Grammar–Guided Genetic Programming

Daniel Manrique
Inteligencia Artificial, Facultad de Informatica, UPM, Spain

Juan Ríos
Inteligencia Artificial, Facultad de Informatica, UPM, Spain

Alfonso Rodríguez-Patón
Inteligencia Artificial, Facultad de Informatica, UPM, Spain

INTRODUCTION

Evolutionary computation (EC) is the study of computational systems that borrow ideas from and are inspired by natural evolution and adaptation (Yao & Xu, 2006, pp. 1-18). EC covers a number of techniques based on evolutionary processes and natural selection: evolutionary strategies, genetic algorithms and genetic programming (Keedwell & Narayanan, 2005).

Evolutionary strategies are an approach for efficiently solving certain continuous problems, yielding good results for some parametric problems in real domains. Compared with genetic algorithms, evolutionary strategies run more exploratory searches and are a good option when applied to relatively unknown parametric problems.

Genetic algorithms emulate the evolutionary process that takes place in nature. Individuals compete for survival by adapting as best they can to the environmental conditions. Crossovers between individuals, mutations and deaths are all part of this process of adaptation. By substituting the natural environment for the problem to be solved, we get a computationally cheap method that is capable of dealing with any problem, provided we know how to determine individuals' fitness (Manrique, 2001).

Genetic programming is an extension of genetic algorithms (Couchet, Manrique, Ríos & Rodríguez-Patón, 2006). Its aim is to build computer programs that are not expressly designed and programmed by a human being. It can be said to be an optimization technique whose search space is composed of all possible computer programs for solving a particular problem. Genetic programming's key advantage over genetic algorithms is that it can handle individuals (computer programs) of different lengths.

Grammar-guided genetic programming (GGGP) is an extension of traditional GP systems (Whigham, 1995, pp. 33-41). The difference lies in the fact that they employ context-free grammars (CFG) that generate all the possible solutions to a given problem as sentences, establishing this way the formal definition of the syntactic problem constraints, and use the derivation trees for each sentence to encode these solutions (Dounias, Tsakonas, Jantzen, Axer, Bjerregard & von Keyserlingk, D. 2002, pp. 494-500). The use of this type of syntactic formalisms helps to solve the so-called closure problem (Whigham, 1996). To achieve closure valid individuals (points that belong to the search space) should always be generated. As the generation of invalid individuals slows down convergence speed a great deal, solving this problem will very much improve the GP search capability. The basic operator directly affecting the closure problem is crossover: crossing two (or any) valid individuals should generate a valid offspring. Similarly, this is the operator that has the biggest impact on the process of convergence towards the optimum solution. Therefore, this article reviews the most important crossover operators employed in GP and GGGP, highlighting the weaknesses existing nowadays in this area of research. We also propose a GGGP system. This system incorporates the original idea of employing ambiguous CFG to overcome these weaknesses, thereby increasing convergence speed and reducing the likelihood of trapping in local optima. Comparative results are shown to empirically corroborate our claims.

BACKGROUND

Koza defined one of the first major crossover operators (KX) (1992). This approach randomly swaps subtrees in both parents to generate offspring. Therefore, it tends to disaggregate the so-called building blocks across the trees (that represent the individuals). The building blocks are those subtrees that improve fitness. This over-expansion has a negative effect on the fitness of the individuals. Also, this operator's excessive exploration capability leads to another weakness: an increase in the size of individuals, which affects system performance, and results in a lower convergence speed (Terrio & Heywood, 2002). This effect is known as bloat or code bloat.

There is another important drawback: many of the generated offspring are syntactically invalid as the crossovers are done completely at random. These individuals should not be part of the new population because they do not provide a valid solution. This seriously undermines the convergence process. Figure 1 shows a situation where one of the two individuals generated after Koza's crossover breaches the constraints established by a hypothetical grammar whose sentences represent arithmetic equalities.

The strong context preservative crossover operator (SCPC) avoids the problem of desegregation of building blocks (also called context) across the trees by setting severe (strong) constraints for tree nodes considered as possible candidates for selection as crossover nodes (D'haesler, 1994, pp. 379-407). A system of coordinates is defined to univocally identify each node in a derivation tree. The position of each node within the tree is specified along the path that must be followed to reach a given node from the root. To do this, the position of a node is described by means of a tuple of n coordinates $T = (b_1, b_2, ..., b_n)$, where n is the node's depth in the tree, and b_i indicates which branch is selected at depth i (counting from left to right). Figure 2 shows an example representing this system of coordinates.

