Theoretical Computer Science 623 (2016) 146–159

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Computing with viruses

Xu Chen [a], Mario J. Pérez-Jiménez [b], Luis Valencia-Cabrera [b], Beizhan Wang [a], Xiangxiang Zeng [c,*]

[a] School of Software, Xiamen University, Xiamen 361005, Fujian, People's Republic of China
[b] Research Group on Natural Computing, Department of Computer Science and Artificial Intelligence, Universidad de Sevilla, Seville 41012, Spain
[c] Department of Computer Science, Xiamen University, Xiamen 361005, Fujian, People's Republic of China

A R T I C L E   I N F O

Article history:
Received 12 August 2015
Received in revised form 11 November 2015
Accepted 7 December 2015
Available online 17 December 2015
Communicated by T. Yokomori

Keywords:
Natural computing
Virus machine
Register machine
Turing completeness

A B S T R A C T

In recent years, different computing models have emerged within the area of Unconventional Computation, and more specifically within Natural Computing, getting inspiration from mechanisms present in Nature. In this work, we incorporate concepts in virology and theoretical computer science to propose a novel computational model, called *Virus Machine*. Inspired by the manner in which viruses transmit from one host to another, a virus machine is a computational paradigm represented as a heterogeneous network that consists of three subnetworks: virus transmission, instruction transfer, and instruction-channel control networks. Virus machines provide non-deterministic sequential devices. As number computing devices, virus machines are proved to be computationally complete, that is, equivalent in power to Turing machines. Nevertheless, when some limitations are imposed with respect to the number of viruses present in the system, then a characterization for semi-linear sets is obtained.

1. Introduction

The present study can be considered as a contribution to the area of *natural computing*, which is a field of research that investigates both human-designed computing inspired by nature and computing that occurs in nature. That is, this field investigates models and computational techniques based on nature, which enables the development of new computational tools (in software, hardware, or wetware) for problem solving. Such tools can synthesize natural patterns, behaviours and organisms, which may result in designing novel computing systems that use natural media for computation [15,20].

Biological systems are rich sources of ideas for designing computing devices and algorithms. In fact, they have inspired numerous classical computing devices, such as neural nets [11,21,22,24] and evolutionary algorithms [25,26]. In recent years, computing devices inspired by cells (or molecules inside cells, such as DNA) have been thoroughly investigated [1,3,8,19]. Most cell-inspired computing systems have been proved to be universal [4,10,18,23,27] and computationally efficient [6,12,14,16,17].

In virology, a virus is a parasitic biological agent that can only reproduce after infecting a host cell. Every animal, plant, and protist species on this planet has been infected by viruses. The process of viral multiplication (replication) starts for the

* Corresponding author. Tel.: +86 0592 2580033.
E-mail addresses: chenxu31@sina.com (X. Chen), marper@us.es (M.J. Pérez-Jiménez), lvalencia@us.es (L. Valencia-Cabrera), wangbz@xmu.edu.cn (B. Wang), xzeng@xmu.edu.cn (X. Zeng).

http://dx.doi.org/10.1016/j.tcs.2015.12.006
0304-3975/© 2015 Elsevier B.V. All rights reserved.

attachment stage (a virus attaches to the potential host), then the virus must effect entry to be able to replicate (penetration stage) and the replication steps (genome replication, transcription and translation, using ribosomes provided by the host cell) occur next. When the new genomes are produced they come together with the newly synthesized virus proteins to form virus particles (assembly stage). Finally, the particles escape from the cell to infect other cells.

Viruses can transmit from one host cell to another in various ways. For instance, viruses in plants are usually transmitted from a plant to another by insects (e.g., aphids), which feed on plant sap. Viruses in animals can spread by blood-sucking insects (e.g., mosquitoes). Coughing and sneezing may spread influenza viruses. HIV, a well-known virus, can transmit through sexual contact or by exposure to infected blood. Viruses can only reproduce in living cells. After infecting a host cell, viruses use the machinery and metabolism of the host cell to produce multiple copies of themselves, while the original infecting virus is dismantled. For additional details on viruses, refer to [7].

