On the computational power of networks of polarized evolutionary processors

Fernando Arroyo\textsuperscript{a}, Sandra Gómez Canaval\textsuperscript{a}, Victor Mitrana\textsuperscript{a,∗}, Stefan Popescu\textsuperscript{b}

\textsuperscript{a} Department of Information Systems, University College of Computer Science, Polytechnic University of Madrid, Ctra. de Valencia km. 7, 28031 Madrid, Spain
\textsuperscript{b} Faculty of Mathematics and Computer Science, University of Bucharest, Str. Academiei 14, RO-010014 Bucharest, Romania

\begin{abstract}
We consider a new variant of networks of evolutionary processors which seems more suitable for a software and hardware implementation. Each processor as well as the data navigating throughout the network are now considered to be polarized. While the polarization of every processor is predefined, the data polarization is dynamically computed. Consequently, the protocol of communication is naturally defined by this polarization. We show that tag systems can be simulated by these networks with a constant number of nodes, while Turing machines can be efficiently simulated by these networks with a number of nodes depending linearly on the tape alphabet of the Turing machine. We also propose a simulation of Turing machines by networks with a constant number of nodes, which is reflected in an increase of the computation time. Finally, we show that every network can be simulated by a Turing machine and discuss the time complexity of this simulation.
\end{abstract}

© 2016 Elsevier Inc. All rights reserved.

1. Introduction

Networks of evolutionary processors (NEP) form a class of highly parallel and distributed computing models inspired and abstracted from the biological evolution. Informally, a network of evolutionary processors consists of a virtual graph in which each node hosts a very simple processor called evolutionary processor. By an evolutionary processor we mean a mathematical construction which is able to perform very simple operations inspired by the point mutations in DNA sequences (insertion, deletion or substitution of a single base pair). Following an informal parallelism with the natural process of evolution, each node may be viewed as a cell having genetic information encoded in DNA sequences which may evolve by local evolutionary events, that is point mutations. Each processor, which is specialized just for one of these evolutionary operations, acts on the local data and then local data becomes a mobile agent which can navigate in the network following a given protocol. Only that data which is able to pass a filtering process can be communicated. This filtering process may require to satisfy some conditions imposed by the sending processor, by the receiving processor or

\textsuperscript{∗} A preliminary version of this work has been published in the Proc. of the 8th International Conference on Language and Automata Theory and Applications-LATA 2014, LNCS 8370, 2014, 101-112.

\textsuperscript{☆} Corresponding author.

E-mail addresses: farroyo@etsisi.upm.es (F. Arroyo), sgomez@etsisi.upm.es (S.G. Canaval), victor.mitrana@upm.es (V. Mitrana), stefipopescu2004@yahoo.com (S. Popescu).

http://dx.doi.org/10.1016/j.jic.2016.06.004
0890-5401/© 2016 Elsevier Inc. All rights reserved.
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.

It is worth mentioning that NEPs resemble a pretty common architecture for parallel and distributed symbolic processing, related to the Connection Machine [8] which was defined as a network of microprocessors in the shape of a hypercube. Each microprocessor was very simple, processing one bit per unit time. Also, it is closely related to the tissue-like P systems [14] in the membrane computing area [20].

NEPs as language generating devices and problem solvers have been considered in [3] and [15], respectively. They have been further investigated in a series of subsequent works. NEPs as accepting devices and problem solvers have been considered in [13]; later on, a characterization of the complexity classes \( \text{NP}, \text{P}, \text{and PSPACE} \) based on accepting NEPs has been reported in [11]. Universal NEPs and some descriptional complexity problems are discussed in [10]. The reader interested in a survey of the main results regarding NEPs is referred to [12].

Software implementations of NEPs have been reported, see, e.g., [4,5,17], most of them in JAVA. They encountered difficulties especially in the implementation of filters. The main idea to simulate the non-deterministic behavior of NEPs has been to consider a safe-thread model of processors, that is, to have each rule and filter in a thread, respectively. Clearly, the threads corresponding to the filters are much more complicated than those associated with the evolutionary rules. Configuration changes in a NEP are accomplished either by a communication step or by an evolutionary step, but these two steps may be realized in any order. This suggests that evolution or communication may be chosen depending on the thread model of processor [5]. It is worth mentioning that the threads do not necessary synchronize their steps (evolutionary or communication), but this does not affect the original definition of an NEP as the itineraries of data through an NEP do not interfere with each other.

