Applied Soft Computing xxx (2016) xxx–xxx

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

A Systolic Genetic Search for reducing the execution cost of regression testing

Martín Pedemonte [a,*], Francisco Luna [b], Enrique Alba [b]

[a] Instituto de Computación, Facultad de Ingeniería, Universidad de la República, Julio Herrera y Reissig 565, 11300 Montevideo, Uruguay
[b] Depto. de Lenguajes y Ciencias de la Computación, Univ. de Málaga, E.T.S. Ingeniería Informática, Campus de Teatinos, 29071 Málaga, Spain

ARTICLE INFO

Article history:
Received 31 December 2015
Received in revised form 16 June 2016
Accepted 4 July 2016
Available online xxx

Keywords:
Regression testing
Evolutionary algorithms
Parallel metaheuristics
GPU
CUDA

ABSTRACT

The Test Suite Minimization Problem (TSMP) is a $\mathcal{NP}$-hard real-world problem that arises in the field of software engineering. It consists in selecting a minimal set of test cases from a large test suite, ensuring that the test cases selected cover a given set of requirements of a piece of software at the same time as it minimizes the amount of resources required for its execution. In this paper, we propose a Systolic Genetic Search (SGS) algorithm for solving the TSMP. SGS is a recently proposed optimization algorithm capable of taking advantage of the high degree of parallelism available in modern GPU architectures. The experimental evaluation conducted on a large number of test suites generated for seven real-world programs and seven large test suites generated for a case study from a real-world program shows that SGS is highly effective for the TSMP. SGS not only outperforms two competitive genetic algorithms, but also outperforms four heuristics specially conceived for this problem. The results also show that the GPU implementation of SGS has achieved a high performance, obtaining a large runtime reduction with respect to the CPU implementation for solutions with similar quality. The GPU implementation of SGS also shows an excellent scalability behavior when solving instances with a large number of test cases. As a consequence, the GPU-based SGS stands as a state of the art alternative for solving the TSMP in real-world software testing environments.

1. Introduction

Software testing is one of the core activities of the software development process. It involves the execution of a piece of software to gather information for evaluating the quality of that software. In general, this execution uses test cases that are designed for exercising at least one feature (functional or non-functional) of the piece of software under evaluation. Although initially testing was considered necessary evil, it has become a key aspect of software development process. Testing has been reported to represent more than fifty percent of the cost of software development [1], while the total labor resources spent in testing range from 30 to 90% [2].

Regression testing is a testing activity performed to ensure that changes made to an existing piece of software do not introduce errors. During the evolution of a piece of software, the software tends to grow in size and complexity. This evolution also provokes that new test cases are continually generated and added to the test suite to validate the latest modifications to the piece of software. For this reason, the execution of the entire test suite can be impracticable. For instance, in a real-world test suite for regression testing from Cisco containing 2320 test cases [3], the runtime of each test case takes about 10–100 min, yielding a final total execution time of the test suite of around 5 weeks. In consequence, different approaches have been proposed in order to reduce the effort devoted to regression testing [4].

Search-based software engineering (SBSE) [5] is one recent field in Software Engineering that is based in applying search-based optimization techniques (as Evolutionary Algorithms, EAs) to software engineering problems. SBSE has been successfully used to solve problems from all the phases of the software development process, being software testing one of the most addressed issues [5]. In particular, the Test Suite Minimization Problem (TSMP) is a $\mathcal{NP}$-hard real-world software testing problem that arises in regression testing. It is based on reducing a large test suite by removing redundant test cases, ensuring that a set of test goals are satisfied [4]. The goal is to find a reduced test suite that minimizes the overall cost of the suite and that covers a given set of elements of the piece of software that is being tested.

As realistic software programs involve thousands of lines of code (and so the test suites used for their testing have thousands of test cases) exact algorithms are discarded for this problem because they could lead to huge computing times. Even metaheuristics may

* Corresponding author. Tel.: +598 27114244x1048.
E-mail addresses: mpedemon@fing.edu.uy (M. Pedemonte), flv@lcc.uma.es (F. Luna), eat@lcc.uma.es (E. Alba).

http://dx.doi.org/10.1016/j.asoc.2016.07.018
1568-4946/

be highly computationally expensive when addressing real-world TSMP instances. In order to tackle this problem properly, we make use of parallel metaheuristics [6]. These algorithms do not only allow to reduce the runtime of the algorithms, but also usually provide new enhanced search engines that could lead to improve the quality of results obtained by traditional sequential algorithms. Despite their advantages, there are very few works that use parallel metaheuristics for solving SBSE problems [7].

Systolic Genetic Search (SGS) is a new optimization algorithm that merges ideas from systolic computing and metaheuristics [8]. SGS was conceived to exploit the high degree of parallelism available in modern GPU architectures. It has already shown its potential for the knapsack problem, the massively multimodal deceptive problem, and the next release problem, finding optimal or near optimal solutions in short runtimes [8].

