BioSystems 94 (2008) 28–33

Contents lists available at ScienceDirect

BioSystems

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

Communication and complexity in a GRN-based multicellular system for graph colouring

Moritz Buck*, Chrystopher L. Nehaniv

Adaptive Systems Research Group, Centre for Computer Science & Informatics Research, University of Hertfordshire, Hatfield, Hertfordshire AL10 9AB, United Kingdom

ARTICLE INFO

Article history:
Received 13 July 2007
Received in revised form 14 November 2007
Accepted 23 May 2008

Keywords:
Genetic Regulatory Networks (GRNs)
Artificial life
Graph colouring problem
Distributed computing
Multicellularity

ABSTRACT

Artificial Genetic Regulatory Networks (GRNs) are interesting control models through their simplicity and versatility. They can be easily implemented, evolved and modified, and their similarity to their biological counterparts makes them interesting for simulations of life-like systems as well.

These aspects suggest they may be perfect control systems for distributed computing in diverse situations, but to be usable for such applications the computational power and evolvability of GRNs need to be studied.

In this research we propose a simple distributed system implementing GRNs to solve the well known NP-complete graph colouring problem. Every node (cell) of the graph to be coloured is controlled by an instance of the same GRN. All the cells communicate directly with their immediate neighbours in the graph so as to set up a good colouring. The quality of this colouring directs the evolution of the GRNs using a genetic algorithm.

We then observe the quality of the colouring for two different graphs according to different communication protocols and the number of different proteins in the cell (a measure for the possible complexity of a GRN). Those two points, being the main scalability issues that any computational paradigm raises, will then be discussed.

1. Introduction

Computer sciences and algorithmics are very powerful nowadays but for certain tasks and computations the standard sequential non-adaptive von Neumann architecture is limited. There is a need of more distributed and adaptive types of models to perform complex computations which are very difficult to design in our actual systems. We need architectures able to adapt to new situations and solve complex problems without the interference or global supervision of a programmer or other human being (Banzhaf, 2004). This might sound almost impossible, yet we are surrounded by these kinds of systems; we ourselves are of this kind. Every living system in nature reacts and adapts to its surroundings and to its needs without any specific overseeing supervision. Hence taking inspiration from nature to design new computational paradigms seems logical. But it is not that simple either: Life took billions of years to evolve something so complex, and still very little is understood about such complex organisations arising in this evolution.

Although gradually we are glimpsing the mechanisms behind the evolution and development of multicellular organisation and its genetic regulatory controls (Buss, 1987; Michod, 2000; Arthur, 2000; Davidson, 2001), every biologist will agree that there remains a tremendous amount to be discovered about the processes inherent to life.

Still, we will propose in this article a small simple model based on a simple network of artificial cells which we will test on a standard computational model: the graph colouring problem. We will here only test the computational capacities of that system, not its adaptive capacities, and will raise some issues which seem of importance in the design of this kind of system, mainly the issues of cell-to-cell communication and upscaling.

For a fixed graph colouring problem, our system will evolve a population of genomes, each encoding a Genetic Regulatory Network (GRN). The genome will control the process of colouring the chosen graph: an instance of the same GRN will control each cell (node) of the network to be coloured via local state and local interactions with neighbouring cells (i.e., with cells at neighbouring nodes in the graph). Artificial GRNs are a relatively new way of representing and using networks in computer sciences; they are deeply inspired by their biological counterparts and are starting to be used for diverse artificial life, artificial intelligence and biological

* Corresponding author.
E-mail addresses: M.Buck@herts.ac.uk (M. Buck), C.L.Nehaniv@herts.ac.uk (C.L. Nehaniv).

0303-2647/$ – see front matter © 2008 Elsevier Ireland Ltd. All rights reserved.
doi:10.1016/j.biosystems.2008.06.002

modelling applications (Banzhaf, 2003; Knabe et al., 2006; Quick et al., 2003; Schilstra and Bolouri, 2002).

The cells (or nodes) of the graph have a very simple way of communicating and should evolve mechanisms to use this information to chose their colour so that every given cell differs in colour from its immediate neighbours.

2. Models

2.1. The Graph Colouring Problem

The graph colouring problem is a very well known combinatorial NP-complete problem. To colour a graph each node of the graph has a colour assigned to it and none of its immediate neighbours is allowed to have the same colour. To decide whether there is a way to colour an arbitrary graph using k colours is NP-complete for $k \geq 3$. It is in fact one of the 21 NP-complete problems described by Karp (1972). This problem is important for numerous real life applications including: map colouring, radio frequency allocation, register allocation in compilers (Mueller, 1993) or scheduling (Marx, 2004). This problem has been approached with numerous different types of algorithms (Biggs, 1990; Costa and Hertz, 1997; Prestwich, 1998; Shawe-Taylor and Zerovnik, 1995), mostly heuristic local optimisation algorithms, with most of these methods being far more effective than our approach to it. But the goal of our work is not to make an especially effective method to solve this problem; rather, we will use the graph colouring as a first test bed for a new cell-based computational model and study the evolvability of different cell-to-cell communication protocols.

