J Math Model Algor (2009) 8:203–225
DOI 10.1007/s10852-009-9103-7

Metabolic Systemic Computing: Exploiting Innate Immunity within an Artificial Organism for On-line Self-Organisation and Anomaly Detection

Erwan Le Martelot · Peter J. Bentley

Received: 17 October 2008 / Accepted: 19 December 2008 / Published online: 14 March 2009

Abstract Previous work suggests that innate immunity and representations of tissue can be useful when combined with artificial immune systems. Here we provide a new implementation of tissue for artificial immune systems using systemic computation, a new model of computation and corresponding computer architecture based on a systemics world-view and supplemented by the incorporation of natural characteristics. We show using systemic computation how to create an artificial organism, a program with metabolism that eats data, expels waste, self-organise cells based on the nature of its food and emits danger signals suitable for an artificial immune system. The implementation is tested by application to two standard machine learning sets and shows excellent abilities to recognise anomalies in its diet as well as a consistent datawise self-organisation.

Keywords Systemic computation · Tissue · Innate immunity · Anomaly detection · Self-organisation · Danger theory · Artificial organism · Artificial metabolism · Artificial immune system

1 Introduction

An increasingly popular view in the field of Artificial Immune Systems (AIS) holds that innate immunity (as enabled by non-adaptive cells such as dendritic cells) can play a significant role in maintaining immunity in computer systems [1]. Notions

E. Le Martelot (✉)
Engineering Department, University College London, London, UK
e-mail: e.le_martelot@ucl.ac.uk

P. J. Bentley
Computer Science Department, University College London, London, UK
e-mail: p.bentley@cs.ucl.ac.uk

such as the Danger Theory suggest that normal self cells may provide signals when damaged, thus helping to encourage the response of immune cells in the right areas of the tissue of an organism at the right time [2]. Previous work has investigated the development of an artificial tissue to serve this function, providing an interface between data and AIS, and performing preliminary data processing and clustering [3].

In this work we extend the previous work on tissue for AIS, and investigate a different implementation based on the recent paradigm and computer architecture, systemic computation (SC) [4] designed to support any bio-inspired system by enabling natural characteristics found in biology. In contrast to previous implementations of tissue, which largely ignore the relationships between real organisms and their environments, here we present a model of organism, implemented as a systemic computation program with its own metabolism that eats data, expels waste, self-organise its cells depending on the nature of its food and can emit danger signals for an AIS. The implementation is tested by application to two standard machine learning sets. Tested against the Breast Cancer dataset [5], the organism shows excellent abilities to recognise anomalies in its diet. Then, we investigate its self-organisation capabilities using first an ideal dataset to illustrate how the organism is ideally behaving, and then using the UCI Wine [6] dataset to investigate further the organism's growth and self-organisation in a real case study. A visualisation of programs' organisation, revealing their inner structure, is also presented to illustrate the on-line self-organisation.

2 Background

Although not commonly modelled, the notion of tissue is fundamental to immunity. The immune system within an organism defends the tissue of that organism. The concept of artificial tissue has been used for instance in the POEtic project, aiming at creating a hardware platform organised with a similar hierarchy as found in biological systems [7], and using reconfigurable circuits to simulate tissue growth [8]. It has also been used in work that implemented an AIS in a sensor network, the sensor nodes taking on the role of tissue cells [9].

In biology, tissue is a crucial part of the immune system and its importance was particularly highlighted by Polly Matzinger when introducing the Danger Model [2]. This view rejected the notion that the immune system differentiates self from non-self and suggested that it instead responds to cellular damage. It thus suggests that cells that die abnormally release signals which encourage immune cells to converge on that location and become more active.

This theory was adopted in [3] to propose two ways of growing tissues where damaged cells would release danger signals exploitable by an AIS. Tissue was defined as the interface between a problem to solve and the AIS. Here we follow a similar view, but attempt to improve the tissue model and its potential advantages by implementing a tissue-growing program designed for AIS using systemic computation - a parallel computer architecture designed to support natural computation.

Systemic computation is not the only model of computation to emerge from studies of biology. The potential of biology had been discussed in the late 1940s by Von Neumann who dedicated some of his final work to automata and self-replicating machines [10]. Cellular automata have proven themselves to be a valuable approach

to emergent, distributed computation [11]. Generalisations such as constrained generating procedures and collision-based computing provide new ways to design and analyse emergent computational phenomena [12, 13]. Bio-inspired grammars and algorithms introduced notions of homeostasis (for example in artificial immune systems), fault-tolerance (as seen in embryonic hardware) and parallel stochastic learning, (for example in swarm intelligence and genetic algorithms) [4].

New architectures are also popular, whether distributed computing (or multi-processing), computer clustering and grid computing and even ubiquitous computing and speckled computing [14]. Thus, computation is increasingly becoming more parallel, decentralised and distributed. However, while hugely complex computational systems will be soon feasible, their organisation and management is still the subject of research. Ubiquitous computing may enable computation anywhere, and bio-inspired models may enable improved capabilities such as reliability and fault-tolerance, but there has been no coherent architecture that combines both technologies. Indeed, these technologies appear incompatible—the computational overhead of most bio-inspired methods is prohibitive for the limited capabilities of ubiquitous devices.

To unify notions of biological computation and electronic computation, [4] introduced systemic computation as a suggestion of necessary features for a computer architecture compatible with current processors, yet designed to provide native support for common characteristics of biological processes.

In this paper we use an approach similar to [3] and deepen the biological analogy by modelling an artificial organism as a program with metabolism. The program does not only mimic some tissue features but also mimics many fundamental properties of living organisms: eating data as food and expelling waste, while growing tissue, and releasing danger signal when its cells die in an abnormal way.

To implement such program SC provides a suitable alternative approach to traditional computation. Indeed with SC, organisms and software programs now share a common definition of computation. The work illustrates how organisms and programs can behave similarly, sharing the notion of metabolism, using SC.

3 Systemic Computation

Looking at a biological brain, an ant colony, an immune system, the growth of a plant or a crystal, nature is clearly performing some kind of computation. We can state that natural computation is stochastic, asynchronous, parallel, homeostatic, continuous, robust, fault tolerant, autonomous, open-ended, distributed, approximate, embodied, has circular causality, and is complex [4]. The traditional von Neumann architecture however is deterministic, synchronous, serial, heterostatic, batch, brittle, fault intolerant, human-reliant, limited, centralised, precise, isolated, uses linear causality and is simple. The incompatibilities seem clear.

To address these issues, [4] introduced Systemic Computation (SC), a new model of computation and corresponding computer architecture based on a systemics world-view and supplemented by the incorporation of natural characteristics (listed above). This approach stresses the importance of structure and interaction, supplementing traditional reductionist analysis with the recognition that circular causality,

embodiment in environments and emergence of hierarchical organisations all play vital roles in natural systems. Systemic computation makes the following assertions:

- Everything is a system.
- Systems can be transformed but never destroyed or created from nothing.
- Systems may comprise or share other nested systems.
- Systems interact, and interaction between systems may cause transformation of those systems, where the nature of that transformation is determined by a contextual system.
- All systems can potentially act as context and affect the interactions of other systems, and all systems can potentially interact in some context.
- The transformation of systems is constrained by the scope of systems, and systems may have partial membership within the scope of a system.
- Computation is transformation.

Computation has always meant transformation in the past, whether it is the transformation of position of beads on an abacus, or of electrons in a CPU. But this simple definition also allows us to call the sorting of pebbles on a beach, or the transcription of protein, or the growth of dendrites in the brain, valid forms of computation. Such a definition is important, for it provides a common language for biology and computer science, enabling both to be understood in terms of computation.

