A Novelty Search and Metamorphic Testing Approach to Automatic Test Generation

Byron DeVries
School of Computing
Grand Valley State University
Allendale, Michigan
Email: devrieby@gvsu.edu

Christian Trefftz
School of Computing
Grand Valley State University
Allendale, Michigan
Email: trefftzc@gvsu.edu

***Abstract*—A common task in search-based testing is automatically identifying valuable test cases for software systems. However, existing approaches tend to either search for unique tests with regard to inputs or outputs (i.e., novelty search) *or* search for inputs that invalidate some expected proposition regarding the software (i.e., metamorphic testing). Problematically, verifying unique tests induces the oracle problem while an invalidated proposition results in a single test case. In this paper we utilize novelty search and metamorphic testing to discover a broad range of unique test cases that are directly verifiable via a metamorphic relation *and* invalidate such an expected proposition in fewer generations of an evolutionary algorithm than direct search. We apply this novelty search and metamorphic testing combination to discover errors in identifying the midpoint of a geodesic as a proof-of-concept.**

***Index Terms*—novelty search, metamorphic testing, test generation**

I. Introduction

Search-based generation of test data is an appealing solution to the time-consuming and difficult process of manually identifying test cases. It is so appealing that we decided to use it to verify our use of geodesic geometry. However, for each test case identified it is also necessary to identify a method of verification - most typically expected results from the software. Problematically, it is difficult to automatically identify expected results (i.e., the oracle problem) and, in some cases, it is infeasible [1].

Two primary methods of generating test cases have been explored in the literature: optimizing for a specific property (e.g., code coverage [2]) that may utilize an attempt to generate a test oracle and novelty search-based methods [3]. Problematically, code coverage does not necessarily directly correlate with eliciting faults [4], [5] and novelty search-based methods require an oracle often derived from the code itself (e.g., mutation-based testing methods [6]).

While direct optimization of a fitness function may identify a verifiable test case of interest and novelty search may identify multiple diverse behaviors, neither identifies a set of multiple test cases that can be automatically verified. In this paper we utilize metamorphic relations to verify multiple test cases identified by novelty search. Further, using novelty search has been shown to optimize objectives in less generations of a genetic algorithm than direct objective optimization despite only explicitly searching for novel (not optimal) solutions [7].

The contributions of this paper are as follows:

- We present a method using both novelty search and metamorphic testing to identify unique and verifiable test cases, and
- We present a proof-of-concept and compare the results of the combined approach to direct search alone.

The remainder of this paper is organized as follows. Section II provides an overview of the background information. Section III introduces the approach. Section V covers related work, and Section VI discusses conclusions and future work.

II. Background

This section covers background in novelty search, metamorphic testing, and the geometry of geodesics.

A. Novelty Search

While most optimization methods attempt to optimize some objective, novelty search ignores the objective entirely. Counterintuitively, novelty search looks for unique phenotypic behaviors from the outputs rather than optimal behaviors [7]. Novelty search does this by setting the objective of a traditional genetic algorithm to reward outputs that are most different than what has been identified previously - often within some set number of nearest neighbors. When a sufficiently novel output is identified, it is added to a novelty archive that other outputs are measured against. Rather than returning a single optimum, novelty search returns a whole range of outputs within its population and novelty archive. In some applications novelty search identifies optimal results in fewer generations than direct optimization [7]. In this paper we refer to traditional genetic algorithms that optimize for a fitness function directly as *direct search* to differentiate from *novelty search*.

B. Metamorphic Testing

Originally presented as a method to generate tests [8], the use of metamorphic testing has expanded to overcome the oracle problem and is used across a wide variety of domains [9]. Rather than test via inputs and expected outputs, metamorphic testing establishes *metamorphic relations* that

indicate how system behavior can be compared to other behavior. Metamorphic relations can either be input driven (e.g., two different inputs should provide the same result) or output driven (e.g., two different outputs are comparable based on a known difference in inputs). These metamorphic relations enable metamorphic testing to provide a method of generating multiple test cases *and* actually verify without an oracle. For example, given a program to detect exactly how many pine needles are on a tree, how could expected results (e.g., the number of pine needles) be found? It is not practical to actually count the pine needles, but it is practical to obtain the program results for a tree, then (secretly) take some number of pine needles off the tree and obtain the program results again. The difference in program response should match the alteration to the tree. By changing an input to the program, a metamorphic relation can be used to compare the outputs without ever knowing the correct result. Similarly, multiple test cases can be generated by removing differing numbers of pine needles.

