[1] G. J. Myers, C. Sandler, T. Badgett. The art of software testing. John Wiley and Sons, 2011.

[2] S. Anand, E. K. Burke, T. Y. Chen, J. Clark, M. B. Cohen, W. Grieskamp, M. Harman, M. J. Harrold, P. McMinn. An orchestrated survey of methodologies for automated software test case generation. *Journal of Systems and Software*, Elsevier, 2013, 86(8): 1978-2001.

[3] R. Baldoni, E. Coppa, D. C. Delia, C. Demetrescu, I. Finocchi. A survey of symbolic execution techniques. *ACM Computing Surveys (CSUR)*, ACM Press, 2018, 51(3): 50:1-50:39.

[4] J. C. King. A new approach to program testing. *Proceedings of the International Conference on Reliable Software*. ACM Press, 1975, pp. 228-233.

[5] D. M. Leonardo, N. Bjørner. Z3: An efficient SMT solver. *Proceedings of the 14th International conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2008)*. Springer, 2008, pp. 337-340.

[6] V. Ganesh, D. L. Dill. A decision procedure for bit-vectors and arrays. *Proceedings of the 19th International Conference on Computer Aided Verification (CAV 2007)*. Springer, 2007, pp. 519-531.

[7] V. Chipounov, V. Georgescu, C. Zamfir. Selective symbolic execution. *Proceedings of the 5th Workshop on Hot Topics in System Dependability (HotDep 2009)*, *Co-located with the 39th IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2009)*, IEEE Computer Society, 2009, pp. 1-6.

[8] P. Odefroid, N. Klarlund, K. Sen. DART: Directed automated random testing. *Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI 2015)*. ACM Press, 2005, pp. 213-223.

[9] C. Cadar, D. Dunbar, D. R. Engler. KLEE: Unassisted and automatic generation of high-coverage tests for complex systems programs. *Proceedings of the 8th USENIX Conference on Operating Systems Design and Implementation (OSDI 2008)*. USENIX, 2008, pp. 209-224.

[10] P. Godefroid, M. Y. Levin, D. A. Molnar. Automated whitebox fuzz testing. *Proceedings of the 15th Annual Network and Distributed System Security Symposium (NDSS 2008)*. ISOC, 2008, pp. 151-166.

[11] A. C. Dias Neto, R. Subramanyan, M. Vieira, et al. A survey on model-based testing approaches: A systematic review. *Proceedings of the 1st ACM International Workshop on Empirical Assessment of Software Engineering Languages and Technologies (WEASELTech 2007), Co-located with the 22nd IEEE/ACM International Conference on Automated Software Engineering (ASE 2007)*. ACM Press, 2007, pp. 31-36.

[12] D. Lee, M. Yannakakis. Principles and methods of testing finite state machines-a survey. *Proceedings of the IEEE*. IEEE Computer Society, 1996, 84(8): 1090-1123.

[13] S. Kansomkeat, W. Rivepiboon. Automated-generating test case using UML statechart diagrams. *Proceedings of the 2003 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists on Enablement Through Technology (SAICSIT 2003)*. South African Institute for Computer Scientists and Information Technologists, 2003, pp. 296-300.

[14] X. Han, T. T. Yu, and D. Lo. PerfLearner: learning from bug reports to understand and generate performance test frames. *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering (ASE 2018)*. ACM Press, 2018, pp. 17-28.

[15] D. M. Cohen, S. R. Dalal, M. L. Fredman, et al. The AETG system: an approach to testing based on combinatorial design. *IEEE Transactions on Software Engineering*, IEEE Computer Society, 1997, 23(7): 437-444.

[16] Y. Lei, R. Kacker, D. R. Kuhn, et al. IPOG/IPOG-D: efficient test generation for multi-way combinatorial testing. *Software: Testing, Verification and Reliability*, Wiley, 2008, 18(3): 125-148.

[17] S. A. Ghazi, M. A. Ahmed. Pair-wise test coverage using genetic algorithms. *Proceedings of the Congress on Evolutionary Computation (CEC 2003)*. IEEE Computer Society, 2003, pp. 1420-1424.

[18] R. C. Bryce, C. J. Colbourn. One-test-at-a-time heuristic search for interaction test suites. *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation (GECCO 2007)*. ACM Press, 2007, pp. 1082-1089.

[19] N. Kobayashi, T. Tsuchiya, T. Kikuno. A new method for constructing pair-wise covering designs for software testing. *Information Processing Letters*, Elsevier, 2002, 81(2): 85-91.