In systemic computation, everything is a system, and computations arise from interactions between systems. Two systems can interact in the context of a third system. All systems can potentially act as contexts to determine the effect of interacting systems. One convenient way to represent and define a system is as a binary string. Each string is divided into three parts: two schemata and one kernel. These three parts can be used to hold anything (data, typing, etc.) in binary as shown in Fig. 1.

The primary purpose of the kernel is to define an interaction result (and also optionally to hold data). The two schemata define which subject systems may interact in this context as shown in Fig. 2.

A system can also contain or be contained by other systems. This enables the notion of scope. Interactions can only occur between systems within the same scope. An SC program therefore comprises systems that are instantiated and positioned within a hierarchy (some inside each other). It thus defines an initial state from which the systems can then randomly interact, transforming each other through those interactions and following an emergent process rather than a deterministic algorithm. For full details see [4] and [15].

Systemic Computation has been used to model genetic algorithms, neural networks and has demonstrated properties of flexibility, fault tolerance, and self-repair [15–17].

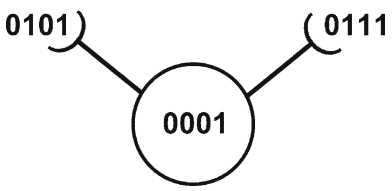
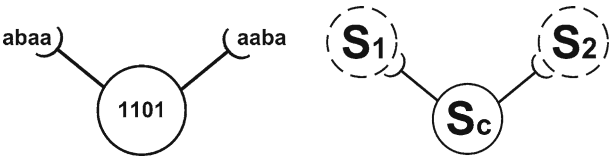


Fig. 1 A system used primarily for data storage. The kernel (in the *circle*) and the two schemata (at the end of the *two arms*) hold data



(a) Content system (b) Interaction

Fig. 2 **a** A system acting as a context. Its kernel defines the result of the interaction while its schemata define allowable interacting systems. **b** An interacting context. The contextual system S_c matches two appropriate systems S_1 and S_2 with its schemata and specifies the transformation resulting from their interaction as defined in its kernel

4 An SC Program with Metabolism

4.1 Systemic Analysis

Systemic computation is an alternative model of computation. Before any new program can be written, it is necessary to perform a systemic analysis [15] in order to identify and interpret appropriate systems and their organisation. The systemic analysis is for a given problem how to think in SC. When performed carefully, such analysis can itself be revealing about the nature of the problem being tackled and corresponding solution. A systemic analysis is thus the expression of any natural or artificial process in the language of systemic computation. The steps are in order:

- Identify the low-level systems (i.e. determine the level of abstraction to be used), Analysis of interactions (which system interacts with which other system in which context system),
- Determine the order and structure (scopes) of the emergent program (which systems are inside which other systems) and the values stored within systems.

The systemic analysis is complete when all systems and their organisation have been designed for the given problem (or biological process or organism), providing an easy to understand graph-based model.

The first stage is to identify the low-level systems. In most artificial immune systems, the level of abstraction is the cell: few approaches require modelling of the internal organelles or genome of cells, and few require modelling of populations of organisms. Here we intend to model the growth of tissue cells, the consumption of "food" (data items), the expulsion of waste and the emission of danger signals. Thus an abstraction at the cellular level is appropriate, with systems being used to explicitly model each element.

The identification of appropriate low-level systems is aided by an analysis of interactions. The organism should be able to eat food from its environment, use this food to grow organs (clusters of cells) by creating new cells and expel waste into the environment.

To prevent being overloaded with systems, the waste can be recycled into new food (a simple ecosystem model). Food and waste could therefore be seen as different states of the same system (in SC systems can be transformed, but never

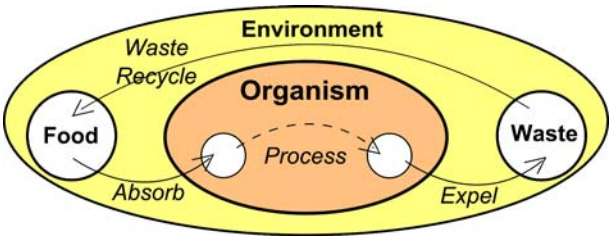


Fig. 3 'Food to waste' cycle for an organism within its environment: food is absorbed by the organism, processed as energy to grow tissues before being expelled when the organism cannot make use of it any more

created from nothing or destroyed). Also, the food is what the organism takes from its environment to be able to grow. Therefore cells and all the necessary matter for the growth should also derive from the food systems.

We can thus visualise the ecosystem between the organism and the environment as shown in Fig. 3.

Looking within the organism, it takes food as input and this food must be sufficient to grow tissue. One simple way to model this is by using the approximation that the food is transformed into cells when absorbed by the organism. However, to enable cells to adhere to each other (rather than float free), cells need some sticky adhesion molecules. Here we do not need to explicitly model all these molecules but an "adhesion surface" is at least required to bind two or more cells together. As SC forbids the creation of systems from nothing, the adhesion surfaces must be obtained either from incoming food or from the cells themselves. In a biological organism each cell has a limited lifespan and thus dies at some point. It may then be consumed by macrophages or dendritic cells and its energy is partially recycled. In the model dead cells can thus be recycled to make adhesion surfaces. A growth process can now attach cells to each other by using adhesion surfaces to create tissue. To regulate this growth and introduce the notion of time, a decay process simulates the aging of cells. When cells die, a split process splits them from the adhesion surfaces they are bound to.

So the organism eats new data, converts each data item into a new cell, and attempts to bind that cell to itself, with cells made from similar data items binding to each other. Thus, a cell unable to bind to any group of cells reveals itself to be significantly different from them—more like the result of an invading pathogen then part of the organism. If this abnormal cell dies unbound, it can therefore be spotted as a potential anomaly. In that case, the death of the cell can entail that cell releasing a Danger signal (i.e. the cell can be converted into a signal). This signal can then be used by an AIS algorithm which can be implemented through the addition of systems corresponding to immune cells. (Here we focus on the organism.)

The organism can also make use of a hunger parameter defining a maximum amount of alive cells it can contain at a time. This parameter can be stored in the organism system and the absorption context then only allows food absorption if the organism is "hungry". This parameter can be useful to avoid having the organism growing too big and using too much memory/data at a time. A bad usage of memory could indeed to some extend slow down the computation process significantly.

The organism food to waste chain is therefore as shown in Fig. 4.

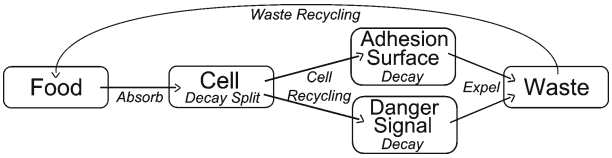


Fig. 4 'Food to waste' cycle within the organism: food is absorbed, transformed into cells. When dying cells can be recycled into adhesion surfaces if they were part of a tissue or turned into a danger signal if they were single. Cells, adhesion surfaces and danger signals have a limited lifespan and decay over time (i.e. when they reach a certain age they die). When dying, cells also need a split process to detach them from the tissue they were part of

From this defined cycle, the interactions and systems in the model can be written in a SC calculus form as follows (also see Fig. 5):

organism }-absorb-{ food	→	organism(cell)
cell }-growth-{ adhesion_surface	→	cell(adhesion_surface)
cell(adhesion_surface) }}-split	→	(cell adhesion_surface)
organism(cell) }}-cell_recycling	→	organism(adhesion_surface or danger_signal)
X[age](time) }}-decay	→	X[age+1](time), X = cell or adhesion_surface or danger_signal
organism(X) }}-expel	→	(organism waste), X = adhesion_surface or danger_signal
universe(waste) }}-waste_recycling	→	universe(food[data])

The absorb system models endocytosis (e.g. via cell receptors), the growth system models the organism's genome, the decay models the aging (progression along the axis of time), the split system models a chemical breakdown between adhesion molecules and cell wall, the cell recycling models the phagocytes, the expel system models exocytosis, waste recycling systems model the ecosystem, the universe models the environment, the organism system models the boundary between tissue and environment, food systems model nutrients, cells model tissue cells, adhesion surfaces model adhesion molecules, danger signal systems model Matzinger's danger signals, waste systems model cell waste (unused or unusable compounds), and the time system models the dimension of time.