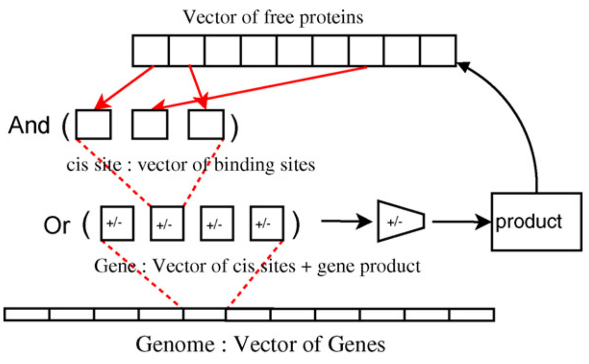
The graph colouring problem is very adapted to our investigations in being simple to understand but still combinatorially complex. Also its inherent local nature is very suited to the structure of a multicellular distributed approach.

2.2. General Setup

The evolutionary approach here is separated in two different time frames: the evolutionary time frame, used for the "training" of the genomes of the multicellular colonies; and a simulation time frame, used for the actual execution and computation by a given (genetically homogeneous) cell colony. In the evolutionary time frame a genetic algorithm (GA) will evolve GRNs which will be evaluated during a simulation run. For this simulation run, a network will be set up, each node of that network will be embodied by a copy of the same GRN (one individual from the genetic algorithm). This simulation of genetic regulatory dynamics on the graph network will run for a certain number of time steps, with cells communicating with neighbours and choosing their colours to achieve the best possible colouring (i.e. no two neighbouring cell having the same colour), and the genome will then be evaluated accordingly. The evolution in the GA is driven by this evaluation to selectively favour fitter genomes.

2.3. Genetic Regulatory Network

The GRNs used for this experiment are Boolean, the same model has been used in (Buck and Nehaniv, 2006a). The structure of a single genome is shown in Fig. 1 Inside a cell their are n different proteins, the level of each protein is modeled by a Boolean value reflecting its presence (*true*) or absence (*false*). The network structure is derived from the genome consisting of a sting of genes, with each gene composed of a regulatory part and a part specifying its protein product. The regulatory part represents the inbound connections of the network whereas the product part represents the outbound. The inbound part is structured in so-called cis-sites,



Vector of free proteins
And
cis site : vector of binding sites
Or
+/-
product
Gene : Vector of cis sites + gene product
Genome : Vector of Genes

Fig. 1. Schematic of the Boolean genetic regulatory network model.

which themselves each consist of a number of binding sites. A binding site returns a Boolean value depending on the presence in the cell of the protein it is supposed to bind.

The values returned by all the binding sites of a cis-site are joined by an *AND* operator. The obtained value is then negated if the cis-site is an *inhibitory* one. Then all the values returned by the cis-sites of a gene are joined by an *OR* operator. This value is then finally negated if the gene is *default on*, if the final value of this operation is *true* then the protein encoded by the gene will be produced, i.e. the value indicating the presence of this protein in the cell will be set to *true*. If more than one gene can produce the same protein, to set the value for that protein to true for the cell, any one of them suffices. The system has a one time step memory; at every simulation time step it takes the protein state vector of the cell in the previous step and creates a new protein state vector using the genetic regulatory network.

Formally, for each gene of a cell's genome, we have for a protein binding site i the binding value b_i,

$$b_i = \begin{cases} \mathit{true} & \text{if binding protein is present} \\ \mathit{false} & \text{binding protein is not present} \end{cases}$$

The expression value c_j of a cis-site j,

$$c_j = \begin{cases} \bigwedge_{\text{all } i} b_i & \text{if j is activatory} \\ \neg\bigwedge_{\text{all } i} b_i & \text{if j is inhibitory} \end{cases}$$

where the logical AND-operation is taken over all binding sites b_i of the given cis-site c_j. And the final protein production p_k of the gene k,

$$p_k = \begin{cases} \bigvee_{\text{all } j} c_j & \text{if k is default off} \\ \neg\bigvee_{\text{all } j} c_j & \text{if k is default on} \end{cases}$$

where the logical OR-operation is taken over all cis-sites c_j of gene k. The new value of p_k will be *true* if and only if at least one gene produces p_k. It can be shown that this system is complete in the sense of combinatorial logic: given a Boolean vector of size n (the vector of the n proteins of the cell) there always exists at least one network computing every one of the $(2^n)^{(2^n)}$ possible Boolean functions. (This can be easily seen by writing the logical function to determine the presence or absence of each protein in conjunctive normal form as function of the activation levels of all proteins in the cell, and translating this form into a genome with n default-on genes.) This model has also some other interesting characteristics. It is quite robust to mutation, at least in principle, for example if you duplicate one gene the function represented by the network is not altered, which is not the case for most continuous GRN models (Buck and Nehaniv, 2006b; Knabe et al., 2006).