[20] F. Medeiros, C. Kästner, M. Ribeiro, et al. A comparison of 10 sampling algorithms for configurable systems. *Proceedings of the 38th International Conference on Software Engineering (ICSE 2016)*. ACM Press, 2016, pp. 643-654.

[21] T. Y. Chen, H. Leung, I. K. Mak. Adaptive random testing. *Proceedings of the 9th Annual Asian Computing Science Conference (ASIAN 2004)*. Springer, 2004, pp. 320-329.

[22] T. Y. Chen, F. C. Kuo, R. G. Merkel. Mirror adaptive random testing. *Information and Software Technology*, Elsevier, 2004, 46(15): 1001-1010.

[23] T. Y. Chen, R. Merkel, P. K. Wong. Adaptive random testing through dynamic partitioning. *Proceedings of the 4th International Conference on Quality Software (QSIC 2004)*. IEEE Computer Society, 2004: pp. 79-86.

[24] M. Harman, S. A. Mansouri, Y. Zhang. Search based software engineering: a comprehensive analysis and review of trends techniques and applications. Technical Report TR-09-03, Department of Computer Science, King’s College London, 2009.

[25] M. A. Ahmed, I. Hermadi. GA-based multiple paths test data generator. *Computers & Operations Research*, Elsevier, 2008, 35(10): 3107-3124.

[26] M. Xiao, M. El-Attar, M. Reformat, et al. Empirical evaluation of optimization algorithms when used in goal-oriented automated test data generation techniques. *Empirical Software Engineering*, Springer, 2007, 12(2): 183-239.

[27] A. S. Ghiduk, M. J. Harrold, M. R. Girgis. Using genetic algorithms to aid test-data generation for data-flow coverage. *Proceedings of the 14th Asia-Pacific Software Engineering Conference (APSEC 2007)*. IEEE Computer Society, 2007, pp. 41-48.

[28] H. Li, C. P. Lam. An ant colony optimization approach to test sequence generation for state-based software testing. *Proceedings of the 5th International Conference on Quality Software (QSIC 2005)*. IEEE Computer Society, 2005, pp. 255-264.

[29] R. Lefticaru, F. Ipate. Automatic state-based test generation using genetic algorithms. *Proceedings of the 9th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC 2007)*. IEEE Computer Society, 2007, pp. 188-195.

[30] B. Baudry, F. Fleurey, J. M. Jézéquel, et al. From genetic to bacteriological algorithms for mutation-based testing. *Software: Testing, Verification and Reliability*, Wiley, 2005, 15(2): 73-96.

[31] K.-Y. Cai. Optimal software testing and adaptive software testing in the context of software cybernetics. *Information and Software Technology*, Elsevier, 2002, 44(14): 841–855.

[32] G. B. Finelli. NASA software failure characterization experiments. *Reliability Engineering and System Safety*, Elsevier, 1991, 32(1): 155–169.

[33] K.-Y. Cai, H. Hu, and F. Ye. Random Testing with Dynamically Updated Test Profile. *Proceedings of the 20th International Symposium on Software Reliability Engineering (ISSRE 2009)*, IEEE Computer Society, 2009, pp. 1-2.

[34] Y. Li, B. B. Yin, J. Lv, and K.-Y. Cai. Approach for test profile optimization in dynamic random testing. *Proceedings of the 39th IEEE Annual International Computer Software and Applications Conference (COMPSAC 2015)*, IEEE Computer Society, 2015, pp. 466–471.

[35] Z. Yang, B. Yin, J. Lv, K.-Y. Cai, S. S. Yau, and J. Yu. Dynamic random testing with parameter adjustment. *Proceedings of the 6th IEEE International Workshop on Software Test Automationthe, Co-located with the 38th IEEE Annual International Computer Software and Applications Conference (COMPSAC 2014)*, IEEE Computer Society, 2014, pp. 37–42.

[36] J. Lv, H. Hu, and K.-Y. Cai. A sufficient condition for parameters estimation in dynamic random testing. *Proceedings of the 3rd IEEE International Workshop on Software Test Automation, Co-located with the 35th IEEE Annual International Computer Software and Applications Conference (COMPSAC 2011)*, IEEE Computer Society, 2011, pp. 19–24.

[37] K. Patel and R. M. Hierons. A mapping study on testing nontestable systems. *Software Quality Journal*, Springer, 2018, 26(4): 1373–1413, 2018.

[38] S. S. Brilliant, J. C. Knight, P. E. Ammann. On the performance of software testing using multiple versions. *Proceedings of the 20th international symposium on fault-tolerant computing (FTCS 2009)*, IEEE Computer Society, 1990, pp. 408–415.

[39] K. Y. Sim, C. S. Low, F. C. Kuo. Eliminating human visual judgment from testing of financial charting software. *Journal of Software*, Elsevier, 2014, 9(2): 298-312.