Figure 5 summarises the organism's organisation and shows the potential interactions. Food is absorbed and turned into cells. Cells can bind to others via adhesion surfaces depending on the data they carry. Each cell contains the dimension of time and a decay process. The decay process increases an age counter held in each cell. When the age reaches the cell's lifespan, the cell is marked as dead. The split process can then detach the dead cell from the adhesion surfaces it was bound to. If the cell was previously bound to other cells (via adhesion surfaces) the cell is recycled (by the cell recycling system) into an adhesion surface whereas if the cell remained single it turns into a danger signal. An adhesion surface can last a minimum of the cell's lifetime but also as long as it is bound to at least one cell (to avoid breaking clusters for non data related reason). Danger signals decay like cells. Once a danger signal or an adhesion surface is dead it can be expelled out of the organism and turned into

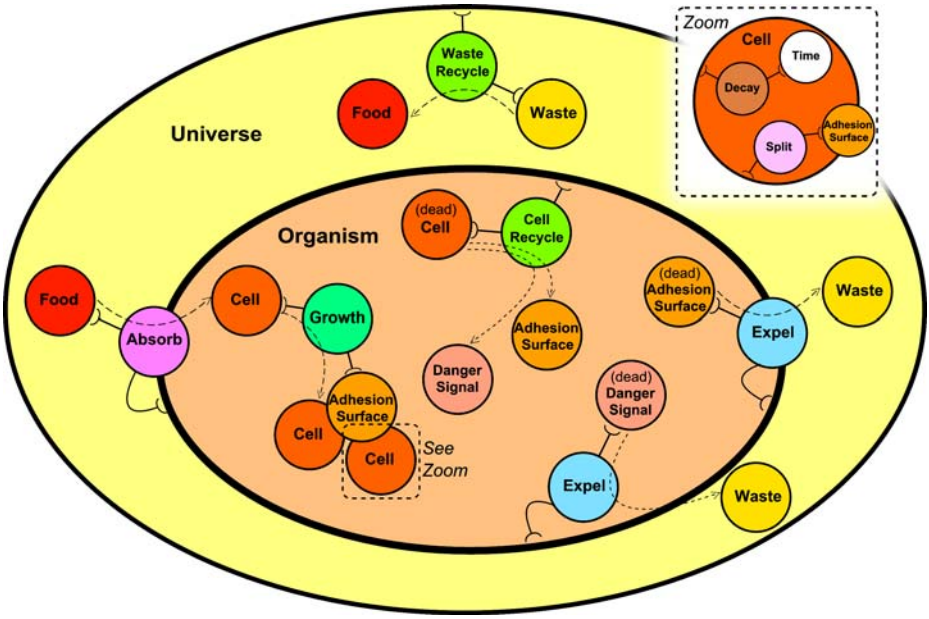


Fig. 5 Systemic organisation of the organism. The universe contains a waste recycling system, some waste and food, and an organism. The organism shares with the universe the absorption context. It contains cells, adhesion surfaces, danger signals, growth contexts, cells recycling contexts and expelling contexts. Finally cells (and thus all derived system states like adhesion surfaces and danger signals) contain the time system, a decay process and a split process. The schemata appear on context systems to show the allowable interactions between systems. The *dashed arrows* indicate the potential transformation of some systems during an interaction. For instance on the far left we can observe a food system interacting with an organism in an absorption context: the food is turned into a cell and injected into the organism

waste by the expel system. Waste can finally be recycled by the ecosystem (waste recycling system) into new food.

Note that waste recycling, absorption, cell recycling and expel systems should have the same amount of instances. Indeed, on average if one food system can be created at a time, then only one can be absorbed, then recycled and finally expelled at a time.

4.2 Data Organisation within the Artificial Organism

So far an organism has been modelled within SC. To use this organism to organise the data, the data processing method has to be defined.

To incorporate data into the organism's metabolism, new data items are placed into food systems, where it is stored in the schemata. Data from an incoming stream can be introduced when recycling waste (i.e. new data are pushed into the resulting food systems). The amount of waste recycling and absorption systems gives the data introduction rate (the more food can be absorbed at a time, the more data are introduced). The data are then absorbed into the organism and transformed to cells. When a growth interaction occurs between a cell and an adhesion surface, the two are bound based on their data similarity. Algorithm 1 describes in pseudo-code the

binding method. For binding a cell to an adhesion surface the adhesion surface is injected into the cell but remains also part of the organism so that more cells can bind to it.

Algorithm 1 Pseudo-code for the growth context binding method. τ is a given threshold. The distance function calculates the Euclidian distance of two vectors.

```
if adhesion surface not bound to anything then
    Bind cell and surface
    Surface data value ← Cell data value
else if distance(Cell data, Surface data) ≤ τ then
    Bind cell to surface
    Surface data value ← Average(Surface data, Cell data)
end if
```

The measure chosen in this implementation to compare data is the Euclidian distance (as was used in [3]). In the organism, cells bind together according to their values, and various clusters may emerge from this, thus reflecting the data distribution. If a cell is left single then it means it cannot bind anywhere and therefore holds data significantly different from the current most common data values contained within the organism. This cell is then turned into a danger signal holding the data that an AIS could use to develop antibodies.

5 Parameters Setup and Behavioural Analysis

5.1 Mathematical Modelling

Depending on how many instances of each system are provided in the organism, its behaviour can vary. For a given absorption rate, a longer cell's lifespan would keep cells longer in the organism, therefore making it bigger. More growth systems would give a higher probability for a given cell and a given surface to interact. To understand better the implications of each parameter, the organism's behaviour can be analysed in terms of probability. A good measure of the behavioural accuracy of the organism is for a given cell holding a datum from class i the probability p_{c_i} to be picked and then bound to other cells from class i via any surface it is elligible to bind to.

The parameters in our organism are:

- n_g amount of growth systems,
- c_l cells' lifespan,
- d_r data introduction rate.

The aim is therefore to express the probability p_{c_i} with these parameters.

The probability for respectively a cell and an adhesion surface to be picked by a growth context are:

$$p_{pick(c)} = \frac{1}{n_c} \qquad \text{and} \qquad p_{pick(s)} = \frac{1}{n_s}$$

with n_c the estimated amount of cells in the organism and n_s the estimated amount of adhesion surfaces.

The estimated amount of cells n_c depends on the cells' lifespan and the data introduction rate:

$$n_c = c_l \,.\, d_r$$

Cells being recycled into adhesion surfaces, the amount of adhesion surfaces n_s is the amount of cells n_c less the amount of danger signals n_d:

$$n_s = n_c - n_d$$

Note that:

$$n_s \leq n_c \qquad \text{and} \qquad n_d \leq n_c$$

The expected amount of cells in the organism for a given class i of the dataset $E(c_i)$ is given by:

$$E(n_{c_i}) = p_{r_i} \,.\, n_c = p_{r_i} \,.\, c_l \,.\, d_r$$

where p_{r_i} is the introduction probability of any sample of class i and $\sum_{k=1}^{n_{cl}} p_{r_k} = 1$. n_{cl} is the number of classes of data in the dataset.
If classes samples are equally introduced, then

$$p_{r_i} = \frac{1}{n_{cl}} \tag{1}$$

This will be referred to as event Eq.