C. Geometry of Geodesics

While the shortest distance between two points in Euclidean space is straight, the shortest distance between two points across the surface of the earth is not. First, since the earth is not flat, the shortest path along the surface must be curved to the surface between the two points. Second, since the path must conform to the surface of the earth, it is shorter to "bend" the path towards the nearest pole to minimize the additional curve to match the surface. In Figure 1 a straight line from point **A** to **C** would actually be longer than the curved line (through **B**). This is one of the primary reasons flights between North America and Europe *appear* to "bend" further north on a two-dimensional projection of the earth. These shortest paths are *geodesic* segments on a *great circle* that represents the largest circle that can be created dividing a sphere in half [10]. The equator, for example, is a great circle, as is every line of longitude.

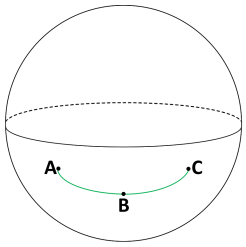

Fig. 1. Shortest Path on a Sphere

In this paper we use a Java port of GeographicLib[1] that uses existing geodesic algorithms in [11] where errors are less than 15 nanometers for our desired operations [12] on an earth-representing WGS84 (World Geodetic System 1984) ellipsoid model.

III. Approach

In this section we describe the geodesic midpoint problem we aim to identify test cases for, a metamorphic relation for the geodesic midpoint problem, our direct search method, and our novelty search method.

A. Geodesic Midpoint Problem

The software under test calculates the midpoint of a geodesic defined by two points on earth using latitude and longitude as part of a geodesic implementation of an existing divide-and-conquer Voronoi diagram generator for two-dimensional Euclidean space [13]. While an exceedingly straight-forward operation in two-dimensional Euclidean geometry, dividing the shortest path between two points on a spheroid has some pitfalls, including:

- Difficulties around, at, and over the poles, and
- Rollover from negative to positive longitude (or vice-versa) at the international date line.

Consider the curved line in Figure 1 from **A** through **B** to **C**. The midpoint (i.e., point **B**) divides the distance on the shortest path into two geodesic segments that are each half the length of the original. Verifying this point is accurately identified is difficult without relying on the same operations used to define the point originally.

B. Metamorphic Relation

Given two points made up of latitude, ϕ, and longitude, λ, we expect the same midpoint to be calculated regardless of which point the midpoint is calculated from. That is, given the points (ϕ_1, λ_1) and (ϕ_2, λ_2), a function that returns the midpoint should return the same value regardless of the order of the points. The following metamorphic relationship is true outside of the expected error (i.e., up to 15 nanometers in the algorithms used by GeographicLib):

$$\text{mid}((\phi_1, \lambda_1), (\phi_2, \lambda_2)) \approx \text{mid}((\phi_2, \lambda_2), (\phi_1, \lambda_1)) \quad (1)$$

Test cases can be created for any set of valid latitude and longitude for two points and verified using Equation 1, without knowledge of the expected results.

C. Direct Search

In order to identify the largest possible error detectable by the metamorphic relation in Equation 1, we identify a fitness function that maximizes the difference between the two calculated midpoints. We use the latitude and longitude as points on a two-dimensional Euclidean plane instead of measuring actual physical distance to avoid relying on the geodesic operations under test:

$$\text{fitness} = \sqrt{(\phi_{\text{mid}_1} - \phi_{\text{mid}_2})^2 + (\lambda_{\text{mid}_1} - \lambda_{\text{mid}_2})^2} \quad (2)$$

We maximize the fitness function defined in Equation 2 via a genetic algorithm with a 3% chance of mutation, simulated binary crossover [14], and tournament selection across three individuals using the Jenetics[2] library. A population of 500 is used and the number of generations is limited to 1000.