[40] W. K. Chan, S. C. Cheung. PAT: A pattern classification approach to automatic reference oracles for the testing of mesh simplification programs. *Journal of Systems and Software*, Elsevier, 2009, 82(3): 422–434.

[41] R. Guderlei, J. Mayer. Statistical metamorphic testing test programs with random output by means of statistical hypothesis tests and metamorphic testing. *Proceedings of the 7th International Conference on Quality Software (QSIC 2007)*, IEEE Computer Society, 2007, pp. 404–409.

[42] T. Y. Chen, S. C. Cheung, S. M. Yiu. Metamorphic testing: A new approach for generating next test cases. Technical Report HKUST-CS98-01, Department of Computer Science, Hong Kong University of Science and Technology, Hong Kong, 1998.

[43] S. Segura, G. Fraser, A. B. Sanchez, A. R. Cortes. A survey on metamorphic testing. *IEEE Transactions on Software Engineering*, IEEE Computer Society, 2016, 42(9): 805-824.

[44] T. Y. Chen, P. L. Poon, X. Xie. METRIC: Metamorphic relation identification based on the category-choice framework. *Journal of Systems and Software*, Elsevier, 2016, 116: 177-190.

[45] J. Zhang, J. Chen, D. Hao. Search-based inference of polynomial metamorphic relations. *Proceedings of the 29th ACM/IEEE International Conference on Automated Software Engineering (ASE 2014)*, ACM Press, 2014, pp. 701-712.

[46] F. H. Su, J. Bell, C. Murphy. Dynamic inference of likely metamorphic properties to support differential testing. *Proceedings of the 10th International Workshop on Automation of Software Test (AST 2015), Co-located with the 37th IEEE International Conference on Software Engineering (ICSE 2015)*, IEEE Computer Society, 2015, pp. 55-59.

[47] C. Sun, Y. Liu, Z. Wang, W. K. Chan. μMT: A data mutation directed metamorphic relation acquisition methodology. *Proceeding of the First International Workshop on Metamorphic Testing (MET 2016), Co-located with the 38th IEEE International Conference on Software Engineering (ICSE 2016)*, IEEE Computer Society, 2016, pp.12-18.

[48] A. Gotlieb, B. Botella. Automated metamorphic testing. *Proceedings of the 27th Annual International Conference on Computer Software and Applications (COMPSAC 2003)*, IEEE Computer Society, 2003, pp. 34-40.

[49] H. Liu, X. Liu, T. Y. Chen. A new method for constructing metamorphic relations. *Proceedings of the 12th International Conference on Quality Software (QSIC 2012)*, IEEE Computer Society, 2013, pp. 59-68.

[50] G. Batra, J. Sengupta. An efficient metamorphic testing technique using genetic algorithm. *Proceedings of 5th International Conference on Information Intelligence, Systems, Technology and Management (ICISTM 2011)*, Springer, 2011, pp. 180-188.

[51] G. Dong, T. Guo, P. Zhang. Security assurance with program path analysis and

metamorphic testing. *Proceedings of the 4th International Conference on Software Engineering and Service Science (ICSESS 2013)*, IEEE Computer Society, 2013, pp. 193-197.

[52] T. Y. Chen, F. C. Kuo, Y. Liu. Metamorphic testing and testing with special values. *Proceedings of the 5th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD 2004)*, IEEE Computer Society, 2004, pp. 128-134.

[53] X. Xie, W. E. Wong, T. Y. Chen, B. Xu. Metamorphic slice: An application in spectrum-based fault localization. *Information and Software Technology*, Elsevier, 2013, 55(5): 866-879.

[54] G. Dong, S. Wu, G. Wang, T. Guo, Y. Huang. Security assurance with metamorphic testing and genetic algorithm. *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT 2010)*, IEEE Computer Society, 2010, pp. 397-401.

[55] C. Sun, G. Wang, B. Mu, H. Liu, Z. Wang, T. Y. Chen. A metamorphic relation-based approach to testing web services without oracles. *International Journal of Web Services Research*, IGI Global, 2012, 9(1): 51-73.

[56] W. K. Chan, S. C. Cheung, K. R. Leung. A metamorphic testing approach for online testing of service-oriented software applications. *International Journal of Web Services Research*, IGI Global, 2007, 4(2): 61-72.

[57] J. Mayer, R. Guderlei. On random testing of image processing applications. *Proceedings of the 6th International Conference on Quality Software (QSIC 2006)*, IEEE Computer Society, 2006, pp. 85-92.