Single adhesion surfaces binding to single cells disregarding their data content, if we only consider new cells and new surfaces, the expected amount of adhesion surfaces linked to cells of a given class i is $E(n_{s_i})$ given by:

$$E_{new}(n_{s_i}) = n_s . p_{r_i} \tag{2}$$

However, throughout a run $E(n_{s_i})$ can vary as the organism can also bind cells in an unwanted way (i.e. cells from different classes bound together). In such case a surface initially bound to one cell of class i could then also be bound to more cells of another class j, making this surface mainly linked to class j data. Therefore Eq. 2 become Eq. 3 throughout execution:

$$E(n_{s_i}) = n_s . p_{r_i} . (1 - p_{lose(i)}) + \gamma_i \tag{3}$$

where $p_{lose(i)}$ is the probability that a surface initially linked to a cell of class i would then bind together more cells of another class, and γ_i is the amount of surfaces initially bound to another class that became bound to more cells of class i.

To study the behaviour of the organism we consider the ideal case where the organism binds cells accurately so that $p_{lose(i)} = 0$ and $\gamma_i = 0$ along the execution. This ideal event will be called Id. Simplifications of further equations using this assumption are thus given using the conditional probability notation.

For a given class i we can now express the probability to pick a surface bound to cells of class i as

$$p_{s_i|i} = \frac{E(n_{s_i})}{n_s}$$

and

$$p_{s_i|i,Id} = p_{r_i}$$

therefore for a given cell of class i the probability to be picked and tested against a surface bound to cells of class i is

$$p_{s_i|c_i} = \frac{E(n_{s_i})}{n_c.n_s}$$

and

$$p_{s_i|c_i,Id} = \frac{p_{r_i}}{n_c}$$

The probability to be picked and tested during a cycle is then

$$p_{cycle\ s_i|c_i} = 1 - \left(1 - \frac{E(n_{s_i})}{n_c.n_s}\right)^{n_g}$$

and

$$p_{cycle\ s_i|c_i,Id} = 1 - \left(1 - \frac{p_{r_i}}{n_c}\right)^{n_g}$$

Finally the probability p_{c_i} is therefore:

$$p_{c_i} = 1 - \left(1 - \frac{E(n_{s_i})}{n_c.n_s}\right)^{n_g.c_l} = 1 - \left(1 - \frac{1}{c_l.d_r}.\frac{E(n_{s_i})}{n_s}\right)^{n_g.c_l} \tag{4}$$

Using our ideal behaviour assumption, this probability can be written as:

$$p_{c_i|Id} = 1 - \left(1 - \frac{p_{r_i}}{c_l.d_r}\right)^{n_g.c_l} \tag{5}$$

If class samples are introduced with equal probability, then using Eqs. 1 and 5 becomes:

$$p_{c_i|Eq,Id} = 1 - \left(1 - \frac{1}{n_{cl}.c_l.d_r}\right)^{n_g.c_l} \tag{6}$$

These results are based on expected values of cells ($E(n_{c_i})$) or surfaces ($E(n_{s_i})$) but ignore for instance the probability of having a lower amount of cells from a class than the expected value. For a given class i the probability of having no cell from this class at a given time in the organism is

$$p_{no\_cell(i)} = (1 - p_{r_i})^{n_c} = (1 - p_{r_i})^{c_l.d_r} \tag{7}$$

Also in our organism, cells bind according to data similarities. So far we considered data of a same class to be straightforward to identify but this identification is part of the process in the organism and its hardness depends on the dataset. So, calling $p_{can\_bind(c_i)}$ the probability for a cell of class i to have a cell (and a surface) from class i to bind to, we can state that:

$$p_{can\_bind(c_i)} = (1 - p_{no\_cell(i)}).\delta_i \tag{8}$$

where δ_i is the dataset compactness factor for class i taking values within the range [0, 1] (i.e. towards one, dataset is compact, towards zero it is sparse).

5.2 Parameters Impact Analysis

n_g Equation 4 shows that increasing the amount of growth systems increases p_{c_i}. The downside of this parameter is from a memory point of view, the higher n_g, the more systems required for the growth task.

d_r Equation 5 shows that increasing the data introduction rate decreases p_{c_i}. However, the decreasing rate (on the denominator) in $\frac{1}{x}$ is not exponential like n_g in $\left(\frac{1}{a}\right)^x$. Therefore, a bigger amount of growth systems could be used to maintain a same p_{c_i} at a higher introduction rate, but this would only be relevant if the rate was significantly increased. Finally, Eqs. 7 and 8 also show that a higher data rate increases the probability for a cell to bind to another cell of its class.

c_l Studying the slope of Eq. 5, we find that it continuously goes down. Investigating the limit towards infinity of c_l we find:

$$\lim_{c_l \to \infty} p_{c_i|I} = \lim_{c_l \to \infty} 1 - \left(1 - \frac{p_{r_i}}{c_l.d_r}\right)^{n_g.c_l} = 1 - e^{-\frac{n_g.p_{r_i}}{d_r}} \tag{9}$$

Therefore as c_l increases p_{c_i} goes down to the value $1 - e^{-\frac{n_g.p_{r_i}}{d_r}}$. However, from Eqs. 7 and 8 it is clear that the higher c_l, the higher $p_{can\_bind(c_i)}$, therefore the chosen value for c_l cannot be taken as low as Eq. 5 only would suggests. A trade-off has to be made.

Number of classes When looking at Eq. 6, we can see that n_{cl} has an impact on p_{c_i}. The greater n_{cl}, the lower p_{c_i}. n_{cl} is not a parameter, it is a property of the dataset. Therefore n_{cl} is a value indicating the difficulty of the classification problem for the organism. Clearly, for a given organism, the more classes, the harder to classify the data.

6 Experiments and Results

To test the model and validate its mathematical analysis, a series of experiments were performed using two standard UCI machine learning datasets and one ideal dataset. First, anomaly detection was investigated using the UCI "breast cancer" dataset [5], also comparing the results with previous work [5] and looking at the state of the program over time. Then self-organisation within the organism is investigated using first an ideal dataset to provide an illustration of an ideal behaviour, and then using the UCI "Wine" [6] dataset.

6.1 Anomaly Detection in Breast-Cancer Dataset

This set of experiments uses the "breast cancer" dataset [5], comprising 458 benign items (class 1) and 241 malignant items (class 2), each item being a vector of 9 real-valued numbers. The choice of this dataset is ideal to test the clustering of benign data together while emitting danger signals for malign data. Choosing this dataset also enables comparison with similar previous models [3].

6.1.1 Tuning the System

To tune the organism for this dataset several settings were employed. Each experiment was repeated 20 times. Each run consisted of 3000 iterations with randomly picked data presented each iteration. Class 1 is treated as the "normal" class of data and class 2 is treated as "abnormal", from which one data item is introduced on average every 25 iterations (these values are taken from [3] to enable comparison), i.e. with a probability of 1/26.

All experiment settings involve a universe, an organism, a time system, an equal amount of waste recycling, absorption, cell recycling, expelling system varying in the experiments (see data introduction rate in Table 1), 250 data systems in experiments 1 to 12 and respectively 500, 750, 1000, 1250 and 1750 in experiments 13, 14, 15, 16 and 17. Each data system contains a decay and a split system. The organism initially contains 5 adhesion surfaces.

Table 1 shows the results of various tunings that were used. These results are computed by discarding the early computations (we discarded here the first 50000 computations) during which the organism grows to an adult (stable) state and stopping the experiments when the flow of data ends (thus not permitting organism's death from starvation).

From the first four experiments we can observe that the data similarity threshold (τ) has a significant impact on the performance of the program. The bigger the threshold, the less benign items are left unbound but the more malign cells can potentially bind somewhere. A threshold of 0.4 was then used for the next experiments.