The input and output filters are implemented as threads extending the \texttt{Runnable} interface. Therefore a processor is the parent of a set of threads, which use all objects from that processor in a mutual exclusion region. When a processor starts to run, it starts in a cascade way the rule threads and filter threads. As one can see, the filters associated with processors, especially if there are both input and output filters, seem to be hardly implementable. Consequently, it would be of interest to replace the communication based on filters among processors by another protocol. A first attempt was to move filters from each node to the edges between the nodes, see, e.g., [6]. Although this variant seems to be theoretically simpler, the attempts towards an implementation have encountered similar difficulties due to the fact that the filters associated with edges are similar to those associated with nodes.

Work [1] considers a new variant of NEP with the aim of proposing a new type of filtering process and discusses the potential of this variant for solving hard computational problems. The main and completely new feature of this variant is the valuation mapping which assigns to each word an integer value, depending on the values assigned to its symbols. Actually, we are not interested in computing the exact value of a word, but just the sign of this value. As we shall see in the proof of Theorem 4, the polarization of a word can be computed without having its value. By means of this valuation, one may metaphorically say that the words are electrically polarized. Thus, if the nodes are polarized as well, the words migration from one node to another through the channel between the two cells seems to be more natural and easier to be implemented.

We consider here a slightly more general variant of networks of polarized evolutionary processors (NPEP) and investigate its computational power. Although the communication protocol based on the polarized processors and the valuation function seems to offer less control, the new variant is still computationally complete. We show that NPEP with a constant number of processors, namely 15, are computationally complete by devising a method for simulating 2-tag systems. As a 2-tag system can efficiently simulate any deterministic Turing machine but not nondeterministic ones, we propose a simulation of nondeterministic Turing machines with NEP which maintains the working time of the Turing machine. That is, every language accepted by a one-tape nondeterministic Turing machine in time \( O(f(n)) \) can be accepted by an NPEP in time \( O(f(n)) \). Unlike the simulation of a 2-tag system, the size of an NPEP simulating an arbitrary Turing machine depends linearly on the number of tape symbols of the Turing machine, but preserves the time complexity. We also propose a simulation of Turing machines by networks with a constant number of nodes, but this constant size reflects in an increase of the computation time. More precisely, the simulation of a Turing machine, with working alphabet \( \mathcal{U} \), that works in \( O(f(n)) \) time can be done by an NPEP of constant size, namely 39, in \( O((f(n) \cdot \text{card}(\mathcal{U}))^2) \) time. Finally, we show that every NPEP, with the input alphabet \( \mathcal{V} \) and the valuation mapping \( \varphi \), can be simulated by a Turing machine in \( O((f(n)(f(n) + n)^2) \) time.

The paper is organized as follows. After a preliminary section in which we recall several concepts and notations and give the definition of the model, we present a simulation of 2-tag systems by NPEP of constant size. Afterwards, we present a simulation of an arbitrary Turing machine by NPEP of a size depending linearly on the working alphabet of the Turing machine. On the other hand, we show that such a simulation can be accomplished by an NPEP of constant size, but with an increase in the computation time. Then we show that every network can be simulated by a Turing machine and discuss the time cost. Finally, we discuss some related works and propose some open problems.

2. Preliminaries

Throughout the paper we assume that the reader is familiar with the basic notions of formal language theory. We start by summarizing the main notions and notations used in this work; for all unexplained notions the reader is referred to [22].
دریافت فوری متن کامل مقاله

امکان دانلود نسخه تمام متن مقالات انگلیسی
امکان دانلود نسخه ترجمه شده مقالات
پذیرش سفارش ترجمه تخصصی
امکان جستجو در آرشیو جامعی از صدها موضوع و هزاران مقاله
امکان دانلود رایگان ۲ صفحه اول هر مقاله
امکان پرداخت اینترنتی با کلیه کارت های عضو شتاب
دانلود فوری مقاله پس از پرداخت آنلاین
پشتیبانی کامل خرید با بهره مندی از سیستم هوشمند رهگیری سفارشات