Design of Explicitly or Implicitly Parallel Low-resolution Character Recognition Algorithms by Means of Genetic Programming

Giovanni Adorni*, Stefano Cagnoni

Dipartimento di Ingegneria dell’Informazione, Università di Parma
*Dipartimento di Informatica, Sistemistica e Telematica, Università di Genova
Email: adorni@dist.unige.it, cagnoni@ce.unipr.it

Abstract. The paper describes two approaches to low-resolution character recognition that are either implicitly or explicitly based on the SIMD (Single Instruction Multiple Data) computation paradigm. In both approaches, a set of binary classifiers have been designed by means of evolutionary computation techniques.

The first approach is based on cellular programming. Each classifier is implemented as a non-uniform cellular automaton, whose state-transition functions are evolved using the algorithm proposed by Sipper. In the second approach, Sub-machine-code Genetic Programming is used to design extremely efficient classifiers that process in parallel binary data packed into a long integer.

The two methods were tested on an extensive set of low-resolution binary patterns (of size 13 × 8 pixels). The paper describes and compares the results obtained by the two approaches.

1 Introduction

Genetic Programming (GP) [1,2] is an Evolutionary Computation (EC) paradigm in which the phenotype of each individual in the population is a program. Several different GP paradigms have been described, the most commonly-used being the one originally proposed by Koza [1], where programs are encoded as syntactic trees or, equivalently, prefix-notation LISP-like functions.

Genetic Programming is usually computationally very intensive. The higher requirements in terms of computing resources with respect, for example, to Genetic Algorithms (GAs) are essentially due to the much wider search space, as convergence is concerned. As regards implementation, the tree-like encoding of solutions makes the decoding process during fitness evaluation, as well as crossover and mutation operators, much more complicated and CPU-demanding. Therefore, there is great interest in developing new variants of GP that improve the computational efficiency of the paradigm. On the other hand, the increasing request for programs that perform real-time tasks has suggested that GP be used not only as a viable way to obtain the desired program output but also to optimize program efficiency. Several approaches that imply some level of parallelization, both in the execution of the evolutionary algorithm or in the structure of the programs resulting from it, have been proposed. In this paper we will focus on two of them: Cellular Programming (CP) and Sub-machine-code GP (SmcGP), describing results obtained by the two
approaches in low-resolution character classification. Both approaches rely, even if at different levels, on the SIMD (Single Instruction Multiple Data) computation paradigm.

Cellular Programming [3] is specifically dedicated to SIMD architectures, and in particular to the cellular automata (CA) paradigm [4]: in CP, a look-up table that describes the state-transition function of a non-uniform cellular automaton is evolved for each cell of the automaton. As in the actual working of CAs, that relies on local interactions of cells with the neighboring ones, evolution in CP takes place through local interactions within the genotypes of neighboring cells. Programs evolved by CP require to be run on special hardware [5] to get the most out of them in terms of computation speed, and can only be simulated on sequential CPUs.

On the contrary, Sub-machine-code GP (SmcGP) [6,7] is aimed at exploiting the intrinsic parallelism of sequential CPUs. Inside a sequential 32-bit CPU, for example, each bitwise operation on integers is performed by concurrently activating 32 logical gates of the same kind. Therefore, the application of a sequence of bitwise logical operators to an integer can be seen as the parallel execution of a program on 32 1-bit operands in parallel, according to the SIMD paradigm. Such an approach can speed up Genetic Programming (GP) by making it possible to evaluate several fitness cases at the same time, if the teaching input is a 1-bit value. When the input patterns of the fitness cases are long bit strings, SmcGP makes it possible to process a 32-bit slice of each input at the same time. In this case, besides speeding up evolution, SmcGP produces programs that are intrinsically parallel and therefore also computationally very efficient. Because of these properties, SmcGP can be used effectively in all those applications in which the same operations must be performed on a large number of small chunks of data, that can therefore be packed into a long integer variable. This is the case for binary-pattern recognition or binary-image processing, especially when processing of 2D patterns can be performed line-wise or block-wise.

In the following we describe how the two evolutionary approaches have been used to design character classifiers and compare and comment the results that were achieved on a wide data set of low-resolution characters from a license-plate recognition application.

2 Character recognition in motor-vehicle license-plates

At first glance, character classification for license-plate recognition [8,9] seems to have much in common with traditional Optical Character Recognition (OCR) applications. However, differently from OCR, in which character classification occurs after a text has been scanned at high resolution, license-plate recognition analyses only very low-resolution patterns, possibly altered by optical distortions and perspective effects. Therefore, classification of license-plate characters is usually a much more critical task with respect to classification of characters contained in printed documents.