Applied Soft Computing xxx (2016) xxx–xxx

Contents lists available at ScienceDirect

Applied Soft Computing

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

Design and analysis of evolutionary bit-length optimization algorithms for floating to fixed-point conversion

Q1 L.S. Rosa*, A.C.B. Delbem, C.F.M. Toledo, V. Bonato

The Institute of Mathematics Sciences and Computation, The University of São Paulo, Brazil

ARTICLE INFO

Article history:
Received 14 October 2015
Received in revised form 11 August 2016
Accepted 19 August 2016
Available online xxx

Keywords:
Fixed-point
Floating-point
Evolutionary genetic algorithms

ABSTRACT

Hardware designs need to obey constraints of resource utilization, minimum clock frequency, power consumption, computation precision and data range, which are all affected by the data type representation. Floating and fixed-point representations are the most common data types to work with real numbers where arithmetic hardware units for fixed-point format can improve performance and reduce energy consumption when compared to floating point solution. However, the right bit-lengths estimation for fixed-point is a time-consuming task since it is a combinatorial optimization problem of minimizing the accumulative arithmetic computation error. This work proposes two evolutionary approaches to accelerate the process of converting algorithms from floating to fixed-point format. The first is based on a classic evolutionary algorithm and the second one introduces a compact genetic algorithm, with theoretical evidence that a near-optimal performance, to find a solution, has been reached. To validate the proposed approaches, they are applied to three computing intensive algorithms from the mobile robotic scenario, where data error accumulated during execution is influenced by sensor noise and navigation environment characteristics. The proposed compact genetic algorithm accelerates the conversion process up to 10.2× against the state of art methods reaching similar bit precision and robustness.

1. Introduction

Q3

Hardware and software optimizations are crucial for embedded systems customized for specific applications. The optimization of these components can improve system performance and energy consumption. Based on the application behavior, a designer can exploit several optimizations to avoid an unnecessary or inappropriate use of hardware resources. For instance, scratchpad memory may be preferable instead of traditional cache memory in order to improve the energy efficiency of a system [1].

Optimizations related to arithmetic operations play a central role in a customization process, especially for embedded computing systems, which are highly sensitive to energy consumption and hardware cost. Important project decisions can be made only by knowing how many bits are necessary for their representations. For instance, this can enable a designer to choose whether it is necessary to have a dedicated arithmetic hardware unit as well as to determine the operations to be implemented on it. All these aspects are important to decide the hardware technology to be used. The authors in [2] present a survey evaluating hardware implementations for several applications.

The present paper introduces a multi-objective compact genetic algorithm (mo-cGA) based on a previous evolutionary algorithm proposed in [3] and on the compact genetic algorithm (cGA) [4]. This approach is applied to estimate bit-lengths for variables with real domain in algorithms according to a maximum error defined by the user.

The method is validated using classical algorithms for mobile robotics, where optimizations regarding performance, power consumption and size are important. The case study is reported over EKF-SLAM [5], Particle Filter (PF) [6] and the Gauss–Jordan Matrix Inversion (MI) [7] algorithms. In this context, the main contributions of this paper are:

- A mo-cGA applied to the bit-lengths estimation problem;
- A practical solution to accelerate the computationally heavy process of defining fixed-point arithmetic parameters, mitigating the whole procedure of design space exploration in hardware design;
- Theoretical evidence that the proposed mo-cGA has reached a near optimal performance, reducing the algorithm size impact

Q2 * Corresponding author at: The Institute of Mathematics Sciences and Computation, Avenida Trabalhador são-carlense, 400, 13566-590 São Carlos, São Paulo, Brazil.
E-mail addresses: leandrors@usp.br (L.S. Rosa), acbd@icmc.usp.br (A.C.B. Delbem), claudio@icmc.usp.br (C.F.M. Toledo), vbonato@icmc.usp.br (V. Bonato).

http://dx.doi.org/10.1016/j.asoc.2016.08.035
1568-4946/© 2016 Elsevier B.V. All rights reserved.

during the conversion process and accelerating up to 10.2×, when compared with the state of art methods of floating to fixed-point conversion, without compromising the bit precision and robustness.

The paper is organized as follows. Section 2 reviews related works with the floating to fixed-point conversion algorithm. Section 3 presents the bit-lengths estimation problem during the floating to fixed-point conversion of an algorithm. Section 4 presents the heuristic approach to the bit-lengths estimation problem. Section 5 presents a classical evolutionary approach, previously applied to the bit-lengths estimation problem, and introduces mo-cGA for the same problem. Section 6 presents a performance study comparing the proposed methods. Finally, Section 7 concludes the paper.

2. Related work

The conversion of an algorithm from floating to fixed-point demands the estimation of bit-lengths for each single variable. The aim is to find the smallest lengths that do not violate the maximum error defined by the user.

Recent works already use fixed-point representation to implement the Extended Kalman Filter (EKF) algorithm to solve the Simultaneous Localization and Mapping (SLAM) problem. The authors in [8] present a fixed-point implementation, which uses a constant bit-length for all variables. Another approach using fixed-point for SLAM is described in [9], where some variables have the bit-lengths defined according to physical constraints of a robot, and the remaining EKF-SLAM variables are left without optimization. These solutions apply fixed-point approach to reduce the computational cost of the whole system. However, further improvements could be achieved if the bit-lengths of each variable are properly defined following a given error.