[58] F. C. Kuo, S. Liu, T. Y. Chen. Testing a binary space partitioning algorithm with metamorphic testing. *Proceedings of the ACM Symposium on Applied Computing (SAC 2011)*, ACM Press, 2011, pp. 1482-1489.

[59] T. H. Tse, S. S. Yau. Testing context-sensitive middleware-based software applications. *Proceedings of the 28th Annual International Computer Software and Applications Conference (COMPSAC 2004)*. IEEE Computer Society, 2004, pp. 458-466.

[60] W. K. Chan, T. Y. Chen, S. C. Cheung, T. H. Tse, Z. Zhang. Towards the testing of power aware software applications for wireless sensor networks. *Proceedings of the 12th AdaEurope International Conference on Reliable Software Technologies (ICRST 2007)*, Springer, 2007, pp. 84-99.

[61] C. Sun, Z. Wang, G. Wang. A property-based testing framework for encryption programs. *Frontiers of Computer Science*, Springer, 2014, 8(3): 478-489.

[62] I. H. Witten, E. Frank, M. A. Hall, and J. P. Christopher. Data mining: practical machine learning tools and techniques. *Journal of Management Science*, Ubon Ratchathani University, 2005, 3(6): 92-96.

[63] K. Pei, Y. Cao, J. Yang, S. Jana. Deepxplore: Automated whitebox testing of deep learning systems. *Proceedings of the 26th Symposium on Operating Systems Principles (SOSP 2017)*, ACM Press, 2017, pp. 1-18.

[64] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, A. Swami. The limitations of deep learning in adversarial settings. *Proceedings of the 1st IEEE European Symposium on Security and Privacy (EuroS&P 2016)*, IEEE Computer Society, 2016, pp. 372-387.

[65] O. Bastani, Y. Ioannou, L. Lampropoulos, D. Vytiniotis, A. V. Nori, A. Criminisi. Measuring neural net robustness with constraints. *Proceedings of the 10th Annual Conference on Neural Information Processing Systems (**NIPS 2016)*, 2016, pp. 2613-2621.

[66] N. Carlini, D. Wagner. Towards evaluating the robustness of neural networks. *Proceedings of the IEEE Symposium on Security and Privacy (SP 2017)*, IEEE Computer Society, 2017, pp. 39-57.

[67] O. Bastani, Y. Ioannou, L. Lampropoulos, D. Vytiniotis, A. Nori, A. Criminisi. Measuring neural net robustness with constraints. *Proceedings of the International Conference on Machine Learning (ML 2017)*, IEEE Computer Society, 2017, pp. 2613-2621.

[68] S. Gu, L. Rigazio. Towards deep neural network architectures robust to adversarial examples. *Proceedings of the International Conference on Learning Representations (ICLR 2015)*, IEEE Computer Society, 2015, pp. 777-780.

[69] X. Huang, M. Kwiatkowska, S. Wang, M. Wu. Safety verification of deep neural networks. *Proceedings of the International Conference on Computer Aided Verification (CAV 2017)*, Springer, 2017, pp. 3-29.

[70] L. Li, W. L. Huang, Y. Liu, et al. Intelligence testing for autonomous vehicles: a new approach. *IEEE Transactions on Intelligent Vehicles*, IEEE Vehicular Technology Society, 2016, 1(2): 158-166.

[71] W. L. Huang, D. Wen, J. Geng, et al. Task-specific performance evaluation of UGVs: case studies at the IVFC. *IEEE Transactions on Intelligent Transportation Systems*, IEEE Intelligent Transportation Systems Society, 2014, 15(5): 1969-1979.

[72] L. Li, D. Wen, N. N. Zheng, et al. Cognitive cars: a new frontier for ADAS research. *IEEE Transactions on Intelligent Transportation Systems*, IEEE Intelligent Transportation Systems Society, 2012, 13(1): 395-407.

[73] O. J. Hernández. Evaluation in artificial intelligence: From task-oriented to ability-oriented measurement. *Artificial Intelligence Review*, Springer, 2017, 48(3): 397-447.

[74] B. D. Argall, S. Chernova, M. Veloso. A survey of robot learning from demonstration. *Robotics and autonomous systems*, Elsevier, 2009, 57(5): 469-483.

[75] A. Kuefler, J. Morton, T. Wheeler. Imitating driver behavior with generative adversarial networks. *Proceedings of IEEE Intelligent Vehicles Symposium (IV 2017)*. IEEE Computer Society, 2017: pp. 204-211.

[76] L. Li, Y. L. Lin, N. N. Zheng. Artificial intelligence test: A case study of intelligent vehicles. *Artificial Intelligence Review*, Springer, 2018: 50(3): 441-465.