[1] https://geographiclib.sourceforge.io/

[2] https://jenetics.io/

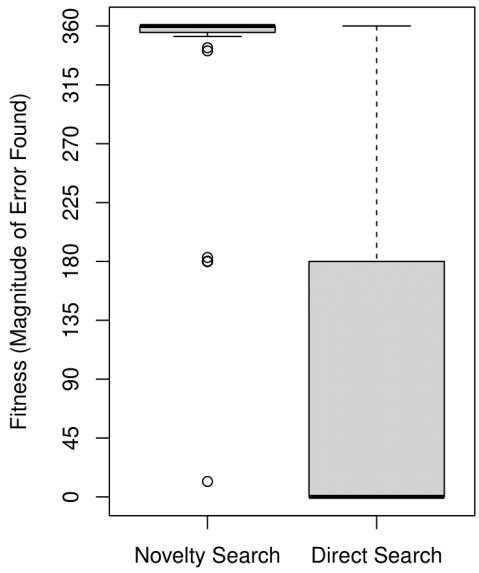

Fig. 2. Comparison of Final Fitness

D. Novelty Search

Rather than optimizing for the fitness function in Equation 2 directly, the genetic algorithm described previously is extended to include a novelty archive (limited to 100 individuals). The fitness function is replaced by the average two-dimensional Euclidean distance between the 10 nearest individuals computed using the two midpoints representing the phenotypic output as a 4-tuple (i.e., ϕ_{mid_1}, λ_{mid_1}, ϕ_{mid_2}, λ_{mid_2}). That is, the novelty measurement is defined as:

$$\text{novelty}(x) = \frac{1}{k}\sum_{i=0}^{k} \text{dist}(x, \mu_i) \tag{3}$$

Where k is 10 and μ is the novelty archive in order of distance to x, and x is the 4-tuple for which the novelty is being calculated.

IV. Results

Both novelty search and a direct search via a genetic algorithm were used to identify the most problematic (i.e., highest fitness from Equation 2) test cases over a series of 50 runs each, limited to 1000 generations with a population of 500 individuals. Novelty search employed a novelty archive of 100, which provides a diverse set of test cases, while the novelty metric was based on the average distance from 10 nearest neighbors.

Figure 2 shows a comparison of the fitness of the most problematic individual at the end of 1000 generations for both the novelty and direct search. Direct search is infrequently able to achieve as high a fitness as novelty search. The median fitness is only 5.5e-11 while the average fitness is 85.0. Novelty search obtained a median fitness of 360.0 and an average fitness of 333.4. We used Wilcoxon-Mann-Whitney U-test to measure statistical significance, due to the non-normal distribution of each dataset [15], [16], and the difference is significant ($p <$ 5e-11).

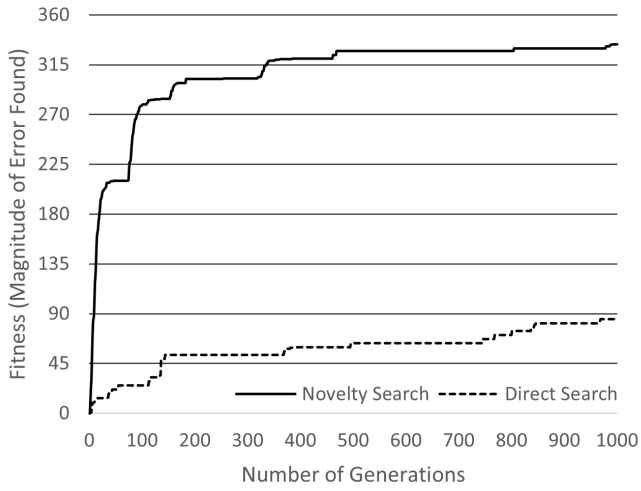

Fig. 3. Comparison of Average Fitness Convergence

Figure 3 shows the average maximum fitness of the 50 runs over the 1000 generations the novelty search and direct search genetic algorithms were executed. It is clear that, despite not expressly searching to maximize the fitness defined by Equation 2, the novelty search converges significantly more rapidly than the direct search method.

Multiple smaller errors are identified due to computations around and on the poles. The largest error, however, is due to both $180°$ and $-180°$ longitude referring to the same location, a problem that could cause significant downstream effects on an application that depends on tight ranges.

A. Limitations & Threats to Validity

While the results are promising, there are several limitations that must be acknowledged. First, the example is limited to a single, limited, case study with a single metamorphic relationship. Second, the rate of convergence is likely to be impacted by the fitness functions or novelty metrics chosen. The novelty metric used in novelty search must be chosen carefully from amongst the possible measures of behavior. Third, metamorphic relations must be manually defined in order to generate test cases, often a difficult task [17], [18]. Finally, the computational cost of each novelty search generation is higher due to the additional overhead of calculating novelty for each individual in the population using the k-nearest neighbors.

