Contents lists available at ScienceDirect

Expert Systems with Applications

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

Evolutionary construction and adaptation of intelligent systems

José M. Font [*], Daniel Manrique, Juan Ríos

Facultad de Informática, Universidad Politécnica de Madrid, Campus de Montegancedo, 28660 Boadilla del Monte, Madrid, Spain

A R T I C L E   I N F O

Keywords:
Evolutionary computation
Intelligent systems
Rule-based systems
Fuzzy rule-based systems
Artificial neural networks
Medical prognosis

A B S T R A C T

This paper introduces evolutionary techniques for automatically constructing intelligent self-adapting systems, capable of modifying their inner structure in order to learn from experience and self-adapt to a changing environment. These evolutionary techniques comprise an evolutionary system that is engineered by grammar-guided genetic programming, enabling the development of sub-symbolic and symbolic intelligent systems: artificial neural networks and knowledge-based systems, respectively. A context-free-grammar based codification system for artificial neural networks and rules, an initialization method and a crossover operator have been designed to properly balance the exploration and exploitation capabilities of the proposed system. This speeds up the convergence process and avoids trapping in local optima. This system has been applied to a medical domain: the detection of knee injuries from the analysis of isokinetic time series. The results of the evolved symbolic and sub-symbolic intelligent systems have been statistically compared with each other as part of a quantitative and qualitative performance analysis.

1. Introduction

Intelligent systems are built from knowledge extracted from a specific domain (Turbam, Aronson, & Liang, 2004). The knowledge is stored within the system in such a way that it can be processed to infer decisions that solve problems located within the domain. Many different kinds of intelligent systems from both the symbolic and sub-symbolic tracks of Artificial Intelligence (AI) have been successfully applied to knowledge areas ranging from industry and machine control to healthcare and education. Symbolic approaches intend to reproduce intelligent behaviour in a system built on a symbol-based representation of the world that is close to human reasoning. Sub-symbolic approaches try to imitate human cognition processes to reproduce intelligent behaviour without using symbols as an intermediate representation of real-world objects (Jones, 2008).

Knowledge engineering is the name given to the area of symbolic AI that acquires and formalizes specific knowledge from a domain, which is stored in a knowledge base (Gomez, Juristo, Montes, & Pazos, 1997). The use of this kind of systems is confined to the domain for which they were designed, for example, diagnosis of a particular disease. Linguistic rules are a symbolic knowledge representation able to act as an interface between human way of reasoning and computer processing. These rules use linguistic terms and have an antecedent-consequent structure, composed of easy-to-understand conditional sentences in the form “if *X* then *Y*”. These rules are also called inference rules because they are able to infer conclusions from current knowledge gathered from the environment. Applications of rule-based systems (RBS) are diverse and usually connected with solving real-world problems: medicine (Ohsaki, Yokoi, Abe, Tsumoto, & Yamaguchi, 2006), agriculture (Lee, Wu, & Wei, 2008), machine control (White & Lakany, 2008), diagnosis (Alonso, Caraça-Valente, González, & Montes, 2002), teaching and education (Canales, Peña, Peredo, Sossa, & Gutiérrez, 2007). Most of these areas use RBS to support human decision making. In knowledge areas where there is no need for human interaction, like machine control, RBS are designed to autonomously manage processes or machines. In the other areas, RBS are executed within a changing environment where systems may evolve over time. Take, for example, a RBS that is able to detect whether or not a patient has an illness considering the status of several symptoms within that patient’s organism. Those features are defined by domain experts during the system design phase according to the latest knowledge of the illness. When that knowledge is updated due to new results from medical research into the disease, the developed RBS, as well as the domain knowledge, becomes obsolete.

Artificial neural networks (ANN) are one of the most representative techniques of sub-symbolic AI. An ANN is an abstraction that simulates biological neural systems through mathematical models (Hassoun, 1995). This technique has been successfully applied to pattern recognition (Chen, Tai, Wang, Deng, & Chen, 2008) and classification problems (García-Pedrajas, Hervás-Martínez, & Ortíz-Boyer, 2005). The main disadvantage of an ANN is the

[*] Corresponding author. Tel.: +34 913366907; fax: +34 913524819.
E-mail addresses: jm.font@upm.es, erfont@gmail.com (J.M. Font), dmanrique@fi.upm.es (D. Manrique), jrios@fi.upm.es (J. Ríos).

0957-4174/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2010.04.070

process of designing of its inner structure and its training. An ANN has a purpose-specific structure, and it is trained by trial-and-error. The complexity of this process grows exponentially as the size of the network increases. As long as real-world problems require big networks, building ANNs to solve such problems will be computationally very costly.

Evolutionary computation (EC) is an area of the AI made up of a set of techniques based on the natural evolution and selection of the species proposed by Charles Darwin, as well as on Gregor Mendel's discoveries in genetics (Holland, 1992). An EC system evolves populations of individuals starting from an initial randomly generated population that evolves through a fixed number of generations (De Jong, 2006). Generation by generation, the population is subjected to selection, crossover, mutation and replacement operators until it reaches the individual that codifies the optimal solution to the problem. EC has been successfully applied to solve searching and optimization problems (Scully & Brown, 2008), such as the generation of both symbolic (Couchet, Font, & Manrique, 2008, 2009) and sub-symbolic (Couchet, Manrique, & Porras, 2007; Manrique, Ríos, & Rodríguez-Patón, 2006) self-adapting intelligent systems. Self-adapting systems can be applied to several different problems, evolving and adapting in tune with the problems: they are robust systems (Podgorelec, Kokol, Stiglic, Hericko, & Rozman, 2005). Another advantage of generating intelligent systems through EC is that it avoids the bottlenecks within the knowledge-based and neural systems development process. In knowledge-based systems, this drawback is due to a knowledge elicitation process that is highly dependant on the domain expert. In neural systems, it is caused by the trial-and-error design process.