Besides biological viruses, there is a special kind of "virus", called *computer virus*, which is similar to the natural virus. A computer virus is a malicious software that can spread to other computers through networks. Furthermore, it can replicate by inserting copies of itself into other computer programs, data files, and so on. For additional details on computer viruses, refer to [2].

In this study, we introduce a new computing model, called *Virus Machine*, which is inspired from the transmissions and replications of viruses. This system consists of several *hosts* (processing units), connected to each other by *channels*. Each host can be viewed as a group of cells (being part of a colony, organism, system, organ or tissue). *Viruses* are placed in the hosts and are assumed to evolve. Each virus can transmit from one host to another by passing through a channel, and can replicate itself while transmitting. Such transmissions and replications in the system are controlled by several instruction units, which are attached to the channels. A virus machine can be viewed as a heterogeneous network that consists of the following three subnetworks:

- A *virus transmission network*, which is a weighted directed network wherein each node represents a *host* and each arc represents a *transmission channel* through which viruses can transmit from one host to another or to the environment. In addition, a natural number (the *weight* of the channel) is associated with each channel, indicating the number of viruses that will be transmitted to the tail host of the channel after transmission (i.e., a virus may replicate itself while transmitting).
- An *instruction transfer network*, which is a weighted directed network wherein each node represents a *control instruction unit* and each arc represents an optional *instruction transfer path* with a natural number (the *weight* of the instruction transfer) associated to it.
- An *instruction-channel control network*, which is an undirected bipartite network, wherein each node represents either a control instruction or a channel and each edge represents a control relationship between an instruction and a channel.

Each transmission channel in a virus transmission network is closed by default. It can be opened by a control instruction unit, which is connected to the channel through an edge of the instruction-channel control network, when the instruction is activated. When it is open, the channel allows a virus (only one virus) to transmit through it. The virus may replicate itself while transmitting, which indicates that the number of viruses in the head host of the channel is decreased by one, whereas the number of viruses in the tail host of the channel is increased by n (being n the weight of the opened channel). Instructions are activated individually along the paths in the instruction transfer network, so that only one instruction is enabled in each computation step. That is, an instruction activation signal is transferred to the network to activate instructions in sequence.

In this paper we deal with virus machines having input hosts, allowing us to introduce some additional numbers of viruses (encoding the information) in certain distinguished hosts. Then, instructions are allowed to activate one after another and the system halts when no path is available to transfer the instruction activation signal. We consider virus machines working in the *computing mode*: if the virus machine halts, then the computation is finished and the result is the total number of viruses sent to a distinguished region (a host or the environment) during the computation; otherwise (i.e., if the instruction activation signal is transferred continuously), the computation fails to produce an output.

In this work, the computational completeness of virus machines working in the computing mode is established by showing that they simulate register machines. Nevertheless, by considering a more realistic condition concerning with a limit (an upper bound) on the number of viruses present in any host during a computation (*bounded virus machines*), this restriction diminishes in fact the computational power of the systems. In this case, a characterization of semi-linear sets of numbers is obtained.

As far as we know, this paper is a pioneer work in this area, and we believe that further investigations on this subject are worth conducting. In this regard, some new exciting lines of research for future works are presented at the end of this paper.

The paper is organized as follows. First, some preliminaries are briefly introduced in order to make the work self-contained. Then, in Section 3, we formally define the computing model of Virus Machines and the structure of the model is graphically illustrated. In Section 4, an example is given to illustrate how such systems work. Then, we discuss the power of virus machines in Section 5 and the computational completeness (via simulating register machines) is stated. In Section 6 we also investigate the power of bounded virus machines by giving a characterization of semi-linear sets. Finally, in Sec-