Information Sciences 333 (2016) 21–43

Contents lists available at ScienceDirect

Information Sciences

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

neat Genetic Programming: Controlling bloat naturally

Leonardo Trujillo [a,*], Luis Muñoz [a], Edgar Galván-López [b], Sara Silva [c,d]

[a] *Tree-Lab, Doctorado en Ciencias de la Ingeniería, Departamento de Ingeniería Eléctrica y Electrónica, Instituto Tecnológico de Tijuana, Blvd. Industrial y Av. ITR Tijuana S/N, Mesa Otay C.P. 22500, Tijuana, BC, Mexico*
[b] *School of Computer Science and Statistics, Trinity College, Dublin, Ireland*
[c] *BioISI – Biosystems & Integrative Sciences Institute, Faculty of Sciences, University of Lisbon, Portugal*
[d] *NOVA IMS, Universidade Nova de Lisboa, Lisbon 1070-312, Portugal*

ARTICLE INFO

Article history:
Received 5 July 2015
Revised 21 October 2015
Accepted 2 November 2015
Available online 10 November 2015

Keywords:
Genetic programming
Bloat
NeuroEvolution of augmenting topologies
Flat operator equalization

ABSTRACT

Bloat is one of the most widely studied phenomena in Genetic Programming (GP), it is normally defined as the increase in mean program size without a corresponding improvement in fitness. Several theories have been proposed in the specialized GP literature that explain why bloat occurs. In particular, the Crossover-Bias Theory states that the cause of bloat is that the distribution of program sizes during evolution is skewed in a way that encourages bloat to appear, by punishing small individuals and favoring larger ones. Therefore, several bloat control methods have been proposed that attempt to explicitly control the size distribution of programs within the evolving population. This work proposes a new bloat control method called *neat*-GP, that implicitly shapes the program size distribution during a GP run. *neat*-GP is based on two key elements: (a) the NeuroEvolution of Augmenting Topologies algorithm (NEAT), a robust heuristic that was originally developed to evolve neural networks; and (b) the Flat Operator Equalization bloat control method, that explicitly shapes the program size distributions toward a uniform or flat shape. Experimental results are encouraging in two domains, symbolic regression and classification of real-world data. *neat*-GP can curtail the effects of bloat without sacrificing performance, outperforming both standard GP and the Flat-OE method, without incurring in the computational overhead reported by some state-of-the-art bloat control methods.

1. Introduction

Genetic Programming (GP) [9,10,21] is an evolutionary computation (EC) paradigm used for automatic program induction, its general goal is to generate computer programs through an evolutionary search. In its most common form, GP can be understood as a supervised learning algorithm that attempts to construct a syntactically valid expression using a finite set of basic functions and input variables, guided by a domain dependent objective or cost function [21]. In its original form [9], GP is characterized by two main features that distinguishes it from other EC techniques. Firstly, evolved solutions represent valid syntactic expressions or programs, that might be used as models, predictors, operators or classifiers. The ability of GP to construct syntactic expressions directly, without assuming a prior model, can allow it to produce highly interpretable solutions, that not only solve the

* Corresponding author. Tel.: +52 6643389391.
E-mail addresses: leonardo.trujillo@tectijuana.edu.mx (L. Trujillo), lmunoz@tectijuana.edu.mx (L. Muñoz), edgar.galvan@scss.tcd.ie (E. Galván-López), sara@fc.ul.pt (S. Silva).
URL: http://www.tree-lab.org (L. Trujillo)

http://dx.doi.org/10.1016/j.ins.2015.11.010
0020-0255/

problem but also provide insights into the problem domain. Secondly, GP uses a variable length encoding scheme, where the set of candidate solutions contains programs of different size and shape.

EC literature contains many examples of the problem solving abilities of GP, that illustrate the flexibility of the search paradigm [8]. Indeed, GP can be understood as a hyper-heuristic, an algorithmic approach for the automatic synthesis of heuristic approaches, a view that has strong theoretical background and real-world applicability [19]. Despite its success, GP is still not used as an off-the-shelf methodology [16], in the way that, for example, Support Vector Machines or Linear Regression are used. This lack of wider acceptance stems from some important pragmatic limitations of the GP approach.