Genetic Programming (GP), proposed by John R. Koza in the early 1990s (Koza, 1992) (Langdon & Poli, 2002), is one of EC's most representative techniques. Its special feature is that it works with individuals that codify programs of non-fixed length (Michal, Ivry, Schalit-Cohen, Sipper, & Barash, 2007). Each individual in the GP population is a program that solves a given problem. A key drawback of GP is that it does not solve the closure problem (Koza, 1992): it can generate invalid individuals not belonging to the solution space during the evolutionary process. Processing these individuals increases the computational cost during the GP execution. In addition, the size of the GP individuals escapes control due to a phenomenon called code bloat (Panait & Luke, 2004).

Grammar-Guided Genetic Programming (GGGP) is a GP specialization aiming to solve the closure problem. It uses a Context-Free Grammar (CFG) to generate the language whose words are the whole set of individuals that codify a solution to a given problem (Whigham, 1995; Wong & Leung, 1995). Individuals are derivation trees of the CFG that, when the algorithm starts, are generated by a grammar-based initialization method. This method cannot generate invalid individuals because they are not contained in the language described by the CFG (O'Neil, 2003). The initialization method plays a very important role, as it influences the later convergence process of the evolutionary system (García-Arnau, Manrique, Ríos, & Rodríguez-Patón, 2007). The design of the crossover operator of a GGGP system is also responsible for solving the closure problem in such a way that the crossover of two valid individuals must generate a valid offspring. To avoid code bloat it is possible to set a parameter to control the size of the generated derivation trees in both the initialization process and the application of the crossover operator. One of the most representative operators is the Fair crossover. The Fair crossover solves this problem by controlling the size of the parts that the crossed individuals exchange (Crawford-Marks & Spector, 2002). Despite its simplicity, this control is very exhaustive and interferes with the operator's exploration capability. Another important crossover operator is Whigham's crossover (Whigham, 1995). This operator improves the exploration capability, as well as solving the closure problem. Even so, this operator still does not properly explore the search space, especially when working with ambiguous CFGs (Couchet, Manrique, Ríos, & Rodríguez-Patón, 2007; Hoai & McKay, 2002).

This paper presents an improved GGGP system. It includes both a grammar-based initialization method and a grammar-based crossover operator whose combined application boosts the search space exploration and exploitation capabilities. This system takes advantage of the grammar's ambiguity, a property whereby different derivation trees represent the same sentence. This is a powerful and very efficient EC technique. The proposed system has been specifically designed to be able to construct and automatically self-adapt (robust) symbolic and sub-symbolic intelligent machines: rule- and fuzzy rule-based systems, and artificial neural networks. To do so, we have developed three different CFGs and codification systems. These automatically generated intelligent systems have been applied to a real-world problem located within the medical domain knowledge: the study of isokinetic time series for injury detection, diagnosis, rehabilitation and injury prevention (Couchet et al., 2008; Gioftsidou et al., 2008). All this is based on the analysis of isokinetic curves output by an isokinetic dynamometer on which patients exercise a muscle or joint.

2. Evolving intelligent systems

The proposed process for evolving intelligent systems is based on GGGP. Given a context-free grammar G, defined as a string-rewiring system comprising a 4-tuple $G = (\Sigma_N, \Sigma_T, S, P)/\Sigma_N \cap \Sigma_T = \emptyset$, where Σ_N is the alphabet of non-terminal symbols, Σ_T is the alphabet of terminal symbols, S represents the start symbol or axiom of the grammar, and P is the set of production rules, written in Backus-Naur Form. The individuals that are part of the genetic population codify a sentence of the language generated by the grammar as a derivation tree. This tree is a possible solution to a problem. Any GGGP system is able to find solutions to any problem whose syntactic constraints can be formally defined by a CFG. The initialization method and the crossover operator have to operate according to these constraints. This assures that they will be enforced during the evolutionary process and prevents individuals not belonging to the CFG-based language from being generated. This leads to the definition of both a grammar-based initialization method and a crossover operator.

2.1. Initialization method for evolving intelligent systems

The initialization method for evolving intelligent systems (IMEIS) is a grammar-based procedure for generating the initial population. To do this, it chooses the productions that generate individuals belonging to the CFG-based language not exceeding a fixed maximum depth rather than at random. This prevents code bloat. For clarity's sake, we have defined a CFG $G_{RBS} = (\Sigma_N, \Sigma_T, S, P)$ shown in Table 1a for use as an example throughout this section.

G_{RBS} generates a language composed of rule-based knowledge bases capable of detecting human knee injuries according to four knee-related input features. The input data used for prognosis has been taken from series of knee exercises performed by several patients. These data are composed of four variables named *TorMax, TorMin, angTorMax* and *angTorMin*. These variables refer to the maximum and minimum values of the torque (strength) exerted by the patient during the exercise and their related knee angle values. Every possible derivation tree generated by the grammar is composed of a non-fixed number of rules of the form *if ANTECEDENT then CONSEQUENT*. Each rule can have multiple (and at least one) antecedents but only one consequent. The consequent states the output of the rule, that is, the prognosis: *normal* or *injured*.