Разработка и исследование модификаций подхода генерации тестовых данных

К.Е. Сердюков[1], Т.В. Авдеенко[1]

[1]*Новосибирский государственный технический университет, К. Маркса, 20, Новосибирск, Россия, 630073*

Аннотация

В данной работе исследуется подходы для решения проблемы генерации данных для множества путей исходного кода при помощи генетического алгоритма. В первом подходе генетический алгоритм используется для поиска наиболее адаптированной хромосомы, представляющей собой набор тестовых данных, обеспечивающих прохождение по самому сложному пути. Множество наборов данных, обеспечивающих максимальное покрытие кода, находится итерационным повторением запуска генетического алгоритма с предварительным обнулением весов операций, соответствующих ранее найденным хромосомам. Другой подход предполагает, что в функцию приспособленности изначально включаются два компонента. Первый компонент, как и в первом подходе, отвечает за сложность каждого пути. Второй компонент отвечает за максимально возможные различия в путях популяции. Для второго подхода были разработаны модификации алгоритма, которые направлены на разрешение проблемы нестабильной сходимости подхода.

Ключевые слова

генерация тестовых данных, генетический алгоритм, множественная генерация данных

1. Введение

Задачу по определению наиболее подходящих тестовых данных очень сложно решить при помощи стандартных алгоритмов, так как фактическое количество всего массива возможных значений невероятно огромно, и подобрать действительно подходящие перебором значений не представляется возможным. При этом, автоматизация данного процесса может существенно уменьшить аналитическую нагрузку на пользователей, занимающихся тестированием.

Использование генетических алгоритмов позволяет сравнивать множество различных вариантов данных для тестирования программы. Широкие возможности к усовершенствованию позволяет увеличить количество начальных тестовых вариантов, количество поколений и добавить новые свойства, благодаря которым можно существенно увеличить возможности нахождения более подходящих вариантов.

2. Модификации подхода генерации данных

Одним из способов решения этой проблемы является замена операции возведения в квадрат операцией модуля и умножение модуля разности на коэффициент k, который является константой, чтобы обеспечить более равномерное изменение уникальности варианта и увеличить влияние первого слагаемого (длина пути).

$$f_{fit}(x_i, C_s(\mathrm{w})) = f_b(x_i) + k * \left| f_{avg\,\mathrm{sim}} - f_{\mathrm{sim}}(P_e(x_i), C_s(\mathrm{w})) \right| \quad (1)$$

Коэффициент k может принимать разные значения в зависимости от тестируемого кода.

Кроме того, предлагается изменить алгоритм двумя способами. Первый метод мы назвали «маркировка элитных вариантов». Если вариант и путь, по которому проходят тестовые

данные, уникальны, то есть отличаются от сгенерированных ранее, то этот вариант помечается как «элитный» и никогда не исключается из генерации в будущем. Это свойство позволяет сохранять уникальные варианты, чтобы избежать их исчезновения при уменьшении значения функции приспособленности в последующих поколениях.

Вторая модификация заключается в динамическом изменении второго слагаемого функции приспособленности, учитывая уникальность рассматриваемого варианта. Чтобы избежать заполнения поколения одними и теми же вариантами и их потомками, было предложена динамическая версия функции приспособленности, в которой второе слагаемое делится на количество путей, соответствующих проверяемому варианту.

$$f_{fit}\big(x_i, C_s(w)\big) = f_b(x_i) + \frac{k * \left|f_{avg\ sim} - f_{sim}\big(P_e(x_i), C_s(w)\big)\right|}{m(x_1, x_2, \dots, x_i)}, \quad (2)$$

где $m(x_1, x_2, \dots, x_i)$ – количество путей из набора $\{P_e(x_1), P_e(x_2), \dots, P_e(x_i)\}$, совпадающих с $P_e(x_i)$.

3. Заключение

Сгенерированные тесты могут служить не только для тестирования алгоритма, но анализа его работоспособности. Данные в своём чистом ви-де уже позволяют определить закономерности программного кода, а при более глубоком изучении могут позволить сделать выводы о возможном последующем улучшении. Такой анализ может быть использован в качестве темы дальнейших исследований в области анализа данных.

Использование генетических алгоритмов в процессе тестирования позволяют обеспечить нахождение наиболее сложных частей программы, в которых риски из-за допущения ошибок наиболее велики. Оценивание происходит за счёт использования функции приспособленности, в качестве параметров которой выступают веса каждой проходимой операции.

4. Благодарности

Исследование выполнено при финансовой поддержке РФФИ в рамках научного проекта № 19-37-90156 «Разработка и исследование методов интеллектуального анализа и тестирования программного кода».

Работа выполнена при финансовой поддержке Министерства Науки и Высшего Образования в рамках Госзадания (проект № FSUN-2020-0009).

5. Литература

[1] Doungsaard, C. An automatic test data generation from UML state diagram using genetic algorithm / C. Doungsaard, K. Dahal, A.G. Hossain, T. Suwannasart // IEEE Computer Society Press. – 2007 – P. 47-52.

[2] Grochtmann, M. Classification trees for partition testing. Software Testing / M. Grochtmann, K. Grimm // Verification and Reliability. – 1993. – Vol. 3(2). – P. 63-82.

[3] Chen, T.Y. An integrated classification-tree methodology for test case generation / T.Y. Chen, P.L. Poon, T.H. Tse // International Journal of Software Engineering and Knowledge Engineering. – 2000. – Vol. 10(6). – P. 647-679.

[4] Cain, A. An Automatic Test Data Generation System Based on the Integrated Classification-Tree Methodology / A. Cain, T.Y. Chen, D. Grant, P.L. Poon, S.F. Tang, T.H. Tse // Software Engineering Research and Applications, Lecture Notes in Computer Science. – 2004. – Vol. 3026.

[5] Serdyukov, K. Investigation of the genetic algorithm possibilities for retrieving relevant cases from big data in the decision support systems / K. Serdyukov, T. Avdeenko // CEUR Workshop Proceedings. – 2017. –Vol. 1903. – P. 36-41.

[6] Praveen, R.S. Application of Genetic Algorithm in Soft-ware Testing / R.S. Praveen, K. Taihoon // International Journal of Software Engineering and Its Applications. – 2009. – Vol. 3(4). – P. 87-96.
[7] Serdyukov, K. Researching of methods for assessing the complexity of program code when generating input test data / K. Serdyukov, T. Avdeenko // CEUR Workshop Proceedings. – 2020. – Vol. 2667. – P. 299-304.