In the present article we investigate the use of a SGS algorithm for solving the cost-aware Test Suite Minimization Problem. Our first concern is to show whether SGS is effective for this problem. With this in mind, a comparative study on the numerical performance is conducted between SGS, two EAs and four heuristics specially designed for the problem at stake. Since SGS is explicitly designed for GPU architectures, a second issue is to evaluate if the GPU implementation of the proposed SGS is efficient and what kind of performance benefits can be obtained by such implementation with respect to a CPU implementation. To this end, we comparatively analyze the performance of the implementation on GPU of SGS and two evolutionary algorithms. Finally, our third concern is how well the number of test cases and test goals impact in the performance of the CPU and GPU implementations of SGS. We can summarize the contributions of this work as follows:

- It presents a new success of SGS for solving an optimization problem in a unexplored domain. The results obtained are not only relevant for the SGS but also are relevant for the cost-aware TSMP. Because SGS is highly effective for solving instances from seven real-world programs and a case study from a real-world program, without using any problem-specific knowledge, outperforming four heuristics specifically designed for this problem.
- It shows that the GPU implementation of SGS is able to achieve a high performance, obtaining a large runtime reduction compared to the sequential implementation for similar solutions. Moreover, the GPU-based SGS is the EA with the best performance of this study. In consequence, the GPU-based SGS is able to both obtain excellent quality solutions and execute in a short runtime.
- It shows that the GPU-based SGS has an excellent scalability behavior when solving instances with a larger number of test cases. On the other hand, when a larger number of test goals is considered, the performance of the GPU-based SGS is just minimally degraded.

This article is organized as follows. Section 2 reviews the preliminaries of this work. Then, in Section 3, we describe the SGS algorithm and how it is instantiated for tackling the TSMP. Section 4 presents state of the art heuristic techniques for the cost-aware TSMP that are used for comparison in this work. Then, Section 5 describes the details of the empirical study and analyzes the results of the experimental evaluation. Threats to validity are discussed on Section 6. Then, Section 7 discusses the related papers in the literature. Finally, in Section 8, we outline the conclusions of this work and suggest future research directions.

2. Preliminaries

In this section we present some background on the TSMP and on the architecture of the GPUs.

2.1. Test Suite Minimization Problem

Three different problem formulations have been proposed for test suite reduction in regression testing [4]. The Test Case Selection Problem consists in choosing a subset of the test suite based on which test cases are relevant for testing the changes between the previous and the current version of the piece of software. Another approach is known as the Test Case Prioritization Problem, which consists in finding an ordering of the test cases according to some specific criteria, such that test cases ordered first should be run first. In this formulation, it is assumed that all the test cases could be executed but the testing process could be stopped at some arbitrary point. As a consequence, if the test processing is stopped, the test cases that maximize the specific criteria used for ordering them have already been executed.

In the present work, we adopt the problem formulation known as Test Suite Minimization Problem (TSMP) [4,9]. TSMP belongs to the class of $\mathcal{NP}$-hard problems since it is equivalent to the Minimal Hitting Set Problem. It lies in reducing a test suite by eliminating redundant test cases, and the goal is to select a minimal set of test cases that cover a set of test goals and minimizes the amount of resources required for its execution. It is formally defined as follows.

Let $T = \{t_1, \ldots, t_n\}$ be a test suite with n test cases for a piece of software and $R = \{r_1, \ldots, r_m\}$ be the set of m test goals (requirements) that has to be covered with the test cases. Each test case covers several test goals and this relation is represented by a coverage matrix $M = [m_{ij}]$ of dimension $n \times m$, whose entries are either 0 or 1. If $m_{ij} = 1$ the test case i covers the test goal j, otherwise it does not covers the test goal. Also, each test case t_i has associated a positive cost c_i that measures the amount of resources required for its execution.

The single objective TSMP consists in finding a subset of test cases of the original test suite that covers all the test goals (100% of coverage) and minimizes its overall cost. The single objective TSMP can be formulated as the integer programming model presented in Eqs. (1)–(3), being x_i the binary decision variables of the problem that indicate whether the test case t_i is included or not in the reduced test suite.

$$\text{minimize} \quad \sum_{i=1}^{n} c_i x_i \tag{1}$$

$$\text{subject to:} \quad \sum_{i=1}^{n} m_{ij} x_i \geqslant 1, \quad \forall j = 1, \ldots, m \tag{2}$$

$$x_i \in \{0, 1\}, \quad \forall i = 1, \ldots, n \tag{3}$$

There are several alternatives that can be considered for the cost associated to the test cases. A classical option [10–15] is to consider all costs equal to one ($c_i = 1$, $\forall i = 1, \ldots, n$). In such case, the problem is equivalent to find a reduced test suite with a minimum number of test cases. However, the cost is often associated with a specific metric that is related to the cost of executing the test case, as the number of virtual code instructions that are run in a profiling tool [16] or the runtime measured in a particular platform [17]. This formulation is also known as the cost-aware TSMP.

Recently, the research community has also paid attention to the multi-objective TSMP. It has been proposed a bi-objective formulation [7,16] in which the conflicting objectives are the number of virtual code instructions executed in a profiling tool and the percentage of coverage of the test goals.

In spite of the existence of this multi-objective formulation, in our opinion the single-objective TSMP is still a relevant and important problem. For this reason, in this paper we adopt the

Please cite this article in press as: M. Pedemonte, et al., A Systolic Genetic Search for reducing the execution cost of regression testing, Appl. Soft Comput. J. (2016), http://dx.doi.org/10.1016/j.asoc.2016.07.018