Only nodes with the same coordinates from both parents can be swapped. For this reason, a subtree may possibly never migrate to another place in the tree. This limitation can cause serious search space exploration problems, as the whole search space cannot be covered unless each function and terminal appears at every possible coordinate at least once in any one individual in the population. This failure to migrate building blocks causes them to evolve separately in each region, causing a too big an exploitation capability, thereby increasing the likelihood of trapping in local optima (Barrios, Carrascal, Manrique & Ríos, 2003, pp. 275-293).

As time moves on, the code bloat phenomenon becomes a serious problem and takes an ever more prominent role. To avoid this, Crawford-Marks &

Figure 1. Incorrect operation of Koza's crossover operator
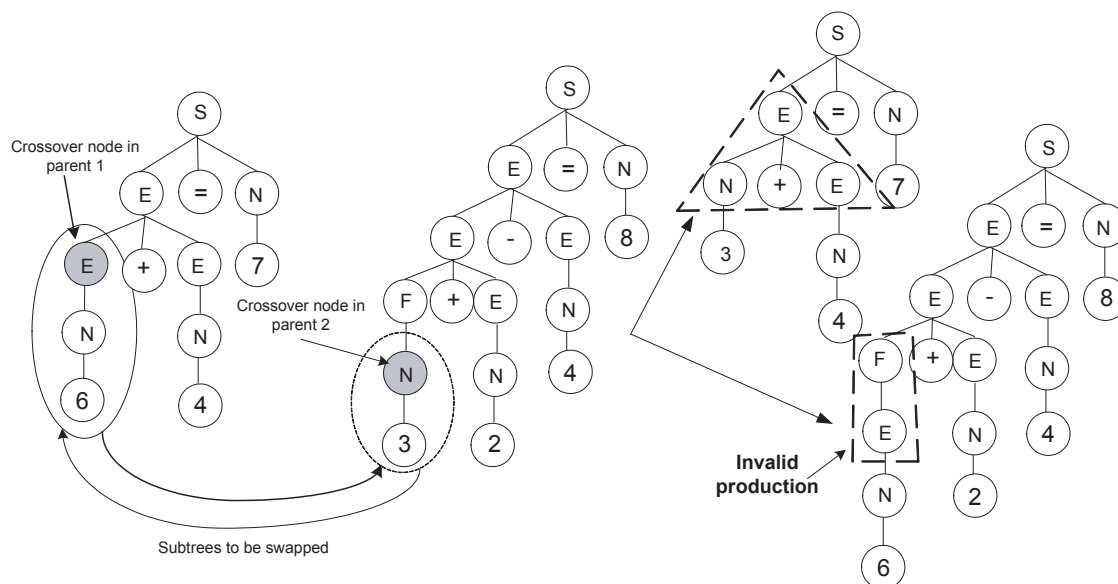
Related Content

Moth-Flame Optimization Algorithm Based Multilevel Thresholding for Image Segmentation

Abdul Kayom Md Khairuzzamanand Saurabh Chaudhury (2018). *Intelligent Systems: Concepts, Methodologies, Tools, and Applications (pp. 771-797).*

www.irma-international.org/chapter/moth-flame-optimization-algorithm-based-multilevel-thresholding-for-image-segmentation/205808

Decision Support Systems and their Application in Construction

Alaa Abdou, Moh'd Radaidehand John Lewis (2008). *Intelligent Information Technologies: Concepts, Methodologies, Tools, and Applications (pp. 1285-1303).*

www.irma-international.org/chapter/decision-support-systems-their-application/24342

KStore: A Dynamic Meta-Knowledge Repository for Intelligent BI

Jane Campbell Mazzagatti (2009). *International Journal of Intelligent Information Technologies (pp. 68-80).*

www.irma-international.org/article/kstore-dynamic-meta-knowledge-repository/2452

Distribution of Artificial Intelligence in Digital Games

Shaun Downeyand Darryl Charles (2015). *International Journal of Intelligent Information Technologies (pp. 1-14).*

www.irma-international.org/article/distribution-of-artificial-intelligence-in-digital-games/139467

Big Data Analytics With Machine Learning and Deep Learning Methods for Detection of Anomalies in Network Traffic

Valliammal Narayanand Shanmugapriya D. (2020). *Handbook of Research on Machine and Deep Learning Applications for Cyber Security (pp. 317-346).*

www.irma-international.org/chapter/big-data-analytics-with-machine-learning-and-deep-learning-methods-for-detection-of-anomalies-in-network-traffic/235048