Chapter 17
A Survey on the Applications of Swarm Intelligence to Software Verification

Tsutomu Kumazawa
Software Research Associates, Inc., Japan

Munehiro Takimoto
Tokyo University of Science, Japan

Yasushi Kambayashi
Nippon Institute of Technology, Japan

ABSTRACT

Applying swarm intelligence techniques to software engineering problems has appealed to both researchers and practitioners in the software engineering community. This chapter describes issues and challenges of its application to formal verification, which is one of the core research fields in software engineering. Formal verification, which explores how to effectively verify software products by using mathematical technique, often suffers from two open problems. One is the so-called state explosion problem that verification tools need too many computational resources to make verification feasible. The other problem is that the results of verification have often too much complexity for users to understand. While a number of research projects have addressed these problems in the context of traditional formal verification, recent researches demonstrate that Swarm Intelligence is a promising tool to tackle the problems. This chapter presents how Swarm Intelligence can be applied to formal verification, and surveys the state-of-the-art techniques.

DOI: 10.4018/978-1-7998-1659-1.ch017

INTRODUCTION

Evolutionary algorithms and Swarm Intelligence algorithms have gotten more and more attention to appeal both researchers and industrial practitioners. People become aware of the effectiveness and efficiency of these algorithms. Today, we have more than one hundred kinds of well-studied Swarm Intelligence algorithms (Xing & Gao, 2016; Boussaïd, Lepagnot, & Siarry, 2013), such as Genetic Algorithms (Hromkovič, 2001), Ant Colony Optimization (Dorigo & Stützle, 2004), and Fireworks Algorithms (Tan & Zhu, 2010). These algorithms share several common characteristics; they are stochastic, nondeterministic, and non-exhaustive. Although these characteristics do not guarantee that those algorithms always produce the optimal solutions, we can find good solutions for many combinatorial optimization problems (Hromkovič, 2001) that are otherwise unsolvable. Owing to the rapid improvements of these algorithms, people have successfully applied such algorithms not only to the well-known practical problems but also to the current theoretical computer science problems.

This chapter surveys a recent research field; the application of Swarm Intelligence algorithms to software engineering problems, especially to the formal software verification. Software engineering is a research area of computer science that studies the methodologies about how to create highly reliable and error-free software systems. We have yet to achieve the ultimate goal of software engineering. There are several reasons. The recent software systems are extremely complex. They run on diverse environments such as various kinds of smart devices, large factories, the Internets and safety-critical machines like robots and automobiles. Such diversity yields a lot of challenges not only in the practical fields but also theoretical field. In fact, many serious accidents relating to industrial software troubles have been reported to date. For example, Leveson analyzed several spacecraft accidents whose causes were involved with software failures (Leveson, 2004). Tamai reported the failure of software systems in financial industry in Japan, and its following lawsuit (Tamai, 2009; Tamai, 2015). These cases indicate that software accidents have high impacts not only on the specific industry and companies, but also on our entire society. Software engineers have tried to establish methodologies for developing safe and reliable systems, but it is still middle-of-the-road.

Software verification is one of the main research topics of software engineering. It aims at assuring reliability of software systems by providing methodologies for checking whether they have defects or not. In general, there are two approaches for verification; software testing and formal verification. Software testing confirms that the results of sample are the same as expected ones. Samples are representative behaviors of the target software. Each one of the samples is executed on the real system and its actual results are compared with its expected ones. If there is any difference between them, it is highly verisimilar that the target system has some problems with respect to the sample setting. Software testing is empirically known to detect serious failures efficiently and effectively. Therefore, it is widely accepted as the de facto standard in software industry. Unfortunately, software testing can show there are problems, but it fails to show the target system is flawless. On the other hand, the formal verification aims to prove the correctness of the system. Given formal descriptions of the target system and properties that are required to be true on the system, a verifier, i.e. a formal verification tool, proves that the system satisfies the properties. If the verifier can make a proof, we can conclude that the system is flawless. Research scientists have developed many methodologies that tackle with finding proofs automatically and efficiently. We focus on a formal verification technique, Model Checking (Clarke & Emerson, 1981), since it is one of the most successful formal verification techniques both in academics and in industry. A main difference between software testing and model checking is that while the former investigates