V. Related Work

Significant work has been done in the area of generating test data, including work that uses novelty search [3]. For example, novelty search has been used to address environmental uncertainty in systems by searching for unexpected behavior [19]–[21] and verifying deep learning systems [22]. Metamorphic testing has been used in a wide variety of applications, including everything from compilers [23] to self-driving cars [24]. Two recent metamorphic testing surveys in

2016 [25] and 2018 [9] cover the broad range of application areas. Our method differs from each of these methods by combining *both* novelty search with metamorphic testing to achieve a wide range of verifiable tests while greatly increasing the convergence rate.

Other methods to generate test data often require some form of implicit, derived, or specified test oracle [26]. In the case of our work, we manually specify the test oracle via the metamorphic relation. However, mutation-driven test oracles [6], crowd-sourced test oracles [27] including those for crowd-sourcing metamorphic relations [28], and even methods to improve code-based test generation via metamorphic testing [29] provide at least some level of automation. While our current case study requires manually defined metamorphic relations, existing work on automatically generating metamorphic relations [30]–[32] could be leveraged while still benefiting from the combined benefit of novelty search.

VI. Conclusions

In this paper, we have presented an approach to automatically identifying valuable test cases for software systems using a combination of novelty search and metamorphic testing.

We have demonstrated our approach on a small case study as a proof-of-concept that shows the benefit to the combination of novelty search and metamorphic testing. Future research directions are focused on extending the domains of the case study to provide greater empirical evidence and address the limitations previously identified. Additionally, we hope to integrate automatically identified metamorphic relations to reduce the manual effort required.

References

[1] S. Segura, D. Towey, Z. Q. Zhou, and T. Y. Chen, "Metamorphic testing: Testing the untestable," *IEEE Software*, 2018.

[2] G. Fraser and A. Arcuri, "Evosuite: automatic test suite generation for object-oriented software," in *Proceedings of the 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering*, 2011, pp. 416–419.

[3] M. Boussaa, O. Barais, G. Sunyé, and B. Baudry, "A novelty search approach for automatic test data generation," in *2015 IEEE/ACM 8th International Workshop on Search-Based Software Testing*. IEEE, 2015, pp. 40–43.

[4] C. Gaffney, C. Trefftz, and P. Jorgensen, "Tools for coverage testing: necessary but not sufficient," *Journal of Computing Sciences in Colleges*, vol. 20, no. 1, pp. 27–33, 2004.

[5] L. Inozemtseva and R. Holmes, "Coverage is not strongly correlated with test suite effectiveness," in *Proceedings of the 36th international conference on software engineering*, 2014, pp. 435–445.

[6] G. Fraser and A. Zeller, "Mutation-driven generation of unit tests and oracles," *IEEE Transactions on Software Engineering*, vol. 38, no. 2, pp. 278–292, 2011.

[7] J. Lehman and K. O. Stanley, "Abandoning objectives: Evolution through the search for novelty alone," *Evolutionary computation*, vol. 19, no. 2, pp. 189–223, 2011.

[8] T. Chen, S. Cheung, and S. Yiu, "Metamorphic testing: a new approach for generating next test cases. technical report hkust-cs98-01," *Hong Kong Univ. of Science and Technology*, 1998.

[9] T. Y. Chen, F.-C. Kuo, H. Liu, P.-L. Poon, D. Towey, T. Tse, and Z. Q. Zhou, "Metamorphic testing: A review of challenges and opportunities," *ACM Computing Surveys (CSUR)*, vol. 51, no. 1, pp. 1–27, 2018.

[10] P. Pokorny, "Geodesics revisited," *Chaotic Modeling and Simulation*, pp. 281–298, 2012.

[11] C. F. Karney, "Algorithms for geodesics," *Journal of Geodesy*, vol. 87, no. 1, pp. 43–55, 2013.

[12] ——, "Geodesics on an ellipsoid of revolution," *arXiv preprint arXiv:1102.1215*, 2011.

[13] E. Smith, C. Trefftz, and B. DeVries, "A divide-and-conquer algorithm for computing voronoi diagrams," in *2020 IEEE International Conference on Electro Information Technology (EIT)*. IEEE, 2020, pp. 495–499.