Table 1 This table shows in percentage the average and standard deviation of data from each class from the breast cancer dataset creating a danger signal (false positive for class 1 and true positive for class 2) for various setups

Exp	d_r	τ	c_l	n_g	Class 1		Class 2	
					mean	stddev	mean	stddev
1	1	0.2	15	100	22.70	1.27	99.87	0.39
2	1	0.3	15	100	11.54	0.76	99.63	0.69
3	1	0.4	15	100	7.56	0.51	98.72	1.01
4	1	0.5	15	100	5.39	0.67	96.19	1.69
5	1	0.4	5	100	9.16	1.11	99.20	1.17
6	1	0.4	10	100	7.85	0.48	99.06	1.14
7	1	0.4	20	100	7.23	0.41	98.73	1.06
8	1	0.4	15	5	14.40	0.81	99.91	0.42
9	1	0.4	15	10	8.98	0.57	99.61	0.62
10	1	0.4	15	25	7.85	0.59	99.53	0.87
11	1	0.4	15	50	7.62	0.40	98.95	0.90
12	1	0.4	15	150	7.62	0.50	98.44	1.00
13	2	0.4	15	100	6.99	0.59	98.56	1.36
14	4	0.4	15	100	7.01	0.68	97.88	1.34
15	8	0.4	15	100	7.48	0.94	98.88	1.24
16	16	0.4	15	100	6.85	0.81	98.33	2.39
17	24	0.4	15	100	7.24	0.90	98.29	2.21

Parameters are respectively in order: new data introduction rate per cycle (d_r), data comparison threshold (τ), cell's lifespan (c_l), and amount of growth systems (n_g). Experiments 1–4 investigate the effect of varying τ, experiments 5, 6, 3, 7 investigate the effects of varying lifespan, experiments 8–11, 3, 12 investigate changing the amount of growth systems, experiments 3, 13-17 investigate varying the data introduction rate

Table 2 This table shows in percentage the average of data from each class creating a danger signal (false positive for class 1 and true positive for class 2) for the two algorithms presented in [3] and using various setups (see [3] for details about the experiments setup)

Exp	Algorithm 1		Algorithm 2	
	Class 1	Class 2	Class 1	Class 2
1	10.5	98.1	20.6	99.99
2	10.5	95.6	21.2	100
3	10.5	92.3	22.4	100
4	14.4	99.9	42.3	100
5	8.9	95.7	15	99.82
6	22.6	98.3	–	–
7	8.3	98.4	–	–
8	10.9	99.6	67.1	100
9	9.9	97.2	14.9	99.98

Experiments 5, 6, 3 and 7 show that increasing the lifespan (c_l) lowers the anomaly detections of class 1 data whilst slightly increasing the one of class 2. Anomaly detection (false positive) for class 1 decreases less and less as c_l increases , as described in Eq. 9. It also shows, as highlighted by Eqs. 7 and 8, that a too short lifespan (e.g. $c_l = 5$) gives significantly less accurate results than a higher value (e.g. $c_l = 10$). For malign data (class 2), staying longer in the organism can only increase the chances of eventually binding somewhere by mistake. The lifespan should thus be long enough for class 1 samples to bind correctly without being overly long. Here experiments show that beyond a lifespan of 10 there is no significant increase in performance.

Experiments 8 to 11, 3 and 12 show that increasing the amount of growth systems decreases the amount of false positives for class 1 (in this case significantly up to $n_g = 50$ and then smoothly) and slightly lesser the anomaly detection success for class 2. As shown in Eq. 4, more growth systems can only give better chances for class 1 samples to bind. The drawbacks are again to also offer more opportunities to bind somewhere some class 2 samples.

Experiments 3 and 13 to 17 show that in this configuration varying the data introduction rate only does not have a significant impact on the organism's performance. Indeed for class 1, $p_{r_i} = \frac{25}{26}$, significantly reducing $p_{no\_cell(i)}$ from Eq. 7 even with a low data rate, making the rate impact less significant.

Comparing these results with the ones from [3] given in Table 2 and looking at the best setups, we clearly outperform their results. It is interesting to notice that the best setups here use a threshold of 0.4 against 0.2 in [3]. It seems that for this study, having a low threshold in a deterministic program as in [3] is better whilst in a stochastic approach such as SC a larger threshold works better.

6.1.2 Looking into the Organism

This section investigates what can be learnt from the organism's inner state over time. It is expected that observing the organism from within should reveal information about the dataset. To be an effective and useful tissue algorithm, suitable for use with AIS algorithms, the organism should organise itself in a stable pattern reflecting the data distribution.

The relevant values to observe along the computation are the amount of cells, clusters of cells and danger signals. Similarly to a real case on-line program execution, Fig. 6 shows the state over time of an organism during a run of 5000 samples using

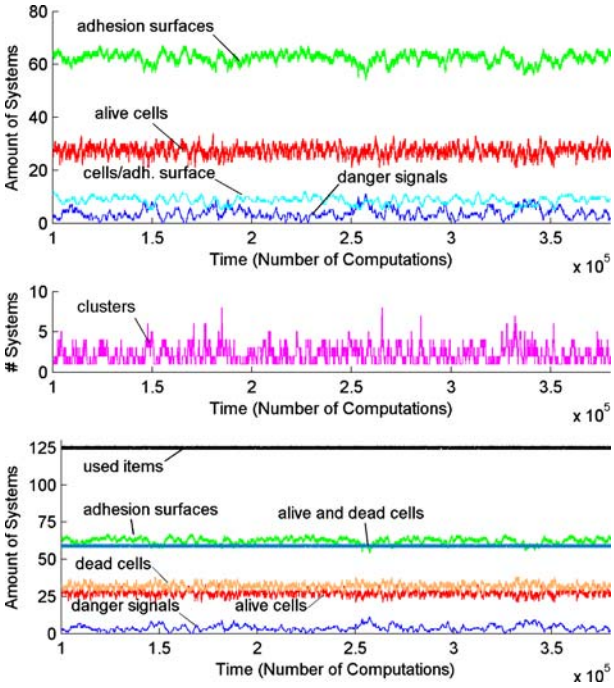


Fig. 6 Organism's inner organisation over a run of 5000 samples

the configuration of experiment 3. Again, the computations corresponding to the organism's early life (here the first 100000 iterations) are discarded in order to focus on the mature aspect of the organism.

Results from Table 1 already provide insights regarding the inner shape of the organism (i.e. its inner organisation). Class 2 (malign) items are very well identified which means that class 2 data are not easily aggregated to other data (otherwise they could not be well detected). Therefore class 1 (benign) data only is actually binding together and it is thus expected to observe on average one main cluster all along the program execution.

Figure 6 shows that the organism has a constant amount of cells in spite of constant cellular death. The organism therefore shows homeostatic behaviour. Danger signals are regularly emitted and represent the detected supposedly malign cells that could then be used by an AIS algorithm. The curve at the bottom shows the amount of clusters over time (in a smaller scale along the Y-axis for clarity). We can observe that as expected the organism keeps settling down into one cluster. New clusters are constantly created with the appearance of new adhesion surfaces but quickly these new clusters bind to the main one.

6.2 Analysing Self-Organisation

The organism was designed to self-organise data in a data stream based on similarities. In the last set of experiments, we emphasised on the anomaly detection capabilities of the organism using a dataset with one class of benign data and another class of malign data. In the next two sets of experiments we investigate further the datawise self-organisation within the organism using two datasets containing three

classes of data linearly separable from each other (i.e. no malign data class and therefore no class should only generate danger signals). First we use a simple artificial dataset to illustrate an ideal dataset's directed organism's behaviour, then we use the UCI wine [6] dataset, providing a real case study with more noisy data.

