Nguyen, Claire Le Goues, and Stephanie Forrest. It does what you say, not what you mean: Lessons from a decade of program repair. ICSE 2019 Plenary Most Inflential Paper, 30 May 2019. URL: https://conf.researchr.org/profile/icpc-2019/westleyweimer.