


## Chapter 13

# Hybrid Multi-Objective Grey Wolf Search Optimizer and Machine Learning Approach for Software Bug Prediction

**Mrutyunjaya Panda**

 <https://orcid.org/0000-0001-5713-9220>

*Utkal University, India*

**Ahmad Taher Azar**

 <https://orcid.org/0000-0002-7869-6373>

*Faculty of Computers and Artificial Intelligence, Benha University, Benha, Egypt & College of Computer and Information Sciences, Prince Sultan University, Riyadh, Saudi Arabia*

### ABSTRACT

*Software bugs (or malfunctions) pose a serious threat to software developers with many known and unknown bugs that may be vulnerable to computer systems, demanding new methods, analysis, and techniques for efficient bug detection and repair of new unseen programs at a later stage. This chapter uses evolutionary grey wolf (GW) search optimization as a feature selection technique to improve classifier efficiency. It is also envisaged that software error detection would consider the nature of the error when repairing it for remedial action instead of simply finding it either faulty or non-defective. To address this problem, the authors use bug severity multi-class classification to build an efficient and robust prediction model using multilayer perceptron (MLP), logistic regression (LR), and random forest (RF) for bug severity classification. Both tests are performed on two software error datasets, namely Ant 1.7 and Tomcat.*

DOI: 10.4018/978-1-7998-5788-4.ch013

## 1. INTRODUCTION

The defect in the software product appears to be the one where the code developed does not meet the requirements of the end-user. Such a defect in the software product is referred to as a bug in the code that prevents the product from working properly. Looking into today's digital age, a human being totally dependent on software and a little bug (or defect) may have serious consequences in human life (Rana et al., 2015). Software testing and bug fixing is a precondition for software delivery that takes almost 50% to 60% of the total effort required for software development (Mishra & Mishra, 2009). Early identification of such software bugs (or defects) with defective modules saves a huge amount of effort and cost. Since the software product is too large, complex and versatile, checking all test classes after a code change is not a good idea, but software metrics such as code-based matrices and/or software change matrices can be used to test faulty software modules (Choudhary et al., 2018; Malhotra, 2016). While code-based matrices are used to find the size and complexity of the code, there is a change between two versions of the software in the software change matrices. Researchers use either the full or subset of features of these software matrices to build an efficient and accurate model to enhance the quality of the software (Gondra, 2008). It is important to develop a quality software product by accurately detecting software bugs at the early stage of the software development life cycle and then taking action to remove them.

Software development takes place at the following stages while handling software bugs in code (Grishma & Anjali, 2015): a) bug spotting, b) bug assortment, c) bug scanning, d) bug prediction and e) bug removal. The first stage refers to the occurrence of software defects, if any, by means of a code inspection, of such a defect and then a test to detect malfunctions in the software. After this initial identification step, the bug assortment is done for categorization followed by a bug scan to analyze the details of the present bug. Then, in the last two stages, make software bug prediction and removal using some promising methods to improve the quality of software.

It is observed that the prevalent seed of finding bugs in the software product includes: user coolness, ambiguous objectives and puzzling objectives, imperfect specification, lack of assets, lack of communication between group members and poor testing (Venkata et al., 2005; Rajkumar & Alagarsamy, 2013). As per IEEE 104 standard, software fault proneness may be classified as falling within any of the following (Mikyeong & Hong, 2014): (a) bug or defect resulting from the failure of the developer to meet the requirements of the end-user; (b) failure to comply with the previous required task of the software products; or incorrect results are received by the customer for each input entered; (c) error occurs by c Several techniques have been proposed by many researchers to detect software bugs based on their types in the recent past (Jia & Han, 2013).

Feature selection algorithms are dimensionality reduction methods often associated to data mining tasks of classification or clustering (Kaen & Algarni, 2019; Czibula et al., 2016; Hassanien et al., 2015; Xu et al., 2000; Emary et al., 2014a; Banu et al., 2014; Aziz et al., 2012, 2013a, 2020; Anter et al., 2020; Sayed et al., 2019). These methods provide a reduced set of the input features while preserving the relevant discriminatory information. Although different types of wrapper-based feature selection have recently been developed, meta-heuristic optimization algorithms have gained a great deal of attention for their crucial role in detecting optimal feature subsets for efficient classification or prediction in many applications because they are based on simple concepts and easily implementable (Pilla et al., 2019; Gorripotu et al., 2019; Najm et al., 2019; Ammar et al., 2019; Ben Smida et al., 2018; Kumar et al., 2015a; Hassanien et al., 2014a; Zhu & Azar, 2011). Some example are particle swarm optimization (Chengying et al., 2012; Kamal et al., 2020; Sallam et al., 2020; Inbarani et al., 2014a,b, 2015a; Jothi et

22 more pages are available in the full version of this document, which may be purchased using the "Add to Cart" button on the publisher's webpage:

[www.igi-global.com/chapter/hybrid-multi-objective-grey-wolf-search-optimizer-and-machine-learning-approach-for-software-bug-prediction/271044](http://www.igi-global.com/chapter/hybrid-multi-objective-grey-wolf-search-optimizer-and-machine-learning-approach-for-software-bug-prediction/271044)

## Related Content

---

### Ways to View the World: A Standard Ontology as the Reality Framework and the World Code

Azamat Abdoullaev (2008). *Reality, Universal Ontology and Knowledge Systems: Toward the Intelligent World* (pp. 28-57).

[www.irma-international.org/chapter/ways-view-world/28309](http://www.irma-international.org/chapter/ways-view-world/28309)

### Cellular Automata

Terry Bossomaier (2008). *Applications of Complex Adaptive Systems* (pp. 57-84).

[www.irma-international.org/chapter/cellular-automata/5134](http://www.irma-international.org/chapter/cellular-automata/5134)

### The Dynamics of Product Development in Software Startups: The Case for System Dynamics

Narendranath Shanbhag and Eric Pardede (2019). *International Journal of System Dynamics Applications* (pp. 51-77).

[www.irma-international.org/article/the-dynamics-of-product-development-in-software-startups/226252](http://www.irma-international.org/article/the-dynamics-of-product-development-in-software-startups/226252)

### Extrapolated Biogeography-Based Optimization (eBBO) for Global Numerical Optimization and Microstrip Patch Antenna Design

M. R. Lohokare, S.S. Pattnaik, S. Devi, B.K. Panigrahi, S. Das and J. G. Joshi (2010). *International Journal of Applied Evolutionary Computation* (pp. 1-26).

[www.irma-international.org/article/extrapolated-biogeography-based-optimization-ebbo/47064](http://www.irma-international.org/article/extrapolated-biogeography-based-optimization-ebbo/47064)

### The Management of Knowledge Risks: What do We Really Know?

Susanne Durst, Guido Bruns and Thomas Henschel (2016). *International Journal of Knowledge and Systems Science* (pp. 19-29).

[www.irma-international.org/article/the-management-of-knowledge-risks-what-do-we-really-know/160845](http://www.irma-international.org/article/the-management-of-knowledge-risks-what-do-we-really-know/160845)