6.2.1 Ideal Dataset

In this set of experiments we generate a dataset of three classes, compact, linearly separable from each other, and well separated from each other in space. The aim is to have the organism organise itself into three clusters, each holding data from just one class. Each class contains 50 samples of 3 real-valued vectors. The vectors are randomly generated around the three points (0.9, 0.2, 0.3), (0.1, 0.8, 0.1) and (0.2, 0.3, 0.9) within the squares centred in these points and of side size 0.2. Therefore the maximum distance between two points of a class is $\sqrt{0.2^2 \times 3} = 0.3464$. The minimum distances between any two elements of different classes are between classes 1 and 2 0.737, classes 1 and 3 0.6445 and classes 2 and 3 0.6658. We use a threshold of 0.4 to ensure that all data from a same class could bind together and that no data from two different classes could bind together. Experiments settings involves a universe, an organism, a time system, a waste recycling, an absorption (thus introduction rate is one), a cell recycling, an expelling system, 200 data systems. Each data system contains a decay and a split system. The organism initially contains 5 adhesion surfaces. Each experiment consisted of 20 runs, consisting of 3000 iterations with randomly picked data presented each iteration. Each class has the same introduction probability (i.e. 1/3 each).

Table 3 shows the results of various experiments where the amount of growth systems is changed. We vary this parameter as Eq. 4 shows that in this stochastic context the more growth systems, the more likely a cell which should bind to another one will actually do so. These results are computed by discarding the early computations (we discarded here the first 50000 computations) during which the organism grows to an adult (stable) state and stopping the experiments when the flow of data ends (thus not permitting organism's death from starvation).

We can observe that the amount of error is very low, but not null. Some data cannot bind which tend to mean they are (almost) single of their class in the organism and their probability p_{c_i} (as investigated in Section 5) could thus be increased. In this respect, as discussed in Section 5.2, increasing the data rate could lower even more the amount of danger signals, provided the organism has a sufficent amount of

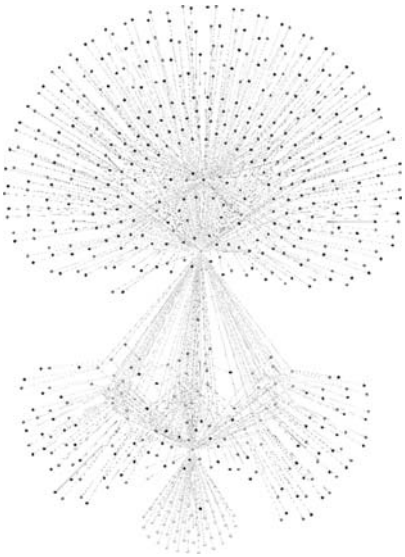
Table 3 This table shows in percentage the average and standard deviation of data from each class creating a danger signal for various amount of growth systems (n_g)

Exp	n_g	Class 1		Class 2		Class 3	
		mean	stdev	mean	stdev	mean	stdev
1	25	0.95	0.45	0.85	0.40	0.95	0.49
2	50	0.22	0.21	0.18	0.13	0.09	0.10
3	75	0.05	0.09	0.06	0.12	0.10	0.17
4	100	0.03	0.08	0.03	0.08	0.04	0.06

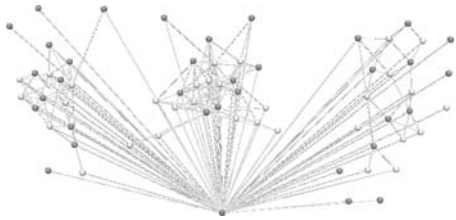
We empirically know that only cells from a same class can bind together in this setup, therefore only danger signals (i.e. cell not classified) can provide mistakes

growth systems. The data rate impact will be further investigated in the next set if experiments using the wine dataset.

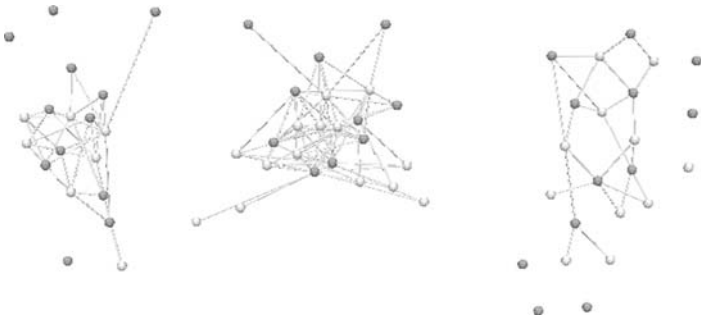
Figure 7 shows a snapshot of the organism visualised during computation in a run of experiment 2 (taken at a mature state at about 50000 computations). The spheres are systems and the links between them represent hierarchy (scopes). A system encompassing another one thus has a link from itself to the encompassed system. Figure 7a figure shows the whole program structure. To analyse the organisation of the organism itself Fig. 7b shows the structure of the organism and Fig. 7c only shows



(a) Whole Program



(b) Cells and Adhesion surfaces within the organism



(c) Cells and Adhesion surfaces only

Fig. 7 Organism's inner organisation over a run of 3000 samples on the ideal dataset. The spheres are the systems and the links represent the ownership of a system to a super system. **a** shows a visualisation of the whole program, **b** only shows the organism system (from where all links start) and its adhesion surfaces (*light grey*) and cells (*grey*), and **c** shows the clusters by only displaying the cells and adhesion surfaces. The cells on their own have not yet been linked to any existing cluster

the cells and the adhesion surfaces, clearly revealing the self-organisation in three distinct organs.

The visualisation and the results presented in Table 3 are respectively telling us that three distinct classes were found and that their elements were identified with accuracy.

6.2.2 UCI Wine Dataset

The previous set of experiments highlighted how the organism works, organising itself with respect to its data flow, but the data stream was an ideal (i.e. easy to organise) stream and datasets taken from real cases tend to have more noisy and sparse data. To investigate further the self-organisation in the organism this new set of experiments uses the UCI Wine dataset [6]. This set contains three classes of respectively 59, 71 and 48 instances, linearly separable from the two others and standing for three types of wine grown in Italy. Each sample is a 13 real-valued vector. All input values were normalised to lie within the [0,1] interval.

The aim is to explore how effectively the organism can grow so that the three classes of data are coherently organised in its structure. (It should be recalled that the aim is not to produce a clustering algorithm, but to provide a way of organising data such that a separate AIS can exploit the coherent tissue structure.) The ideal data stream in Section 6.2.1 was compact, with no noise, distributed over three dimensions. Here the data set comprises thirteen dimensions and is made of real data measures, therefore making the organisation more complicated and requiring a more precise analysis.

From the set of experiments in Section 6.1.1 we could observe that the threshold is the most crucial parameter, assuming the other parameters are set to values not too restrictive (e.g. cell's lifespan to short, data introduction rate too slow, amount of growth systems too low). The threshold is indeed directly linked to the precision of cells similarities. Therefore we give particular attention to this very parameter. However, as seen in Eqs. 7 and 8, the quantity of data present at a time to compare them is important, therefore we also investigate several introduction rates to observe the impact on the organism's behaviour.

Again all experiments were run 20 times, with each run consisting of 3000 iterations with randomly picked data presented each iteration. Each class has the same introduction probability (i.e. 1/3 each). All experiment settings involve a universe, an organism, a time system, a waste recycling, an absorption, a cell recycling, an expelling system, respectively 200, 400 and 600 data systems for introduction rates of 1, 2 and 3. Each data system contains a decay and a split system. The organism initially contains 5 adhesion surfaces.