[14] K. Deb and R. B. Agrawal, "Simulated binary crossover for continuous search space," *Complex Systems*, vol. 9, no. 3, pp. 1–15, 1994.

[15] A. Arcuri and L. Briand, "A practical guide for using statistical tests to assess randomized algorithms in software engineering," in *2011 33rd International Conference on Software Engineering (ICSE)*. IEEE, 2011, pp. 1–10.

[16] ——, "A hitchhiker's guide to statistical tests for assessing randomized algorithms in software engineering," *Software Testing, Verification and Reliability*, vol. 24, no. 3, pp. 219–250, 2014.

[17] W. B. Langdon, S. Yoo, and M. Harman, "Inferring automatic test oracles," in *2017 IEEE/ACM 10th International Workshop on Search-Based Software Testing (SBST)*. IEEE, 2017, pp. 5–6.

[18] P. A. Nardi and E. F. Damasceno, "A survey on test oracles," 2015.

[19] A. J. Ramirez, A. C. Jensen, B. H. C. Cheng, and D. B. Knoester, "Automatically exploring how uncertainty impacts behavior of dynamically adaptive systems," in *Proceedings of the 2011 26th IEEE/ACM International Conference on Automated Software Engineering*. IEEE Computer Society, 2011, pp. 568–571.

[20] M. A. Langford, G. A. Simon, P. K. McKinley, and B. H. Cheng, "Applying evolution and novelty search to enhance the resilience of autonomous systems," in *2019 IEEE/ACM 14th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*. IEEE, 2019, pp. 63–69.

[21] M. A. Langford and B. H. Cheng, "Enhancing learning-enabled software systems to address environmental uncertainty," in *2019 IEEE International Conference on Autonomic Computing (ICAC)*. IEEE, 2019, pp. 115–124.

[22] V. Riccio and P. Tonella, "Model-based exploration of the frontier of behaviours for deep learning system testing," in *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2020, pp. 876–888.

[23] Q. Tao, W. Wu, C. Zhao, and W. Shen, "An automatic testing approach for compiler based on metamorphic testing technique," in *2010 Asia Pacific Software Engineering Conference*. IEEE, 2010, pp. 270–279.

[24] M. Zhang, Y. Zhang, L. Zhang, C. Liu, and S. Khurshid, "Deeproad: Gan-based metamorphic testing and input validation framework for autonomous driving systems," in *2018 33rd IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2018, pp. 132–142.

[25] S. Segura, G. Fraser, A. B. Sanchez, and A. Ruiz-Cortés, "A survey on metamorphic testing," *IEEE Transactions on software engineering*, vol. 42, no. 9, pp. 805–824, 2016.

[26] E. T. Barr, M. Harman, P. McMinn, M. Shahbaz, and S. Yoo, "The oracle problem in software testing: A survey," *IEEE transactions on software engineering*, vol. 41, no. 5, pp. 507–525, 2014.

[27] F. Pastore, L. Mariani, and G. Fraser, "Crowdoracles: Can the crowd solve the oracle problem?" in *2013 IEEE Sixth International Conference on Software Testing, Verification and Validation*. IEEE, 2013, pp. 342–351.

[28] Y. Yang and C. Xu, "Mr hunter: Hunting for metamorphic relations by puzzle solving," in *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops*, 2020, pp. 404–409.

[29] P. Saha and U. Kanewala, "Improving the effectiveness of automatically generated test suites using metamorphic testing," *arXiv preprint arXiv:2004.08518*, 2020.

[30] A. Nair, K. Meinke, and S. Eldh, "Leveraging mutants for automatic prediction of metamorphic relations using machine learning," in *Proceedings of the 3rd ACM SIGSOFT International Workshop on Machine Learning Techniques for Software Quality Evaluation*, 2019, pp. 1–6.

[31] B. Zhang, H. Zhang, J. Chen, D. Hao, and P. Moscato, "Automatic discovery and cleansing of numerical metamorphic relations," in *2019 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 2019, pp. 235–245.

[32] U. Kanewala, "Techniques for automatic detection of metamorphic relations," in *2014 IEEE Seventh International Conference on Software Testing, Verification and Validation Workshops*. IEEE, 2014, pp. 237–238.