Methods of floating to fixed-point conversion can be divided into two main classes: formal and non-formal methods. Formal methods are methods that, given the algorithm input range and a maximum acceptable error, give a solution (ranges or bit-lengths) that are mathematically proven to respect the maximum acceptable error as long as the input range is within the input range given to the method. Note that, in order to prove the ranges, a formal method might restrain the algorithm of having some structures, which are generally non-affine loops or unpredictable branches. On the other hand, non-formal methods cannot guarantee the maximum acceptable error obedience, but they generally do not imply constraints on the algorithm to be converted. It is worthy to note that these definitions do not imply optimality.

Approaches, orientated to Digital Signal Processors (DSP) applications, were proposed to convert from floating-point to fixed-point format [10–15] focusing DSP applications. These approaches are not applicable to algorithms with unpredictable feedbacks (e.g. `while` loops with stop condition statically indeterminable), leaving an open gap related to the types of algorithms that can be converted.

Formal method approaches for fixed to floating-point conversion, such as *Interval Arithmetic (IA)*, *Affine Arithmetic (AA)*, and *Symbolic Methods*, are also oriented to DPS applications. These methods present performance decay when applied on strongly non-affine computations, which is a problem mitigated by *Satisfiability-modulo Theory (SMT)* based methods [16,17]. Kinsman and Nicolici [16] present an SMT-based solution which allows estimating fixed or floating-point custom bit lengths given an error for DSP applications. [17] extends [16] to apply SMT on iterative computations (a.k.a. `for` loops) based on representing the error as $error = (knee, slope)$ instead of the general error magnitude, what mitigates "Catastrophic Cancellation" problems and is more robust than the previously cited methods.

As reported in [17], the capabilities of applying the method to iterative computations is restricted to the solver capabilities of solving the equation systems for the errors and precisions, which are limited. It is worth to note that in [17] all cases of study have its iteration spaces bounded by the user, based on mathematical formulations, what can not be generically applied, especially if we consider algorithm with unpredictable feedbacks, which are common in the autonomous robotics fields.

Boland and Constantinides [18] present a *Polynomial Algebraic Approach (PAA)*, which represents the computations as polynomials of the δ, suchas $|\delta| \leq \Delta = 2 - m$, and m is the mantissa size of a floating point representation. Then, the equations pass through a heuristic to define the bit-sizes. Furthermore, the Polynomial Algebraic Approach presents a promising scalability that is not presented in the SMT solvers [18,19].

Boland and Constantinides [19] present a detailed analysis of the IA, AA, *Polynomial Algebraic Approach using Handelman representations (Handelman)* [20] and *Taylor methods with Interval Remainder bounds (TwIR)* [21] showing that these approaches scalability fades quickly when applied to large algorithm. Further then, Boland and Constantinides [19] present a scalable approach to the bit estimation problem, which represents the source code operations by a pair of different polynomials, gathering the IA and Handelman approaches in order to balance the Handelman complexity (NP-Hard) with the IA complexity (linear), and also balancing the IA loose solutions (too many bits) with the Handelman solutions tightness.

The approaches presented in [18,19] calculate bit lengths for floating point representations given an algorithm, which is different from our floating to fixed-point conversion problem. Furthermore, [18] is not applicable to feedback computations, while [19] handles statically bounded iterative computations (`for` loops with bounds defined at compilation time) by unrolling the loops. Thus, these approaches cannot be applied in our scenario.

Sarbishei et al. [22] present an algorithm to estimate fixed-point bit lengths for an input algorithm with unpredictable feedbacks. This approach targets infinite impulse response filters, supposing that the application is Bounded-Input–Bounded-Output and that there is a user given parameter W which is greater than the filter order. Note that these two suppositions are not true in our scope, making this approach not applicable as well.

A fundamental limitation of these formal approaches is that they can only handle data flows which can be converted to static single assignment (SSA) form. In other words, they cannot handle algorithms which the branch conditions can depend on data values and loops with iteration space dynamically defined [23]. Boland and Constantinides [23] present an approach to contour these limitations based on substituting the stop conditions by a ranking function, aiming to estimate floating-point mantissa and exponent lengths. Even though, there are no scalable techniques to find such functions if the loop body contains non-linear functions, which is present in most of the autonomous robotic algorithms. If [23] tool fails in its attempt to find a ranking function, the user will be inquired for one.

Extensions of bit-lengths estimation for algorithms with unpredictable feedbacks are presented in [24]. The authors in [24] extend [15] to handle unpredictable feedbacks based on training sets. However, the proposed methods are computer-intensive and time-consuming for complex algorithms. The authors in [3] introduce improvements over [24] with an evolutionary algorithm (EA), reducing both conversion time and bit lengths.

In the present paper, a complete analysis is carried out on this previous EA. We also introduce a mo-cGA to solve this problem, which is based on an estimation of distribution algorithm proposed