Table 4 shows the results of various tunings that were used, disclosing in percentage the amount of danger signals emitted for each class. However an indication of how well the data is organised within the organism would be useful. We therefore also measure the amount of unwanted bonds (referred to as "bridges") within the organism. A link between a cell of a class i and a surface is considered as a bridge if the surface binds more cells of another class j together than of class i. Bridge evaluation is done when cells die and is given for each class in percentage of the total amount of connections cell-surface. As before, all results are computed by discarding the early computations (we discarded here the first 50000 computations) during

Table 4 This table shows in percentage the average danger signals and bridges (links between cells of a given class and surfaces binding together more cells of another class) per class from the wine dataset for various data introduction rates (d_r), threshold (τ) and amount of growth systems (n_g) values

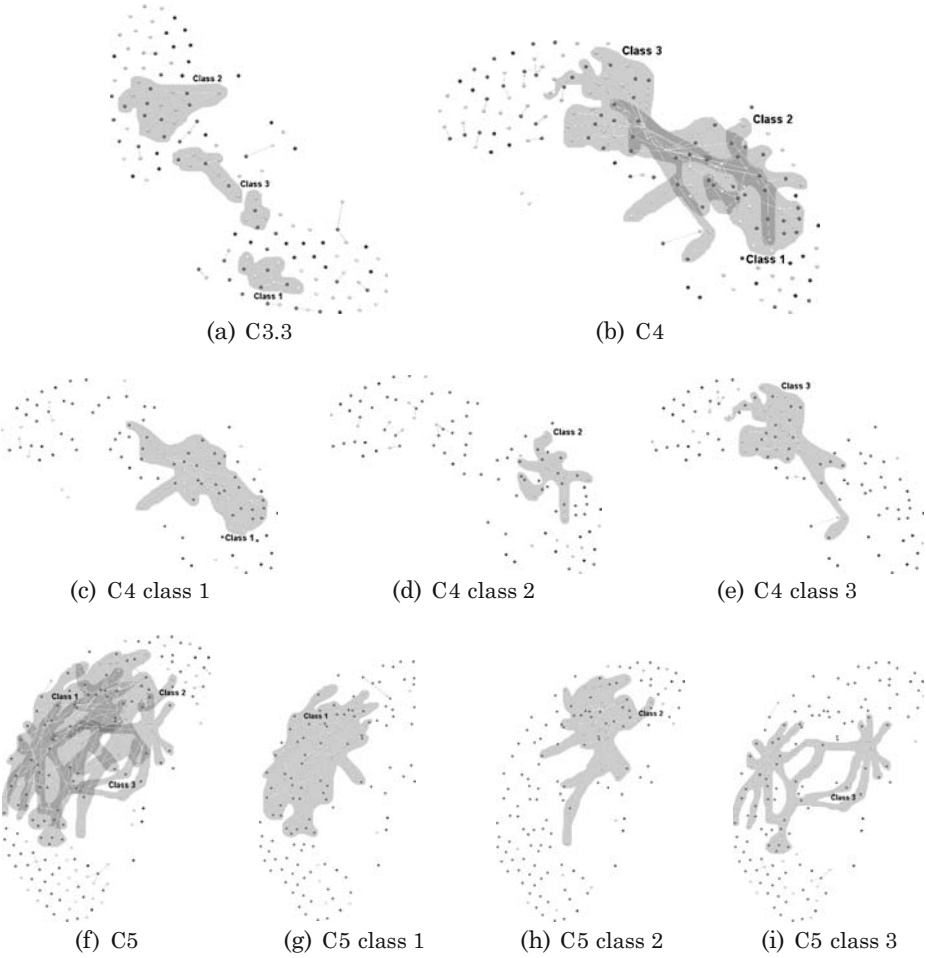
Exp	d_r	τ	n_g	Class 1		Class 2		Class 3	
				Danger	Bridge	Danger	Bridge	Danger	Bridge
A1	1	0.2	50	100.00	0.00	100.00	0.00	100.00	0.00
A2	1	0.3	50	100.00	0.00	100.00	0.00	100.00	0.00
A3	1	0.4	50	100.00	0.00	100.00	0.00	100.00	0.00
A3.3	1	0.43	50	88.23	0.56	92.40	0.49	91.74	0.95
A3.5	1	0.45	50	36.88	2.37	61.97	2.66	59.08	3.99
A3.7	1	0.47	50	29.20	3.68	51.74	4.49	48.03	6.69
A4	1	0.5	50	9.44	8.85	30.05	10.01	24.69	14.38
A5	1	0.6	50	1.39	23.78	9.93	36.65	3.57	36.14
A6	1	0.7	50	0.43	31.95	1.78	47.09	0.69	40.87
A7	1	0.8	50	0.05	37.20	0.41	44.60	0.37	39.39
B1	2	0.2	50	100.00	0.00	100.00	0.00	100.00	0.00
B2	2	0.3	50	100.00	0.00	100.00	0.00	100.00	0.00
B3	2	0.4	50	48.15	0.52	72.48	0.56	71.68	0.58
B3.3	2	0.43	50	30.30	1.14	58.66	1.60	54.46	1.99
B3.5	2	0.45	50	20.05	2.52	48.76	2.73	43.46	3.72
B3.7	2	0.47	50	14.21	3.53	40.21	4.56	33.68	5.79
B4	2	0.5	50	9.00	6.23	29.69	7.03	23.28	10.56
B5	2	0.6	50	1.72	19.79	10.07	26.26	3.60	30.35
B6	2	0.7	50	0.33	28.15	1.80	41.94	0.66	37.41
B7	2	0.8	50	0.09	33.44	0.43	42.02	0.37	37.47
C1	3	0.2	50	100.00	0.00	100.00	0.00	100.00	0.00
C2	3	0.3	50	100.00	0.00	100.00	0.00	100.00	0.00
C3	3	0.4	50	44.98	0.55	71.16	0.58	69.95	0.62
C3.3	3	0.43	50	38.60	1.14	64.76	1.24	62.43	1.45
C3.5	3	0.45	50	24.21	1.92	53.82	2.32	50.31	2.82
C3.7	3	0.47	50	18.85	2.70	47.16	3.18	41.58	4.65
C4	3	0.5	50	13.31	4.92	34.64	5.20	30.85	7.66
C5	3	0.6	50	2.97	15.38	13.29	19.68	5.93	23.59
C6	3	0.7	50	0.58	23.32	2.91	34.99	1.14	32.01
C7	3	0.8	50	0.10	28.90	0.49	37.29	0.33	33.51
C3a	3	0.4	75	42.65	0.42	68.00	0.57	65.68	0.80
C3b	3	0.4	100	33.30	0.67	57.57	0.79	54.64	0.69

which the organism grows to an adult (stable) state and stopping the experiments when the flow of data ends (thus not permitting organism's death from starvation).

From the results, we can observe that the amount of danger signals and the amount of bridges vary in opposite directions. Increasing the threshold makes it easier for cells to bind together, making less likely the emission of danger signals, but also aggregates data less strictly thus increasing the amount of bridges, and vice versa. Looking at the threshold, varying it little can have a significant impact on the binding results, particularly here between 0.4 and 0.5. However the data introduction rate is also crucial, as a low rate does not provide enough data for the organism to function properly, as seen in Eqs. 7 and 8, and as shown by experiments B3 and C3 compared to experiment A3. The amount of danger signals can also be lowered by adding

more growth systems into the system, as discussed in Section 5.2, and shown by experiments C3, C3a and C3b.