In particular, syntactic search can be inefficient, due to its poor local structure and ill-defined fitness landscape (refer to [18] and [22] where the authors reviewed some of the main open issues in GP). Among them, one of the most studied problems is the bloat phenomenon, which occurs when program trees tend to grow unnecessarily large, without a corresponding increase in fitness [21,25]. In some sense, bloat seems to be an unavoidable consequence of the nature of the search space in GP and fitness driven search [11,12]. Moreover, bloat causes several undesirable side effects, since evaluating large programs is more time consuming, and large solutions are more difficult to interpret. Therefore, multiple approaches have been studied to deal with bloat, ranging from modifications of the basic search operators up to investigating the use of different search spaces, such as semantic space [6,33] and behavioral space [30].

This paper presents a novel approach toward bloat control, that leverages the insights of recent studies [23] and an algorithm originally developed for neuroevolution [28]. Silva [23] suggests that a powerful bloat control strategy is to induce a uniform distribution of program sizes within the evolving population. In particular, she proposed the Flat Operator Equalization bloat control method (Flat-OE), which explicitly forces the evolving population to follow a uniform distribution of program sizes, while the range of the distribution remains constant across all generations.

The contribution of our work is the development of a GP-based system based on the NeuroEvolution of Augmenting Topologies (NEAT) algorithm, which uses speciation to protect novel solution topologies and promote the incremental evolution of complexity [28]. In our recent work, we showed that NEAT can run bloat free, using a careful parametrization and system configuration [29]. However, it was unclear if the results obtained from neuroevolution could be replicated in a traditional GP domain.

The proposed algorithm is called *neat*-GP and it can be understood as a stripped down version of the original NEAT algorithm, which is adapted to the GP paradigm, designed to induce similar search dynamics as those shown by Flat-OE. Experiments are carried out using a tree-based representation and tested on several benchmark problems for both symbolic regression and classification. The results show that a *neat*-GP based search can outperform a standard GP search, based on test performance and especially with regards to solution size and depth. These results agree with those reported in [29], with the added advantage that the bloat control method does not incur in any additional computational cost exhibited by other state-of-the-art bloat control methods [23,26].

The remainder of this paper proceeds as follows. Section 2 provides a comprehensive overview on both the bloat phenomenon and the NEAT algorithm, discussing the theoretical causes of bloat, state-of-the-art bloat control methods and how bloat relates to NEAT. The proposed *neat*-GP algorithm is presented in Section 3, discussing different possible variants and detailing important algorithm features. The experimental work is presented in Section 4, discussing system setup, benchmarking and results. Finally, a summary and concluding remarks are outlined in Section 5.

2. Background

This section presents a comprehensive discussion of the most relevant background topics related to the current research paper.

2.1. Bloat

In what follows, the bloat phenomenon in GP is presented, focusing on theoretical aspects and state-of-the-art bloat control methods; a more complete survey on this topic can be found in [25,26].

2.1.1. Bloat theory and bloat control methods

The most well-established explanation of bloat is the fitness-causes-bloat theory (FCBT), originally developed by Langdon and Poli [11]. The FCBT assumes the following common features in a GP search: (a) there is a many-to-one mapping from syntactic space to fitness space; and (b) for a particular fitness value (e.g., the optimum), there are exponentially more large programs than there are small programs with the same fitness. Hence, if a particular fitness value is desired, there is a tendency toward larger, or bloated, programs during a GP search, simply because there are more of them within the search space. Indeed, stating that the search for fitness is the main cause of bloat is by now uncontroversial, since it is basically the underlying factor in all major bloat theories [25]. Moreover, recent works suggest that a GP search that does not consider fitness explicitly can in fact avoid bloat altogether, by searching for novelty instead of solution quality [13,30].

Currently, one of the most useful bloat theories is the crossover bias theory (CBT) [20]. Focusing on canonical GP, the CBT states that bloat is produced by the effect that subtree crossover has on the distribution of program size. While the average size of trees is not affected, the size distribution is skewed in a particular way, producing a large number of small trees. For most fairly challenging problems, small trees will have a relatively low fitness. This in consequence will bias the selection operator