Solving NP-Complete Problems with Networks of Evolutionary Processors

Juan Castellanos[1], Carlos Martín-Vide[2], Victor Mitrana[3], and Jose M. Sempere[4]

[1] Dept. Inteligencia Artificial - Facultad de Informática, Universidad Politécnica de Madrid - Campus de Montegancedo, Boadilla del Monte, 28660 Madrid, Spain - jcastellanos@fi.upm.es - http://www.dia.fi.upm.es

[2] Research Group in Mathematical Linguistics - Rovira i Virgili University - Pça. Imperial Tàrraco 1, 43005 Tarragona, Spain - cmv@correu.urv.es

[3] Faculty of Mathematics, University of Bucharest[‡] - Str. Academiei 14, 70109 Bucharest, Romania - mitrana@funinf.math.unibuc.ro

[4] Department of Information Systems and Computation - Polytechnical University of Valencia, - Valencia 46071, Spain - jsempere@dsic.upv.es

Abstract. We propose a computational device based on evolutionary rules and communication within a network, similar to that introduced in [4], called network of evolutionary processors. An NP-complete problem is solved by networks of evolutionary processors of linear size in linear time. Some furher directions of research are finally discussed.

1 Introduction

A basic architecture for parallel and distributed symbolic processing, related to the Connection Machine [8] as well as the Logic Flow paradigm [5], consists of several processors, each of them being placed in a node of a virtual complete graph, which are able to handle data associated with the respective node. Each node processor acts on the local data in accordance with some predefined rules, and, then local data becomes a mobile agent which can navigate in the network following a given protocol. Only such data can be communicated which can pass a filtering process. This filtering process may require to satisfy some conditions imposed by the sending processor, by the reveiving processor or by both of them. All the nodes send simultaneously their data and the receiving nodes handle also simultaneously all the arriving messages according to some strategies, see, e.g., [6, 8].

Starting from the premise that data can be given in the form of strings, [4] introduces a concept called network of parallel language processors in the aim of investigating this concept in terms of formal grammars and languages. Networks of language processors are closely related to grammar systems, more specifically to parallel communicating grammar systems [3]. The main idea is that one can

[‡] Research supported by the Dirección General de Enseñanza Superior e Investigación Cientifica, SB 97–00110508

place a language generating device (grammar, Lindenmayer system, etc.) in any node of an underlying graph which rewrite the strings existing in the node, then the strings are communicated to the other nodes. Strings can be successfully communicated if they pass some output and input filter.

In the present paper, we modify this concept in the following way inspired from cell biology. Each processor placed in a node is a very simple processor, an evolutionary processor. By an evolutionary processor we mean a processor which is able to perform very simple ooperations, namely point mutations in a DNA sequence (insertion, deletion or substitution of a pair of nucleotides). More generally, each node may be viewed as a cell having a genetic information encoded in DNA sequences which may evolve by local evolutionary events, that is point mutations. Each node is specialized just for one of these evolutionary operations. Furthermore, the data is each node is organized in the form of multisets, each copy being processed in parallel such that all the possible evolutions events that can take place do actually take place.

These networks may be used as language (macroset) generating devices or as computational ones. Here, we consider them as computational mechanisms and show how an NP-complete problem can be solved in linear time.

It is worth mentioning here the similarity of this model to that of a P system, a new computing model inspired by the hierarchical and modularized cell structure recently proposed in [11].

2 Preliminaries

We start by summarizing the notions used throughout the paper. An *alphabet* is a finite and nonempty set of symbols. Any sequence of symbols from an alphabet V is called *string (word)* over V. The set of all strings over V is denoted by V^* and the empty string is denoted by ε. The length of a string x is denoted by $|x|$.

A *multiset* over a set X is a mappingg $M : X \longrightarrow \mathbf{N} \cup \{\infty\}$. The number $M(x)$ expresses the number of copies of $x \in X$ in the multiset M. When $M(x) = \infty$, then x appears arbitrarily many times in M. The set *supp*(M) is the support of M, i.e., *supp*$(M) = \{x \in X \mid M(x) > 0\}$. For two multisets M_1 and M_2 over X we define their union by $(M_1 \cup M_2)(x) = M_1(x) + M_2(x)$. For other operations on multisets the reader may consult [1].

A *network of evolutionary processors* (NEP for short) of size n is a construct

$$\Gamma = (V, N_1, N_2, \ldots, N_n),$$

where:

– V is an alphabet,
– for each $1 \leq i \leq n$, $N_i = (M_i, A_i, PI_i, FI_i, PO_i, FO_i)$ is the i-th evolutionary node processor of the network. The parameters of every processor are:
  • M_i is a finite set of evolution rules of one of the following forms only
    - $a \rightarrow b$, $a, b \in V$ (substitution rules),
    - $a \rightarrow \varepsilon$, $a \in V$ (deletion rules),