Finally, another effect of a low threshold, or slow data introduction rate, only visible for individual runs, is that the organism might die at birth. The organism then does not manage to bind any cell to others at its early stage (if the datastream provides no similar enough data), leading to the recycling of cells into danger signals



(a) C3.3 (b) C4

(c) C4 class 1 (d) C4 class 2 (e) C4 class 3

(f) C5 (g) C5 class 1 (h) C5 class 2 (i) C5 class 3

Fig. 8 Organism's inner organisation over a run of 3000 samples on the UCI wine dataset with setup of experiments C3.3, C4, and C5. The spheres are the systems and the links represent the ownership of a system to a super system. This visualisation highlights the organisation of the cells classwise. Within clusters cells are in *light grey* and surfaces are *grey*. Danger signals are outside in *dark grey*. The three setups illustrate three different behaviours. **a** displays an organisation with few bonds and many danger signals due to a strict threshold. **b** displays a less strict organisation where we can observe data from different classes binding in some places. **c**, **d**, and **e** show the same display as **b** but with data from respectively only class 1, 2 or 3. **f** illustrates the impact of a large threshold, gathering large amounts of data together with few danger signals but creating many bridges. **g**, **h**, and **i** show the same display as **f** but with data from respectively only class 1, 2 or 3

and the death of the initial adhesion surfaces. From then on no adhesion surface can be created and the organism is dead. It is what is happening in all the 20 runs in experiments A1-3, B1-2, C1-2 and is responsible for the 100% of danger signals. In these cases the threshold is too strict, the introduction rate too slow, or a combination of both, to bind things together. B3 and C3 compared to A3 indeed show, as discussed in Section 5.2, that the rate has an impact. However it also occurs in other experiments. This happened 13 times in experiment A3.3, 3 times in experiments A3.5, A3.7, B3, C3a, 2 times in experiment C3.3, and once in experiment B3.3 and C3b (so the results for the experiments where it happens rarely are slightly less good as they would be without these runs). This early death is less and less likely to happen as more data is sent into the organism and as the threshold is increased. However, note that it remains stochastic as it occurred in C3.3 and not in C3, even though it was more likely to happen in C3 than in C3.3.

Figure 8 shows visualisations during computation (taken at mature states at about 50000 computations) of the configurations from experiments C3.3, C4 and C5. Figure 8a illustrates the organism's behaviour with a strict threshold ($\tau = 0.43$) where we can observe an organisation with few bonds and many danger signals. Figure 8b, c, d, e displays a less strict organisation ($\tau = 0.5$). We can observe that data from different classes start to bind in some places. Figure 8f, g, h, i illustrates the impact of a large threshold ($\tau = 0.6$), gathering large amounts of data together with few danger signals but creating many bridges. Looking at these three configurations we can clearly observe the impact of the threshold. As it increases the organism has more and more links, creating bigger and more interconnected clusters, but data from different classes also get to bind more and more together.

Overall, looking at the experiments in Table 4 and visualisation in Fig. 8, class 1 is clearly easier to classify than class 2 or 3. The classification results for class 1 are better and the visualisation shows that data aggregate better in class 1. Class 1 set is therefore more compact than the other sets, and would thus consistently have a compactness factor higher than the other classes, as described in Eq. 8.

7 Discussion

As the organism is designed to grow to match the data rate, such a program is therefore able to cope with various (unexpected) parameter changes, self-(re)organising with the data flow, and providing information over time regarding detected potentially abnormal data items. When used in conjunction with an artificial immune algorithm, the good accuracy of detection (albeit with a high false positive rate), and the automatic organisation of similar data, binding them together, should enable excellent performance overall. While temporal information is currently lost because of the stochastic computation of SC, this could be added as additional features of data items, enabling self-organising according to similar timings in addition to data values.

One advantage of SC is the simplicity of modelling new stochastic systems, so an immune algorithm could be added to this model by simply adding two or three new types of system (e.g. B-cell, T-cell, antibody) that would then automatically interact with the existing tissue systems. Another valuable advantage of using SC in our approach is the fault-tolerance and self-repair ability an SC model can naturally have, as investigated in [17]. Having robust software can indeed be an important

feature in network security to ensure the program can survive even severe damage provoked for instance by hacking.

8 Conclusion

In this paper we introduced and developped the notion of artificial metabolism using systemic computation to create an organism for organising data that is suitable for an artificial immune system. This work is inspired by Matzinger's Danger Theory and uses the notion of danger signals. Starting from scratch and working on-line our organism is able to aggregate data according to its similarities and can provide danger signals when cells die in an abnormal way for the current organism. Our organism proved to be able to detect anomalous UCI Breast Cancer data with better accuracy than in previous work [3]. The study of the evolution over time of our organism with various datasets showed that its inner organisation reflects the data distribution of the current flow. Also, previous results have shown that SC programming is very robust, with programs easily showing fault-tolerance and self-repair abilities even when undergoing severe damage [17]. Further work could investigate a dendritic cell algorithm [18] based innate immunity approach, and also investigate the introduction of adaptive immune response into the organism. The model could finally also be tested against real-time applications like controlling data flows for computer security.

References

1. Aickelin, U., Greensmith, J.: Sensing danger: innate immunology for intrusion detection. Elsevier information security technical report, pp. 218–227. Elsevier, Amsterdam (2007)
2. Matzinger, P.: Tolerance, danger and the extended family. Annu. Rev. Immunol. **12**, 991–1045 (1994)
3. Bentley, P.J., Greensmith, J., Ujjin, S.: Two ways to grow tissue for artificial immune systems. In: Proc. of the Fourth Intl. Conf. on Artificial Immune Systems (ICARIS 2005). LNCS 3627, pp. 139–152. Springer, New York (2005)
4. Bentley, P.J.: Systemic computation: a model of interacting systems with natural characteristics. Int. J. Parallel Emergent Distrib. Syst. **22**(2), 103–121 (2007)
5. Breast Cancer Wisconsin (Diagnostic) Dataset: Creator: Wolberg, W.H., Donor: Mangasarian, O. UCI machine learning repository. http://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic) (1992)
6. Wine Dataset: original owners: Forina, M., et al., PARVUS, Donor: Aeberhard, S. UCI machine learning repository. http://archive.ics.uci.edu/ml/datasets/Wine (1991)
7. Tempesti, G., Roggen, D., Sanchez, E., Thoma, Y.: A POEtic architecture for bio-inspired hardware. In: Proc. of the 8th Intl. Conf. on the Simulation and Synthesis of Living Systems (Artificial Life VIII), pp. 111–115. MIT, Cambridge (2002)
8. Thoma, Y., Tempesti, G., Sanchez, E., Moreno Arostegui, J.-M.: POEtic: an electronic tissue for bio-inspired cellular applications. BioSystems **76**, 191–200 (2004)
9. Wallenta, C., Kim, J., Bentley, P.J., Hailes, S.: Detecting interest cache poisoning in sensor networks using an artificial immune algorithm. J. Appl. Intell. doi:10.1007/s10489-008-0132-0 (2008)
10. von Neumann, J.: The Theory of Self-reproducing Automata. University of Illinois Press, Champaign (1966)
11. Wolfram, S.: A New Kind of Science. Wolfram Media, Champaign (2002)
12. Holland, J.H.: Emergence, from Chaos to Order. Oxford University Press, Oxford (1998)
13. Adamatzky, A.: Computing in Nonlinear Media and Automata Collectives. Institute of Physics, Bristol (2001)

14. Arvind, D.K., Wong, K.J.: Speckled computing: disruptive technology for networked information appliances. In: Proc. of the IEEE Intl. Symposium on Consumer Electronics (ISCE'04), pp. 219–223. IEEE, Piscataway (2004)
15. Le Martelot, E., Bentley, P.J., Lotto, R.B.: A systemic computation platform for the modelling and analysis of processes with natural characteristics. In: Proc. of 9th Genetic and Evolutionary Computation Conference (GECCO 2007), pp. 2809–2816. ACM, New York (2007)
16. Le Martelot, E., Bentley, P.J., Lotto, R.B.: Exploiting natural asynchrony and local knowledge within systemic computation to enable generic neural structures. In: Proc. of 2nd International Workshop on Natural Computing (IWNC 2007), Nagoya, 10–12 December 2007
17. Le Martelot, E., Bentley, P.J., Lotto, R.B.: Crash-Proof systemic computing: a demonstration of native fault-tolerance and self-maintenance. In: Proc. of 4th IASTED International Conference on Advances in Computer Science and Technology (ACST 2008). ACTA, Anaheim (2008)
18. Greensmith, J., Aickelin, U., Cayzer, S.: Introducing dendritic cells as a novel immune-inspired algorithm for anomaly detection. In: Proc. of the Fourth Intl. Conf. on Artificial Immune Systems (ICARIS 2005). LNCS 3627, pp. 153–167. Springer, New York (2005)